University of Liège

Faculty of Applied Sciences
Department of Electrical Engineering & Computer Science

PhD dissertation

Machine Learning of Proxies for Power Systems Reliability Management in Operation Planning

by Laurine Duchesne

Advisor: Prof. Louis Wehenkel
July 2021

Jury members

Prof. Pierre Geurts (President)	University of Liège, Belgium;
Prof. Louis Wehenkel (Advisor)	University of Liège, Belgium;
Prof. Spyros Chatzivasileiadis	Technical University of Denmark, Denmark;
Prof. Bertrand Cornélusse	University of Liège, Belgium;
Dr. Efthymios Karangelos	University of Liège, Belgium;
Prof. Quentin Louveaux	University of Liège, Belgium;
Prof. Daniel Molzahn	Georgia Institute of Technology, United States;
M. Patrick Panciatici	RTE-France, France.

Acknowledgments

This thesis would not have been possible, but also not as interesting and enjoyable as it was, without all the persons I met during this journey and I would like to thank them here.

First and foremost I would like to thank my supervisor Louis Wehenkel, who introduced me to the thrilling and timely field of power systems applications. I would like to thank him for his availability, his thoughtful advices and valuable guidance during the fives years of this thesis, both during the research and when I was writing this manuscript. He was always there to guide me and encourage me when it was necessary. I also want to thank him for the many opportunities he offered me to travel, share our work and meet interesting people along the thesis.

Then I would like to thank Efthymios Karangelos who was always there to help me but also challenge me to carry out the best research during these five years. I will also always remember the nice moments we spent together with Louis in conferences or in Cambridge, especially the visit of Manchester, the trip to Lake District and all the moments we spent together, visiting or discussing while enjoying a beer or a cider.

I would also like to thank the members of the jury for their interest in this work, their careful reading of the manuscript and their valuable comments on this research.

A special thanks to Patrick Panciatici and the team of RTE for all the interesting discussions we had about research and the valuable inputs, and also to Patrick for his various comments to improve this dissertation.

I would like also to express my gratitude to Bertrand Cornélusse and Quentin Louveaux for our research discussions and their involvement for the research we carried out together.

I would also like to thank my colleagues in Montefiore, Antonio for our long discussions, his enthusiasm, but also his unfailing support when I needed help, Marie for all our (small?) breaks and laughs together, Antoine, Romain, Jean-Michel, Nicolas, Pascal, Anaïs for all the jokes, discussions and laughs together, Antoine again for organizing the PhD m'eatings with me, Sophie for your help and kindness, but also Michael, Pierre Geurts, Vân Anh, Gilles, Raphael, Matthia, Ulysse, Adrien Deliège, Anthony, Simon, Fred, Damien, Ioannis, Miguel, David, Jonathan, Sélim, Selmane, Laurent, Tom, Sami, Emeline, François, Pierre Sacré, Arnaud, Adrien Bolland, Renaud and all the others I forgot to mention here. Thanks to all of them I enjoyed the day-to-day work as a PhD student, the lunches, the coffee/tea breaks, the lab meetings and PhD m'eatings, and all the discussions we had about research or other subjects.

I would like to thank my friends for their support, with a special thanks to Romain and Elodie for their valuable proofreading of my manuscript.

Finally, this thesis would not have been possible without the unconditional love and support of my family, in the good and the less good moments. In particular, thanks to my parents and my sister Emilie for their help when I needed it, for all the nice meals to encourage me and jokes when I was not feeling well. I also have a particular thanks to Quentin. Thank you for being always there for me.

Contents

Abstract

Power systems reliability management aims at taking decisions ahead in time, from several years to a few minutes ahead, so as to facilitate the operation of the system over a future target horizon and ensure a continuous supply of electricity to end-users. As electricity is essential in our modern societies, this activity is critical. However, power systems are currently undergoing major changes. Among others, the increasing penetration of renewable energy, the liberalization of the electricity sector and the aging of the grid result in a large increase in the amount of uncertainties in power systems, complicating the task of the operators and calling for new methods for operation and planning, taking into account these uncertainties.

Focusing on the short-term operation planning context, that aims at taking decisions from several days to several hours ahead to ensure that adequate resources will be available in real-time operation to meet the electricity demand, we propose in this thesis new decision-making tools leveraging machine learning, Monte-Carlo simulations and optimization, to tackle the increasing uncertainties in power systems. In particular, we propose to exploit simplified models, called *proxies*, of the behavior of the operator in response to realization of uncertainties over the future target horizon considered. These proxies must be fast and yet accurate, in order to replace traditional heavy models of the behavior of the operator and allow one to anticipate the impact of a candidate operation planning decision over a very large number of possible future operating conditions in a short amount of time.

In this thesis, we propose a methodology to build these proxies with machine learning and we illustrate how they can be used for the two aspects of reliability management: reliability assessment (i.e. evaluating the anticipated outcomes of real-time operation) and control (i.e. selecting which decisions to commit).

More specifically, we develop a methodology based on supervised machine learning to build proxies of real-time operation and we show on a case study that such built proxies have good accuracies and are several orders of magnitude faster than traditional models of real-time operation.

Considering the reliability assessment problem, we combine a variance reduction technique called the control variates approach with our machine learnt proxies in order to speed up the estimation of the induced expected costs of real-time operation for a given candidate operation planning decision. We show that this method yields unbiased estimates of the expected costs of real-time operation while requiring a significantly smaller number of scenarios compared to classical Monte-Carlo techniques for a given target accuracy.

We then generalize this approach to several unseen candidate decisions to further help choosing among operation planning decisions. We show that this approach can be used to rank a list of candidate decisions according to their induced expected costs of real-time operation in order to identify good operation planning decisions.

Motivated by the reliability control problem, we propose to exploit input convex neural networks to build convex approximations of non-convex feasible domains of optimization problems and we demonstrate that the such learnt approximation can be expressed as a set of linear inequalities in a lifted space.

Finally, we review recent works applying machine learning techniques for reliability management, to showcase the potential of these techniques and the progress achieved to date.

Résumé

En gestion de la fiabilité des réseaux électriques, le gestionnaire du réseau de transport doit prendre des décisions quelques minutes à plusieurs années à l'avance pour se couvrir vis-à-vis des aléas qui peuvent être rencontrés ultérieurement et assurer l'approvisionnement continu en électricité des consommateurs finaux. Étant donné l'importance de l'électricité dans notre société, cette activité est essentielle. Cependant, les réseaux électriques subissent actuellement d'importants changements. Parmi ceux-ci, le développement des énergies renouvelables, la libéralisation du secteur de l'électricité et le vieillissement des infrastructures ont pour effet d'augmenter le niveau d'incertitudes dans les réseaux électriques, ce qui complique les prises de décision des gestionnaires de réseau. Il est donc nécessaire de développer de nouvelles méthodes pour la planification et la conduite des réseaux électriques.

Dans cette thèse, nous considérons le contexte de la préparation à la conduite du réseau, qui consiste à prendre des décisions quelques heures à quelques jours à l'avance pour s'assurer que les ressources nécessaires seront disponibles en conduite en temps-réel pour satisfaire la demande en électricité, et nous proposons des outils d'aide à la décision exploitant les techniques d'apprentissage automatique, les simulations de Monte-Carlo et l'optimisation pour s'adapter aux incertitudes croissantes dans les réseaux électriques. Nous proposons en particulier de construire et exploiter des modèles approchés, appelés *proxies*, de la réponse de l'opérateur aux réalisations des incertitudes (telles que la production renouvelable et la consommation) sur l'horizon de temps considéré. Ces *proxies* doivent être à la fois rapides et suffisamment précis que pour remplacer les modèles traditionnels assez lourds modélisant la réponse de l'opérateur, afin de permettre de simuler pour un très grand nombre de scénarios futurs la réponse de cet opérateur et ainsi estimer en très peu de temps l'incidence d'une décision prise lors de la préparation à la conduite sur la conduite future du réseau.

Dans cette thèse, nous proposons une méthodologie pour construire ces *proxies* avec de l'apprentissage automatique et nous illustrons comment ces *proxies* peuvent être exploités dans le cadre de la gestion de la fiabilité en préparation à la conduite, à la fois pour évaluer la fiabilité d'une décision candidate et pour déterminer une politique de décision assurant un niveau requis de fiabilité tout en optimisant en espérance les objectifs économiques.

Plus spécifiquement, nous développons une méthodologie qui utilise l'apprentissage automatique supervisé pour construire des *proxies* modélisant la conduite en temps-réel et nous montrons que ces *proxies* sont plus rapides de plusieurs ordres de grandeur que les modèles traditionnels tout en ayant une précision suffisante.

Ensuite, afin d’évaluer la fiabilité d’une décision candidate, nous proposons une méthode combinant les simulations de Monte-Carlo, une technique de réduction de la variance basée sur une variable de contrôle et nos *proxies* afin d’estimer l’espérance des coûts de conduite en temps-réel induits par la décision candidate. Nous montrons que l’approche proposée permet d’obtenir des estimateurs non-biaisés de l’espérance de ces coûts induits, tout en réduisant le nombre de scénarios nécessaires comparé à une méthode de Monte-Carlo classique pour obtenir un même niveau de précision.

Nous généralisons ensuite cette approche pour évaluer l’incidence de plusieurs nouvelles décisions candidates sur la conduite future, afin de choisir parmi ces décisions lesquelles sont les plus appropriées. En particulier, nous montrons que cette approche basée sur les *proxies* permet de classer ces décisions par ordre croissant d’espérance de coûts de conduite en temps-réel induits et ainsi d’identifier les décisions minimisant l’espérance de ces coûts.

Etant légèrement différente de nos autres contributions, mais tout de même motivée par le problème d’optimisation de la fiabilité, nous proposons une méthode basée sur des réseaux de neurones dont la sortie du réseau est une fonction convexe des entrées pour construire des approximations convexes d’ensembles admissibles non convexes de problèmes d’optimisation et nous démontrons que ces approximations peuvent être exprimées par un ensemble de contraintes d’inégalité linéaires.

Finalement, nous examinons les récents travaux exploitant des techniques d’apprentissage automatique dans le cadre de la gestion de la fiabilité des réseaux électriques, afin d’illustrer le potentiel de ces techniques pour la gestion de la fiabilité et de montrer les progrès déjà réalisés.

List of acronyms

AC	Alternating Current
CMC	Crude Monte-Carlo
DC	Direct Current
DSO	Distribution System Operator
ED	Economic Dispatch
ET	Extremely Randomized Trees
ICNN	Input Convex Neural Network
***k*NN**	*k* Nearest Neighbors
KRR	Kernel Ridge Regression
LSh	Load Shedding
MDI	Mean Decrease Impurity
MDP	Markov Decision Process
MILP	Mixed Integer Linear Programming
MINLP	Mixed Integer Non Linear Programming
MIP	Mixed Integer Programming
ML	Machine Learning
MSE	Mean Squared Error
NN	Neural Network
OPF	Optimal Power Flow
PF	Power Flow
PICNN	Partially Input Convex Neural Network
PMU	Phasor Measurement Unit
QC	Quadratic Convex
RBF	Radial Basis Function

RD	ReDispatch
ReLU	Rectified Linear Unit
RF	Random Forest
RMAC	Reliability Management Approach and Criterion
RR	Ridge Regression
RT	Real-Time
SCOPF	Security-Constrained Optimal Power Flow
SDP	Semi-Definite Programming
SMC	Stacked Monte-Carlo
SOC	Second Order Cone
SVM	Support Vector Machine
SVR	Support Vector Regression
TSO	Transmission System Operator
UC	Unit Commitment
WC	Wind Curtailment

1 Introduction

1.1 Motivations

In our modern societies, electricity plays a key role as numerous activities depend on it. In many cases, an interruption of these activities results in serious economic and social consequences. It is thus primordial to ensure a continuous supply of electricity from the *electric power systems*.

However, power systems are currently undergoing major changes, which pose new challenges to ensure the reliability of these systems.

One of these changes is the rapid increase in the penetration of renewable energies in the energy mix, as we transition toward cleaner energy to tackle climate change. In Europe, it has more than doubled between 2004 and 2019 (see Figure 1.1) and it is expected to continue to increase in the future, as the European Union plans to be climate neutral by 2050, in particular through the decarbonization of the energy sector [European commission, 2019]. As a consequence, the electricity generation is becoming an exogenous uncertainty for power systems operators. Indeed, while most thermal generation units based on fossil fuel are controllable, it is not the case of generating units based on renewable energies such as wind and sun, which are weather-dependent, uncertain and highly intermittent. Furthermore, contrarily to the large conventional power plants, these renewable generators are also connected to the medium- and low-voltage distribution systems, modifying the power flow patterns and causing voltage violations.

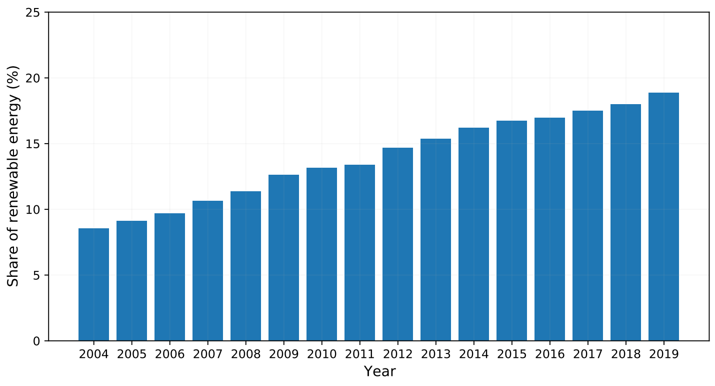


Figure 1.1: Evolution of the share of renewable energy in gross final energy consumption in the Europe of 28 (data from [Eurostat, 2021]).

Coupled to the increasing penetration of renewable generation, the liberalization of the electricity sector and the aging of the power systems infrastructures increase even more the amount of uncertainties in power systems. The unbundling of the electricity sector to foster competition creates more uncertainties and possible congestions in the power systems, while the aging of the grid results in more maintenance and increased probability of failures of components.

Moreover, the dependency on the continuous supply of electricity is expected to increase in a near future due, for instance, to the electrification of some sectors such as the transport (e.g. electric vehicles) or heating (e.g. heat pumps) sectors in the context of the energy transition.

All these changes impact both the organization and the infrastructures of power systems, leading to potential service interruption. Currently, large interruptions of power supply such as blackouts are relatively rare events but they may become more frequent in the future. For instance, this winter 2021, Europe almost experienced a blackout [Bloomberg, 2021] while in Texas, millions of households were left without electricity, and thus for some without heat in freezing temperature, due to severe winter storms [Wikipedia, 2021].

These challenges make it necessary to develop new operation and planning methods to handle these uncertainties and ensure the reliability of the system.

1.2 Context

In order to facilitate the operation of the system and ensure the continuous supply of electric power with as few interruptions as possible, operators need to take decisions ahead in time. For instance, building a new transmission line is planned years ahead, the maintenance of assets several months in advance and the starting-up/shutting-down time of generating units several hours ahead. Taking decisions to ensure the continuous functioning of the system over a given time horizon (from a few minutes ahead to several years) is the aim of *reliability management*. It is typically divided into assessment and control (see Figure 1.2). Reliability assessment consists in anticipating the impact of a decision on future time horizons while reliability control consists in selecting which decisions to commit to ensure the continuous supply of electricity in the considered time period.

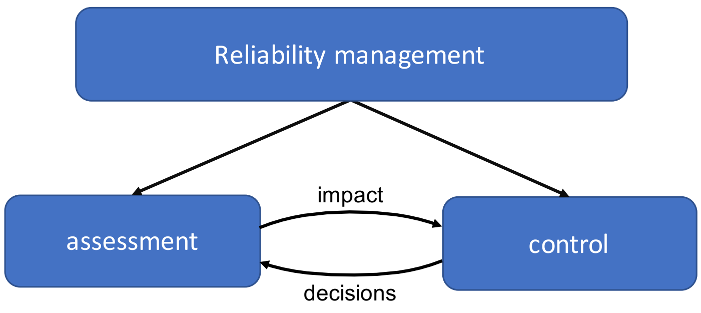


Figure 1.2: Decomposition of reliability management between assessment and control.

The types of decisions addressed in reliability management can be divided into several contexts and their corresponding time horizons. This is represented in Figure 1.3. In this thesis, we focus on *operation planning* in transmission systems, that aims at taking decisions from several months to several hours ahead to ensure that adequate resources will be available in real-time operation to meet the electricity demand. More precisely the focus of this dissertation is the short-term operation planning context, from several days to a few hours before real-time operation. Decisions in this context typically involve postponing the planned maintenance of an asset in the system, rescheduling generating unit outputs, calculating network capacities for market coupling, or acquiring flexible resources (reserves) for real-time operation.

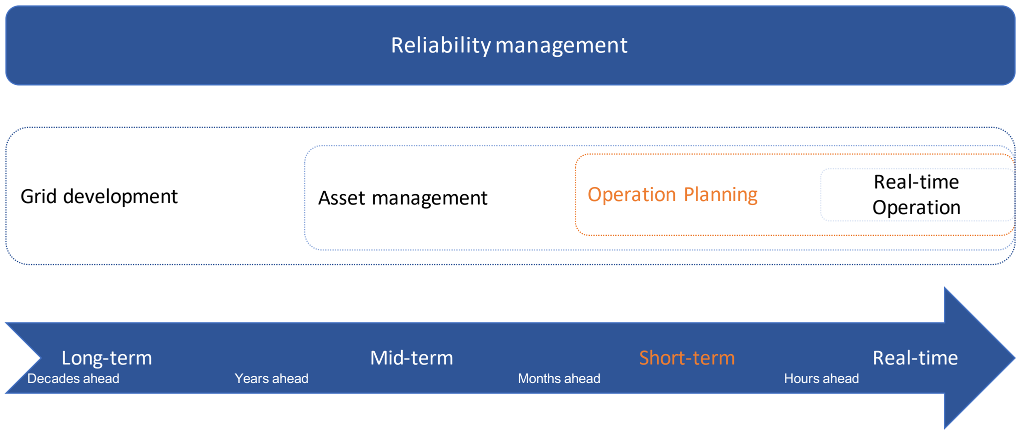


Figure 1.3: Different contexts of reliability management and corresponding time horizons. We focus on the operation planning context in transmission systems.

Traditional approaches for operation planning in power systems are based on a single (or a few) 'most likely' forecast along the considered look-ahead time horizon. For instance, in day-ahead operation planning, the operator considers a single forecast of demand and renewable generation along the next day to decide if a given maintenance should be postponed or not to maintain a sufficient level of reliability for the system.

However, with the increasing uncertainties in power systems, the observed real-time realizations deviate further and further away from the forecast, making it more difficult for the operator to ensure the continuous functioning of the system with the traditional approaches. Therefore new methods for power systems operation planning, taking into account uncertainties, are needed to tackle this problem.

Meanwhile, the field of artificial intelligence, and *machine learning* in particular, is evolving rapidly. Indeed, the continuous growth of computational power and data makes it a very active field of research, such that new techniques and ideas are available, broadening the range of applications of machine learning and allowing its use in complex real-world problems. There are therefore many possibilities to leverage these advances in the field of power systems.

In light of these, we see an opportunity to approach the problem of increasing amount of uncertainties in power systems operation planning by exploiting machine learning techniques.

1.3 Contributions

The objective of this thesis is to propose new decision-making tools for reliability management in the context of operation planning. These tools leverage *machine learning*, *optimization* and *Monte-Carlo simulations*, in order to deal with the increasing amount of uncertainties in power systems.

To take into account uncertainties, we choose to plan operation over a representative set of possible future operating conditions (instead of one 'most likely' trajectory), while modeling the way the operator would respond along these trajectories, as is schematized in Figure 1.4. This set of possible future operating conditions, also called scenarios or trajectories, must be large enough to correctly span the range of uncertainties in operation planning. The purpose would be to choose operation planning decisions making the compliance with real-time reliability targets feasible (with high enough probability) while minimizing (the mathematical expectation of) operating costs.

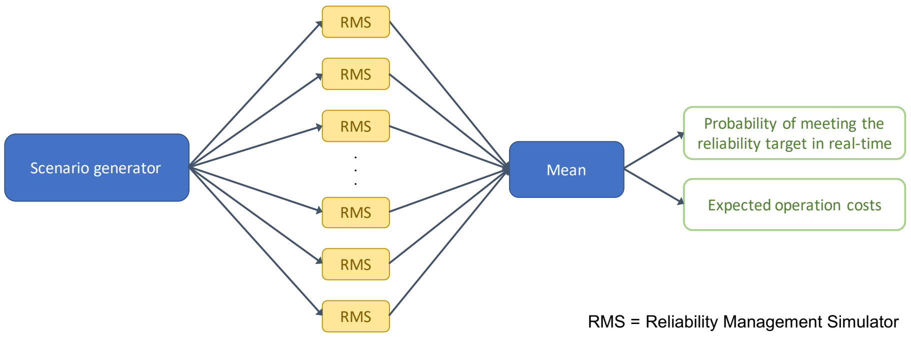

Scenario generator
RMS
RMS
RMS
RMS
RMS
RMS
Mean
Probability of meeting the reliability target in real-time
Expected operation costs
RMS = Reliability Management Simulator

Figure 1.4: Simulation of power systems operation along the considered time horizon for a representative set of possible future scenarios, in order to estimate the probability to meet the feasibility target in real-time and the expected operation costs.

For this approach, it is necessary to model in a suitable way the reliability management strategy over many time steps and many look-ahead scenarios. However, modeling precisely the response of the operator to the realization of uncertainties over the time horizon considered is challenging and linked to a computational burden. Indeed, it often consists in solving non-convex optimization problems with a large number of variables for multiple time steps. As such, it is therefore not suitable to be used on many scenarios in short-term operation planning, since only several hours are available in this context to tackle the reliability assessment and control problems.

In this dissertation, we therefore propose a method to build simplified models, that we call *proxies*, of the response of the operator to realizations of uncertainties in shorter-term contexts. These proxies must be *fast* enough to be used in the operation planning context, while being sufficiently *accurate* to correctly commit and assess operation planning decisions. We then illustrate how these proxies can be used for reliability assessment and control in short-term operation planning, taking into account uncertainties.

Our contributions are the following ones.

We first propose a methodology to automatically build proxies of the real-time reliability management response to the realization of uncertainties, with supervised machine learning. We test different learning algorithms to identify the most suitable ones for this application and we verify that these proxies are faster and sufficiently accurate to replace a more traditional model of real-time reliability management response.

We then propose to exploit these proxies for probabilistic reliability assessment in short-term operation planning. In particular, we propose to combine a variance reduction technique called *the control variates approach* with proxies in Monte-Carlo simulations to evaluate the expected outcome of real-time operation of a candidate look-ahead decision, given the probability distribution of uncertainties. Combining the proxies with the control variates approach allows to speed up Monte-Carlo simulations, while still computing unbiased estimates of the expected outcome of real-time operation.

At the boundary between reliability assessment and control, we propose a methodology generalizing the probabilistic reliability assessment approach to several candidate look-ahead decisions. We then exploit the approach to rank a list of candidate look-ahead decisions and identify 'good' operation planning decisions, based on the estimation of the expected outcome of real-time operation (in this case the expected cost of real-time operation).

Concerning the reliability control problem, an optimal look-ahead decision can be computed by solving a non-linear non-convex optimization problem. To simplify such type of problems, we propose a method based on machine learning to learn convex piecewise linear approximations of feasible sets of optimization problems, in such a way that the constraints of the approximated problem are expressed by a series of linear inequalities. We illustrate the performance of this method on toy examples.

Finally, we also realize a survey of the recent works applying machine learning techniques in the context of power system reliability assessment and control, to showcase both the progress achieved to date and the important future directions for further researches.

To sum up, the contributions of this thesis are the following ones:

- A methodology based on machine learning to build proxies of real-time operation.
- An approach leveraging a variance reduction technique and the proxies of real-time operation for probabilistic reliability assessment in operation planning.
- A generalization of the probabilistic reliability assessment approach to several look-ahead decisions and its exploitation to rank candidate decisions and identify the 'good' ones.
- A method based on machine learning to build convex piecewise linear approximations of any domain, that could be used to learn convex approximations of the feasible sets of non-convex optimization problems.

- A survey of recent works applying machine learning for power systems reliability management in general.

1.4 Outline of the manuscript

This manuscript is organized in four parts.

The first part describes both the context of this thesis, i.e. reliability management in power systems, and the main methods applied, i.e. machine learning.

Chapter 2 introduces reliability management in power systems to set the background of this thesis. In particular, it describes reliability assessment and control, the main uncertainties in power systems and the different contexts of reliability management, with a focus on operation planning and real-time operation. It also presents some reliability criteria to assess the reliability level of the system and the main tools used in this thesis for assessment and control in operation planning.

Chapter 3 presents the main concepts of machine learning and in particular supervised machine learning, on which our contributions are based. In particular, it describes the different types of problems, introduces a probabilistic formalization of supervised learning, and explains how to practically learn a model and assess its performances. It also describes the different learning algorithms used in this dissertation.

Both chapters 2 and 3 are intended for readers not familiar with these topics.

Chapter 4 reviews recent works applying machine learning in the context of power systems reliability management. In this manuscript we limit ourselves to (mostly) static reliability management, since it is the focus of this dissertation, but our survey paper [Duchesne et al., 2020b] also reviews recent works in dynamic security. This chapter is organized considering the power systems tools for reliability assessment and control that can be enhanced/replaced with machine learning models. It ends with a discussion presenting some challenges for machine learning approaches in power systems reliability management, as well as some future research directions we identified.

The second part of this manuscript includes all our contributions regarding the proxies. Each one of these contributions corresponds to a paper that has already been published.

Chapter 5 describes our methodology to build proxies of real-time operation with machine learning and applies this methodology with different supervised learning algorithms on a case study to assess the validity of the method and identify the most suitable learning algorithms for this application.

Chapter 6 presents our approach for probabilistic reliability assessment of a candidate look-ahead decision. It introduces the theory of Monte-Carlo simulation and the control variates approach, then describes our methodology to exploit the proxies with the control variates approach. A case study is presented to showcase that the method is faster than classical Monte-Carlo simulations while also computing unbiased estimates.

Finally, *chapter 7* expands the approach of chapter 6 to several candidate look-ahead decisions. It presents a methodology to build a dataset of several look-ahead decisions and real-time scenarios and a method based on the proxies to rank these look-ahead decisions. A case study is presented to show that this approach based on proxies can be used to rank a list of candidate look-ahead decisions and identify the good ones.

The third part, while being motivated by the reliability control in operation planning problem, is slightly different.

Chapter 8 presents a method based on supervised learning to build convex approximations of non-convex feasible sets of optimization problems and illustrates the approach on two-dimensional toy problems.

Finally, the last part of this dissertation, *chapter 9*, concludes this manuscript and outlines possible directions of future researches.

1.5 Publications

Publications that are directly related to this work include:

- Duchesne, L., Karangelos, E., and Wehenkel, L. (2017). Machine learning of real-time power systems reliability management response. In *2017 IEEE Manchester PowerTech*, pages 1–6. IEEE;

- Duchesne, L., Karangelos, E., and Wehenkel, L. (2018). Using machine learning to enable probabilistic reliability assessment in operation planning. In *2018 Power Systems Computation Conference (PSCC)*, pages 1–8. IEEE;

- Duchesne, L., Karangelos, E., Sutera, A., and Wehenkel, L. (2020a). Machine learning for ranking day-ahead decisions in the context of short-term operation planning. *Electric Power Systems Research*, 189:106548;

- Duchesne, L., Karangelos, E., and Wehenkel, L. (2020b). Recent developments in machine learning for energy systems reliability management. *Proceedings of the IEEE*, 108(9):1656–1676;

- Duchesne, L., Louveaux, Q., and Wehenkel, L. (2021). Supervised learning of convex piece-wise linear approximations of optimization problems. Submitted for publication.

During the course of this thesis, a collaboration has also led to the following publication, which is not discussed within this dissertation:

- Duchesne, L., Cornélusse, B., and Savelli, I. (2019). Sensitivity analysis of a local market model for community microgrids. In *2019 IEEE Milan PowerTech*, pages 1–6. IEEE.

Part I

Power systems reliability management and machine learning

2

Background in power systems reliability management

Overview

This chapter introduces the main concepts in power systems reliability management, as a background for the work developed in the following chapters of this manuscript. In particular, it defines reliability management and its subdivision between reliability assessment and control, it describes the threats and disturbances faced in reliability management, the different contexts and corresponding temporal horizons. It also presents some reliability criteria and the tools used in this thesis to tackle operation planning.

2.1 Power system context

Power systems are infrastructures that supply and deliver electricity to end-users. They exist since more than one hundred years and are one of the largest and most complex man-made systems. They are divided in many sub-systems, interacting with each other at different levels, and are composed of numerous facilities (e.g. power plants) and components (e.g. overhead lines, cables, transformers, circuit breakers, loads). They also involve a large number of stakeholders, such as producers, operators, consumers, energy suppliers, traders, regulators, etc.

A schematic representation of a power system is presented in Figure 2.1. As can be seen on this figure, the delivering system is typically divided into two parts: the transmission system and the distribution sub-systems.

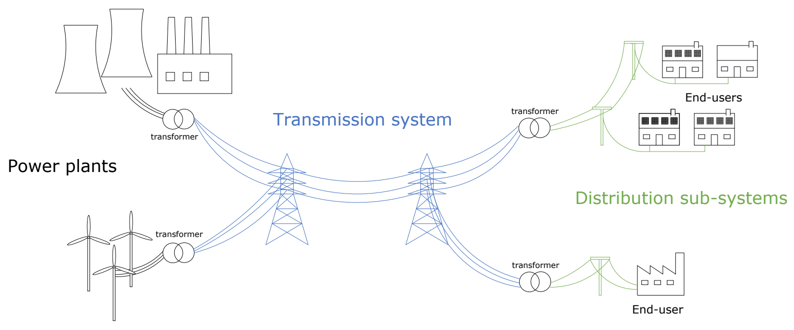


Figure 2.1: Schematic representation of a power system. Inspired from [U.S.-Canada Power System Outage Task Force, 2004, p. 5].

The *transmission system* carries the power at high voltages (from 30 kV to 380 kV in Belgium [Elia, 2021]) from the large power plants to distribution sub-systems via load substations. It is not directly connected to the final electricity consumers, except for some large industrial end-users. The transmission network has a meshed structure to limit as much as possible the impact of a failure in the system and is operated by the Transmission System Operators (TSOs).

The *distribution sub-systems* distribute the electricity from the transmission system to the final consumers. Since recently, with the development of distributed energy resources such as solar panels, the distribution sub-systems are also directly connected to small generators and micro-grids. The voltage level goes from mid-voltage to low voltage, typically from 0.4kV to 30kV in Belgium. The networks are generally radial as less end-users are impacted if there is a failure in the system and are operated by the Distribution System Operators (DSOs).

In this dissertation, we consider the point of view of a TSO to study the reliability of power systems.

2.2 Reliability management in power systems

Reliability quantifies the ability of a system to perform its intended functions without failure, under given conditions. In case of power systems, it quantifies the ability of the system to continuously deliver electricity to end-users, in spite of the various disturbances and threats that can act on it. The main disturbances and threats in transmission systems can be for instance a failure of an overhead line or a large difference between the predicted demand and the true demand.

2.2.1 Reliability, security and resilience

Before defining reliability management, it is important to clarify the notion of reliability with respect to resilience and security.

Security is the ability of the system to respond to sudden disturbances arising in the system such as electric short circuits or non-anticipated loss of system components without major service interruptions [Billinton and Allan, 1996]. In this dissertation, reliability and security are used interchangeably.

Resilience is "the ability of a power system to recover quickly following a disaster" [Panteli and Mancarella, 2015]. It is thus different from reliability in the sense that a resilient system is able to recover from "extraordinary and high-impact low probability events" such as extreme weather events. Black-outs in these cases are almost always unavoidable but what is important is to reduce the impacts of these extreme events (typically by avoiding the propagation of black-outs), as well as the ability of the system to recover rapidly from them. Reliability management, on the other hand, deals with more common threats and disturbances and must avoid as much as possible service interruption.

2.2.2 Definition of reliability management

Given the large number of end-users impacted in case of a failure of the transmission system, reliability is of utmost concern for the operator. To ensure that the system is reliable enough, the TSO has to take decisions as early as necessary, from a few minutes ahead to several decades, to facilitate the operation of the system. One important constraint in the decision-making processes is that some decisions require time to be implemented and thus must be taken when the level of uncertainties is still large. Taking decisions in order to ensure, with a high level of confidence, the continuous operation of the system for a given time interval is the aim of *reliability management*. It addresses various types of decisions over different temporal horizons.

2.2.2.1 *Reliability criterion*

Reliable enough or high level of confidence are vague notions related to the level of reliability of the system. In practice, the minimum level of reliability is defined with a *reliability criterion*. Decisions are taken is such a way that the *probability* to meet the reliability criterion over the time horizon of consideration is high. The most common reliability criterion is the *N-1 criterion*, that considers that the system should be able to withstand the loss of any single one of its elements[1]. A discussion about reliability criteria can be found in section 2.5.

2.2.2.2 *Required reliability level*

In theory, reliability can easily be achieved by massive investment and high operation costs in the power system. However, when decisions are taken, there is always a trade-off between reliability and economy. This trade-off is sometimes encompassed by the notion of *social-welfare* that balances adequately the cost of the decisions, the resulting costs on system operation, the ecological impact of the decisions and the expected socio-economic cost of service interruption for end-users [Marceau et al., 1997]. Ideally, decisions should optimize the social-welfare. In practice, attributing a value for all these elements is a difficult task.

2.2.3 Reliability assessment & control

Reliability management is typically divided into assessment and control, where reliability assessment is used to inform reliability control, as is represented in Figure 2.2.

[1]See [ENTSO-E, 2009] for the version of the N-1 criterion at the level of ENTSO-E, the European association for the cooperation of TSOs for electricity.

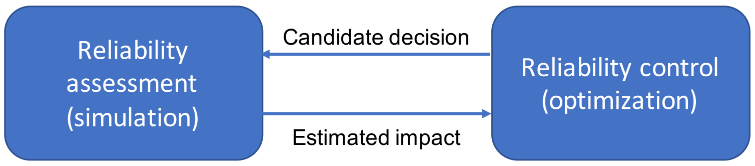

Reliability assessment (simulation)
Candidate decision
Estimated impact
Reliability control (optimization)

Figure 2.2: Interaction between reliability assessment and control.

2.2.3.1 *Reliability assessment*

Reliability assessment is used to evaluate the potential impact of a candidate decision over the time horizon considered. More precisely, it evaluates the induced probability of meeting a reliability target and the expected operating cost over a certain future time period. It can also be used to evaluate other reliability and socio-economic indicators, such as the probability of service interruption (called the loss of load probability), the expected energy not served, the expected cost of energy not served, etc. More information about these reliability indices can be found in [Billinton and Allan, 1996].

Reliability assessment therefore allows to verify, for a given candidate decision, if that decision would allow to operate the system in real-time while meeting the reliability criterion. It can also be used to evaluate the effect of already taken decisions over some past operational period.

For a given candidate decision, it typically consists in *simulating* power system operation and modeling the behavior of the system for plausible future system states obtained from forecast models, in order to compute reliability and socio-economic indicators.

2.2.3.2 *Reliability control*

Reliability control is used to compute decisions such that the system complies with a reliability criterion over the time horizon considered, while optimizing the social-welfare. It typically consists in solving an *optimization problem*, with a socio-economic objective function, in order to find among a set of candidate decisions an optimal decision meeting the reliability criterion. Because reliability control screens a much larger set of candidate decisions, this problem is more complex than the reliability assessment problem and one may have to relax some constraints to obtain a (near-optimal) feasible solution.

Note that there is always a trade-off in reliability control between taking decisions at the last moment to reduce the level of uncertainty and taking them in advance to ensure to have enough time to compute good decisions, given that computing good decisions is a complex task.

2.2.4 Static vs dynamic security

Security can be further divided between static and dynamic security. *Dynamic security* studies the ability of the system to remain in a stable equilibrium when in a normal operating state and to regain a state of stable equilibrium after a disturbance. Rotor-angle, voltage and frequency stabilities are the three main physical phenomena. Their dynamics range over a period of some seconds to several minutes after the occurrence of a disturbance. More information about the physics and mathematical models of dynamic security can be found in [Kundur et al., 1994].

Static security studies the steady-state reached after the occurrence of a sudden loss of an element in the system. The time period typically ranges from 5 to 60 minutes. It studies compliance with the permanent capabilities of the components, for instance by checking if there is no violation of equipment loading limits [Duchesne et al., 2020b].

The sudden loss of one or more elements in the power system, such as the loss of a generator, a transmission line or a transformer, is called a *contingency*. In this dissertation, we focus on static security and therefore consider the impact of contingencies on the power system steady-state. Note however that the approach proposed in this thesis, i.e. the use of proxies, could be extended to address dynamic issues.

2.3 Uncertainties in reliability management

Operating a power system is complex not only because of its large size and the number of stake-holders, but also because of the many uncertainties faced by the operator when a decision must be taken. We describe in this section the most important ones, although in this dissertation we consider only the uncertainties from demand, renewable generation and component sudden outages for our case studies.

2.3.1 Weather

The weather has a significant impact on power systems. Indeed, power systems are exposed to external aggressions, such as rain, wind, ice, storm, lightning, that can damage the system and significantly increase the probability of failures of components. It also influences the equipment tolerances. The overhead lines can for instance carry more current when it is windy, due to the cooling effect of the wind [Fu et al., 2010].

The weather not only impacts directly the components of the system but also the demand and renewable generation, as we will see in the next two sections.

Weather forecasts are therefore necessary for planning and operation of the power systems.

2.3.2 Demand

One of the main difficulties of power systems operation is due to the fact that electricity cannot be stored economically at large scale. The production must therefore constantly match the demand, to avoid instabilities in the system.

It is thus important for operators to correctly anticipate the future demand in the system, in order to be able to satisfy it. For this purpose, the operators rely on models predicting the demand for future horizons. They then take decisions based on the forecast demand.

The demand (or load) is impacted by many variables. Some variables are deterministic such as the season, the day of the week (working day vs weekend), the hour of the day but others are difficult to predict exactly such as the weather conditions. For instance, the use of electrical heating systems or air conditioning systems directly depends on the temperature [Hor et al., 2005].

The difference between the predicted load and the observed one is the *forecast error*. It is unavoidable and on average decreases as the forecast is done closer to real-time, due to a decrease in uncertainty. For example, for a medium-sized utility in the US, the day-ahead relative load forecast error is typically around 3% [Hong and Fan, 2016].

2.3.3 Renewable generation

Uncertainty in renewable generation is mainly due to wind and solar power generations, which are weather-driven, highly intermittent and volatile.

Wind power generated by wind turbines is highly dependent on the wind speed, which is impacted by the temperature and the pressure but also by the height, obstacles and the geography [Lei et al., 2009]. It is by nature stochastic and has a strong impact on the wind power forecast error: due to the cubic relationship between the wind speed and the wind power, small errors in wind speed forecast may lead to large errors in wind power forecast [Yan et al., 2015a].

Regarding photovoltaic panels, the solar power is directly related to solar irradiance. Sun position and cloud cover have a strong impact on the output power but solar irradiance also depends on weather conditions such as temperature, pollution, wind speed and direction, and humidity [Ahmed et al., 2020].

The increasing penetration of renewable energy makes the operation and planning of the system more challenging. Similarly to the load, operators rely on models, which are based notably on the weather forecast, to predict renewable generation and anticipate operation. Their forecast error is larger than the load forecast error (except at night for solar power). For instance, the typical day-ahead wind power forecasting error is around 15% in a medium-sized utility in the US [Hong and Fan, 2016].

2.3.4 Component forced outages

A power system is composed of many elements such as overhead lines, cables, transformers, generators, etc. All these elements can fail unexpectedly, due to adverse weather, equipment failure, poor maintenance, line tripping, overloading, etc. [Kundur et al., 1994].

Depending on the type of component, the location in the network and the fact that there is a single failure or multiple failures, forced outages could or not cause problems. In the best case, the system remains within operational constraints with no service interruption while in the worst case, the forced outage initiates a cascade of failures leading to a blackout. The problem is that paths of the electrical flux are defined by physical laws and are hardly controllable. A loss of a transmission line can thus lead to an overload of other lines, that are then tripped, causing more overloading and the cascade of failures.

2.3.5 Energy markets

The liberalization of the electricity sector more than 20 years ago allowed for competition in electricity generation and retail, but at the price of new uncertainties for the operator. For instance, the electricity prices in power systems have become much more volatile [Dyner and Larsen, 2001] and are only known after the market clearing. The market clearing is an operation carried out by the market operator in day-ahead and intra-day, taking as inputs the producers offers and the consumers bids, in order to determine (based on the merit order list) the electric energy to be provided by each producer, the electric energy to be consumed by each consumer and at which price (called the market clearing price) [Conejo and Baringo, 2018]. Some approximations of grid constraints are also used in the market clearing process to avoid stressing the system.

More information about the market mechanism can be found in [Biggar and Hesamzadeh, 2014].

2.3.6 Policies, technological breakthroughs, change of behaviors

Almost impossible to forecast several years in advance, incentive policies, technological breakthroughs and change of behaviors can significantly modify the shape of the energy landscape. It can either bring new challenges for planning and operating the system or on the contrary facilitate the work of operators. To illustrate the impact of these uncertainties, we give some examples hereafter.

An important incentive policy is the decarbonization policy. The different incentives of the governments accelerated the development of renewable generation, which in consequence, increased rapidly the amount of uncertainties in the power system.

Another example of regulatory policy is the decision in European countries, following a European directive, to roll out smart meters. Smart meters can bring demand flexibility, by allowing the consumers to adapt their consumptions to the price of electricity, in order to better spread the load over the day and thus reduce the peak demand. The speed and extent to which smart meters are installed are greatly influenced by the regulation [Zhou and Brown, 2017].

Concerning technological breakthroughs, we cite as examples cloud computing, that offers access to large computing resources and thus allows running large-scale and computationally intensive simulations for planning studies [Luo et al., 2018]; and electrical vehicles that push forward the electrification of the society, change the demand profile, and could provide demand response [Shao et al., 2012].

Change of behaviors, for instance due to the deindustrialization, to better energy efficiency or to a trend to use electric heating systems, can lead to a change in the demand profile that is difficult to predict [Boßmann and Staffell, 2015]. Note that change of behaviors is strongly related to incentive policies and progress in technology.

2.3.7 Cyber-attacks, software bugs, human errors

As the power system is becoming more and more a cyber-physical system with more integration of information and communication technologies to help monitoring and controlling the system in real-time, the software bugs and cyber-attacks become a problem for the operator, with possibly disastrous consequences. For instance, the main cause of the major blackout of 2003 in North America was a software bug in the alarm system of the control room [Wikipedia, nd], while the blackout of 2015 in Ukraine was due to a cyber-attack [Liang et al., 2016]. Similarly, human errors, such as taking wrong or ill-timed decisions, or cranes hitting power lines, can also lead for instance to unwanted tripping of elements or interruption of normal operation, and thus possibly to a blackout [Haes Alhelou et al., 2019]. All these events are difficult to predict.

2.4 Types of decisions and corresponding time horizons

The decisions in reliability management can be organized in five contexts: regulatory decisions, grid development, asset management, operation planning and real-time operation. Figure 2.3 shows the last four contexts with respect to their corresponding time horizons.

We review shortly these contexts in this section, with a particular focus on operation planning since it is the subject of this dissertation, and real-time operation given that it is intrinsically linked to operation planning.

2.4.1 Regulatory decisions

Regulatory decisions in reliability management are decisions taken at the political level or by regulators that impact the reliability and reliability management of the system. A good example of a regulatory decision impacting reliability management is the decision of the European commission that most TSOs in Europe should comply with the N-1 security criterion [European Commission, 2017, Article 35].

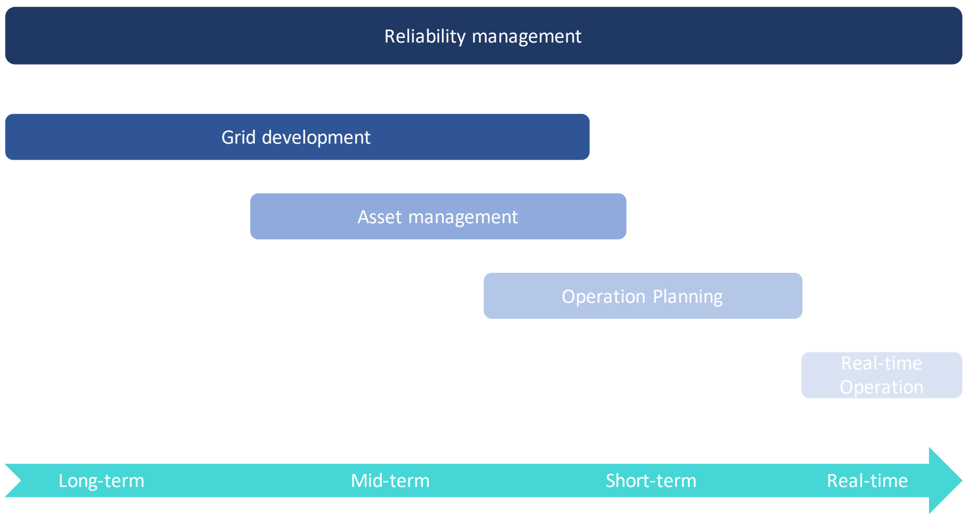


Figure 2.3: Main contexts of reliability management and their corresponding time horizons. Long term corresponds to several decades to several years ahead, mid-term to several years to several months ahead, short-term to a few months to several hours ahead and finally real-time to one hour to a few minutes ahead.

2.4.2 Grid development

The grid development context consists in identifying the future needs of the grid, deciding what elements (and which technology) should be added to the grid, where and when, while taking into account the future level of reliability as well as economic constraints. Depending on the type of component, the time horizon of consideration varies from several decades to several months ahead.

Grid expansion is performed in the long-term (several decades ahead). Indeed, the decision to build new transmission lines must be taken years in advance to take into account the construction time but also and mainly all the regulatory aspects, legal and administrative procedures, authorizations, land acquisitions and possible public comments and opposition that may delay the execution of the project [Battaglini et al., 2012].

Mid-term grid development (several years ahead) concerns investments in the grid infrastructures, such as building new substations, developing the telecommunication system, adding smart meters, etc.

Finally, short-term grid development concerns the construction of new phase-shifters or protection systems and can be planned several months ahead [Khuntia et al., 2016].

2.4.3 Asset management

The asset management context considers both the definition of policies to maintain the grid and the actual scheduling and execution of elements inspection, maintenance and replacement. The time horizon of the asset management decisions goes from several years ahead, in particular to define maintenance policies, to several months ahead to actually plan the maintenance and replacement [Khuntia et al., 2016].

2.4.4 Operation planning

Operation planning prepares the system for real-time operation, and is done at several time intervals, from several weeks to several hours ahead, to ensure that real-time operation will turn out in the best possible way.

2.4.4.1 *Uncertainties*

The market clearing output in day-ahead and intra-day operation planning are known in this context. The uncertainties therefore mainly relate to demand and renewable generation forecast errors, as well as forced outages of components (contingencies).

2.4.4.2 *Examples of decisions*

Several tasks are performed by the operator during operation planning. An important task is the evaluation of outage requests. The operator evaluates if the system can comply with the security criterion, given the planned outages of components for asset management or grid development purposes. If the operator assesses that the system is too vulnerable with the considered outages, it can postpone them.

It is also during operation planning that the operator plans and commits reserves (e.g. back-up production capacity) to cover possible future imbalance between demand and production, and prepares voltage control by setting the position of tap-changing transformers and by committing generators to provide reactive power.

In a post-market clearing context, the operator verifies that the result of the market clearing leads to no violation of the transmission constraints and that the system is able to withstand contingencies as per the defined reliability criterion. If necessary, the operator can acquire redispatch flexibility.

Finally, in the specific case of a centralized framework in which the operator takes also the production decisions, the operator schedules in day-ahead the dispatchable generating units, which are units that can be controlled, contrarily to weather-dependent generating units such as wind farms [Conejo and Baringo, 2018]. Determining optimally the on/off status of generating units during the 24 hours of next day can be done with a Unit Commitment (UC) problem, which is described in section 2.6.3.

A more complete description of European TSO decisions in operation planning can be found in [GARPUR Consortium, 2015].

2.4.4.3 *Assessment and control*

In operation planning, reliability assessment is used to evaluate the potential future system states, by analyzing the power flows and impact of contingencies (*contingency analysis*). For this purpose, operators simulate the operation of the system over future time periods, given an operation planning candidate decision, for instance by using optimization programs such as Optimal Power Flow (OPF) or Security-Constrained Optimal Power Flow (SCOPF), which are described respectively in sections 2.6.1 and 2.6.2. If the result of the assessment is not satisfactory, reliability control computes an operation planning decision that would comply with the security criterion.

2.4.5 Real-time operation

Real-time operation is the last reliability management context and goes from one hour to a few minutes ahead. The objective of the operator is to ensure that the electricity demand is met while satisfying operational constraints (such as components capacity limits) by taking appropriate control actions when the system undergoes unexpected disturbances.

2.4.5.1 *Uncertainties*

In this context, it is considered that the demand and renewable generation are well known. The uncertainties that remain concern possible disturbances such as unexpected outages of components.

2.4.5.2 *Types of operating states of the system*

To describe the security of the system, Dy Liacco [1967] identified five possible states of the power system. These states and their transitions are represented in Figure 2.4, which is adapted from [Fink and Carlsen, 1978]. The system is in *normal state* when it is stable and within its operating limits. Furthermore, the security criterion is satisfied. Then, if a disturbance such as a forced component outage or an imbalance between demand and generation occurs, or if the security criterion is not met anymore, the system enters in *alert state* or *emergency state*. In alert state, the system is still stable and within operating limits but the security criterion is not met anymore. If preventive actions are taken such that no limit violation appears in consequence to the disturbance and the security criterion is met again, the system goes back to the normal state. In *emergency state*, operational limits are not met anymore and the system is at this point insecure. If no further action is taken (or corrective actions fail), part(s) or the totality of the system collapses, resulting in a partial or total service interruption. The system is then said to be in *in extremis state*. Finally, when the system stabilizes, it enters in *restorative state* before going back either to normal or to alert state when actions are taken to restore the lost parts of the system (components, loads, generators).

2.4.5.3 *Preventive and corrective control*

There are therefore two types of security control that can be identified: preventive control and corrective (or emergency) control.

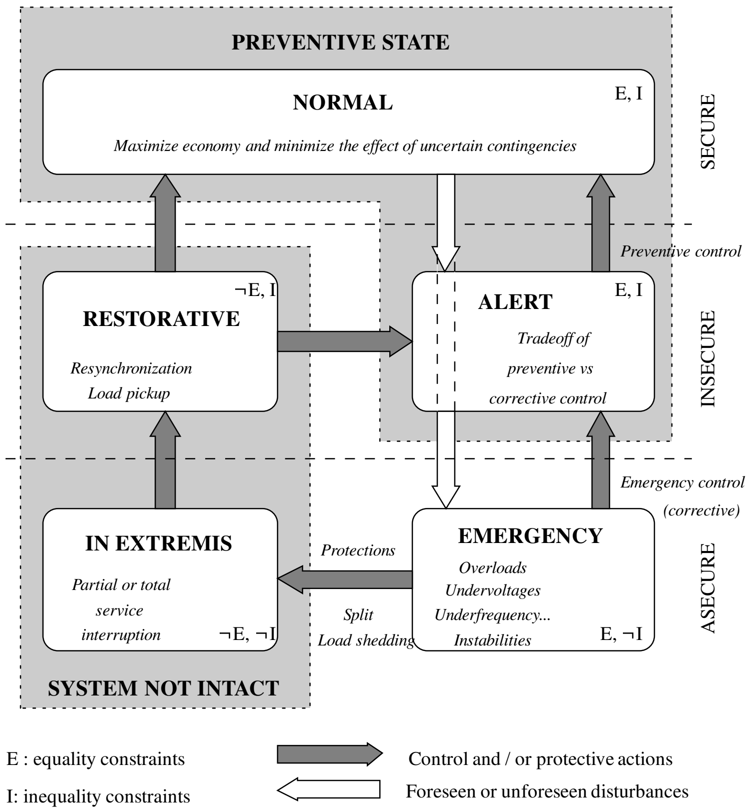

PREVENTIVE STATE
NORMAL
E, I
Maximize economy and minimize the effect of uncertain contingencies
SECURE
Preventive control
RESTORATIVE
¬E, I
Resynchronization
Load pickup
ALERT
E, I
Tradeoff of preventive vs corrective control
INSECURE
Emergency control (corrective)
IN EXTREMIS
Partial or total service interruption
¬E, ¬I
Protections
Split
Load shedding
EMERGENCY
Overloads
Undervoltages
Underfrequency...
Instabilities
E, ¬I
ASECURE
SYSTEM NOT INTACT
E : equality constraints
I: inequality constraints
Control and / or protective actions
Foreseen or unforeseen disturbances

Figure 2.4: State transition diagram for security control (from [Wehenkel, 1997]).

Preventive control corresponds to decisions taken in a normal state, before the occurrence of a disturbance, in order to prepare the system to withstand disturbances. These decisions allow the system to stay within operating limits for a set of selected disturbances. Choosing preventive actions corresponds to a trade-off between security and economy.

On the contrary, *corrective control* corresponds to actions taken in reaction to a disturbance to avoid operational constraint violations and service interruption. Generally, the system goes from alert to emergency state during the time necessary to react to the contingency, and then returns to a normal state thanks to their application. Security of supply in that case becomes the primary concern, over economy.

Preventive and corrective actions can be for instance generation redispatch or curtailment, load shedding, topology change by opening or closing bus splitting breakers, setting tap position of phase-shifting transformers and activating reserves.

2.4.5.4 *Assessment and control*

In real-time, reliability assessment is used continuously to evaluate the security state of the system and verify that it is able to withstand the most credible contingencies while reliability control is used to determine both preventive and corrective actions, for instance with an OPF or a SCOPF, taking into account all automatic controls installed in the system.

2.4.6 Coherence between reliability management contexts

As was mentioned earlier, it is important to maintain a coherence between the different reliability management contexts. Indeed, decisions in longer-term contexts are taken assuming how shorter-term contexts are managed, in particular regarding the considered contingencies and availability of control actions.

Security management can therefore be mathematically modeled as a multi-stage optimization problem, where each stage represents a different context, or a different time horizon within the same context.

2.5 Reliability criteria

A reliability criterion defines a minimum level for the reliability of the system. There exist many reliability criteria and they often depend on the operator. One can distinguish classical criteria and probabilistic ones.

2.5.1 Classical reliability criteria

With classical reliability criteria, a set of credible contingencies is determined and the system must be able to withstand all contingencies in the defined set, without service interruption or violation of operating constraints.

Among the classical criteria, the most famous one is the N-1 criterion, where N represents the number of elements in the system, which states that the system must remain secure even in case of the sudden loss of any single one of its elements (e.g. transmission line, generating unit, transformer). The contingency set therefore contains all single forced outages. This is the security criterion (or a variation of this) followed by most TSOs [GARPUR Consortium, 2014] and the one implemented in this thesis for the case studies.

Other classical criteria include for instance the N-0 security criterion, that only considers operational constraints but no contingency, the (N-1)-1 security criterion which accounts for the fact that we might have another contingency before restoring the required level of reliability after an N-1 contingency, the N-2 security criterion that considers as contingencies the simultaneous loss of one or two components, etc.

Classical criteria are easy to understand and have led until now to highly reliable systems. However, it might not be the case in the future, because of the increasing uncertainties in power systems. Studies have shown that the level of reliability of a system operated following the N-1 criterion may vary, depending on the situation [Kirschen and Jayaweera, 2007]. Indeed, this criterion can be over-conservative at some periods, for instance when the probability of failure of some components is very low, while leading to a system not reliable enough at others, for instance when the weather is such that multiple outages are very likely to occur. Probabilistic reliability criteria can counterbalance this drawback.

2.5.2 Probabilistic reliability criteria

Contrarily to classical criteria, probabilistic reliability criteria take into account the stochastic nature of the power system, by considering the probability of each contingency to occur and ideally their estimated impact (or severity) on the system. The measure encompassing both the probability of an event and its impact is called the *risk* of the event.

An example of a probabilistic reliability criterion is the probabilistic Reliability Management Approach and Criterion (RMAC) proposed in [Karangelos and Wehenkel, 2016]. It is composed of three components: a reliability target, a socio-economic objective and a discarding principle. The *reliability target* defines the minimum level of reliability in terms of a lower bound on the probability of reaching unacceptable system states. An unacceptable system state can be a state where operational constraints are violated or a state with a certain amount of service interruption. The *socio-economic cost* consists of the cost induced by control decisions, that can be preventive and/or corrective, and the induced risk of service interruption (for instance if corrective control fails). Finally the *discarding principle* is used to define the subset of contingencies that can be neglected when computing the control decisions. This subset of contingencies must be such that the aggregated risk of these contingencies is below a given threshold.

Note that it is possible with a probabilistic reliability criterion to revert to classical reliability criteria such as the N-1 criterion by selecting appropriately the parameters and inputs of the approach, for instance by assigning the same probability to occur to all N-1 contingencies and by considering all the other contingencies as the subset of contingencies that can be neglected.

A probabilistic approach presents multiple advantages. It would allow the operators to evaluate the risk of a contingency and determine if it would be profitable to take actions against this contingency. It would also allow for a better choice between preventive and corrective actions and to consider the probability that corrective control fails. It is therefore a solution to operate the system closer to its limits while maintaining an acceptable cost, allowing the system to be both more reliable and more economical.

Research in reliability management tends to develop techniques based on a probabilistic criterion to plan and operate the system (see, for instance the European project GARPUR and references [Heylen et al., 2019; Karangelos et al., 2013; Karangelos and Wehenkel, 2016; Perkin et al., 2017; Strbac et al., 2016]). However, there is still a gap between research and practice, due to barriers for the wide implementation of a probabilistic reliability criterion. The main barriers are data quality issues (for instance to correctly estimate the contingency probabilities or the economical impact of service interruption), the complexity of the decision-making approach and induced computational complexity, and lack of practical experience [Heylen et al., 2019].

2.6 Tools for reliability management

We present in this section the tools used in this dissertation for modeling reliability assessment and control in operation planning.

2.6.1 Optimal power flow

The Optimal Power Flow (OPF) is an optimization problem used to determine optimal control actions to satisfy the demand while considering operational constraints. It was first formulated by Carpentier [1962] and is used both for planning and operation.

2.6.1.1 *Objective function*

In OPF problems, the objective function most often minimizes the cost of operation but other objective functions are possible. For example, one can minimize the transmission losses in the system, minimize the CO_2 emissions, minimize voltage deviations, etc.

2.6.1.2 *Constraints and variables*

The variables in the optimization problem both describe the system state (complex voltage at each bus) and the control actions (generator active and reactive power dispatch or redispatch, load shedding, power line switching, topology reconfiguration, etc.), while the equality constraints relate to the physics of the system, and the inequality constraints to the network operational limits and the physical limits of the control actions. We list here some of these constraints. Depending on the choice of formulation and the control actions available, other constraints can be considered.

- *Active and reactive power balance*: the sum of the power injections (that can be positive or negative) at each bus (also called busbar) must be equal to 0. This results from the Kirchhoff's laws. The power injections at each bus come from the loads and generators connected to the bus, but also from the flows in the transmission lines connected to the bus. These power flows are a function of the complex voltages of the buses.

- *Voltage limits*: the voltage magnitude at each bus and the voltage phase difference between two directly connected buses are bounded to maintain safe operation.

- *Thermal limits on transmission lines*: the flow in each transmission line is limited due to the thermal limit of the conductors.

- *Generator active and reactive power limits*: the generating units have generally a minimum and maximum level of output power. Redispatch control actions are therefore limited by these bounds.

- *Generator ramping limits*: the output power of a generating unit cannot be instantaneously increased or decreased. The operator must take into account the ramping limits of the generators, which depend on the amount of time available to apply these controls, when choosing redispatch control actions.

2.6.1.3 *Some challenges for OPF problems*

The OPF is a non-linear and non-convex optimization problem, with a large number of constraints (both equality and inequality constraints) and variables (that can be both continuous and discrete). It is therefore a hard problem to solve, with no guarantee to find the global optimum.

Therefore, there is a large body of literature studying how to relax the formulation of OPF problems in order to simplify their solving. The most simplified version of the OPF is the copper plate approximation, that neglects the network constraints. It is generally too unrealistic to be used in practice. The second one is the popular Direct Current (DC) approximation [Wood and Wollenberg, 2012], that linearizes the OPF equations by considering that voltage magnitudes are constant and equal to their nominal value and by neglecting reactive power. With the DC approximation, the OPF becomes a linear problem, much easier to solve than the classical Alternating Current (AC)-OPF. It is often used in planning studies for security assessment.

Nowadays, research is still very active to convexify the AC-OPF, by means for instance of Second Order Cone (SOC), Semi-Definite Programming (SDP) or Quadratic Convex (QC) relaxations [Coffrin et al., 2015; Low, 2014; Molzahn and Hiskens, 2019]. Recently, machine learning techniques have also been exploited to help solving OPF problems. The most recent works in this field are presented in chapter 4.

Another challenge in OPF problems is the explicit consideration of uncertainty in their formulations. Methods have been proposed in the literature, for instance via chance-constrained formulations [Roald and Andersson, 2017].

Finally, the last challenge discussed here is the inclusion of security in the OPF formulation. This is called a security-constrained optimal power flow and is described in the next section.

2.6.2 Security-constrained optimal power flow

The Security-Constrained Optimal Power Flow (SCOPF) is an extended version of the OPF, where constraints representing the operation of the system under some postulated contingencies are also considered. It is more complex to solve than the OPF, given the significantly larger number of variables and constraints and the presence of more discrete variables to represent control actions, such as network switching [Capitanescu et al., 2011].

Contrarily to the OPF, the SCOPF allows to consider preventive and corrective control actions in response to the possible occurrence of a contingency.

Since it considers the security of the system regarding credible failures, it allows for a better trade-off between security and economy.

We refer the reader to [Capitanescu, 2016] for a review of the advances and challenges in the field of OPF and SCOPF.

An example of the mathematical formulation of a SCOPF can be found in appendix A. The DC approximation is used and preventive and corrective actions are generation redispatch, load shedding and wind curtailment. The considered contingencies are single outages of transmission lines and transformers.

2.6.3 Unit commitment

A Unit Commitment (UC) problem is an optimization problem used to optimally determine when to use the generating units, while meeting some technical and operational constraints. It is a multi-period Mixed Integer Programming (MIP) problem, where binary variables are used to represent if a given generating unit is on or off at a certain moment of the considered period. It is often combined with an Economic Dispatch (ED) problem, which determines the actual power output of each available generating unit ('on' status) to meet the forecast *net demand* (demand minus renewable generation), while minimizing the production cost. A schematic representation of a solution of a UC and ED problem can be seen in Figure 2.5.

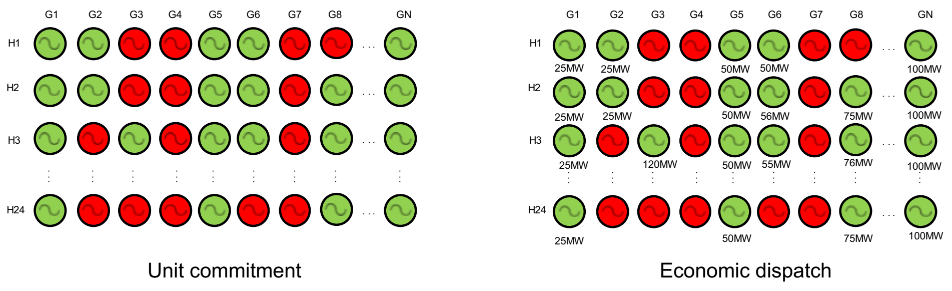


Figure 2.5: Illustrative example of a solution of a day-ahead UC and ED problem. Each row represents an hour of next day and each column a generating unit, which is green when it is on and red otherwise.

The objective function of the UC typically minimizes the production cost, which includes the start-up and shut-down costs of generating units as well as a cost function depending on the level of output power of each generator, but it can also be maximizing the energy production profit of a generation company in a free market context [van Ackooij et al., 2018].

The problem generally includes as constraints, besides demand and reserve requirements, the minimum up and down time of generating units, the time needed to start up a unit, the minimum and maximum output power levels and the ramping constraints. As for the OPF and SCOPF, the formulation and the choice of constraints depend on the application and operational environment.

The UC is quite hard to solve, due to integer variables and the large size of the problem. In the UC used by generation companies to decide their offering strategy in the electricity market, the network constraints are generally not considered in the formulation, resulting in a Mixed Integer Linear Programming (MILP) problem [Conejo and Baringo, 2018].

A UC considering transmission constraints is called a transmission-constrained UC [Dvorkin et al., 2014]. It can be linear, convex or non convex, depending on the relaxations used. Security constraints (for instance with generation failures) can also be considered in the problem, leading to a security constrained UC [Wu et al., 2007]. These models are used by operators for planning, market clearing and reliability assessment.

Uncertainty can also be considered in UC formulations, for instance by the use of chance-constrained formulations [Pozo and Contreras, 2012] or stochastic UC [Håberg, 2019].

An example of the mathematical formulation of a security-constrained UC problem with the DC approximation and considering the N-1 criterion for transmission lines can be found in appendix A. This is the formulation we use to generate candidate operation planning decisions.

3

Background in machine learning

Overview

This chapter introduces the main concepts in machine learning, with a particular focus on supervised machine learning. It is intended for a reader not familiar with these concepts. It starts by introducing the different types of machine learning problems, then it focuses on supervised learning and describes its characteristics, the procedure to apply supervised learning, how to assess the accuracy of a model, the main supervised learning algorithms and finally feature selection and feature engineering. We refer the interested reader to more general textbooks for further information [Goodfellow et al., 2016; Hastie et al., 2009; Murphy, 2021].

3.1 General notions

3.1.1 Definition of machine learning

Machine learning is the field of study of algorithms that allow computers to learn from observations or experiments. Learning in this case is defined by Mitchell [1997] in the following way:

A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P, if its performance at tasks in T, as measured by P, improves with experience E.

A machine learning algorithm is therefore an algorithm that improves automatically with experience, given in the form of data. It can be applied to a broad variety of tasks. Depending on the nature of the tasks and the data, different types of machine learning problems exist. They are described in the next section.

3.1.2 Different types of problems

We introduce in this section the three main types of machine learning problems. Other types of problems, not discussed further in this dissertation, exist such as semi-supervised learning or transfer learning. More information about these types can be found for instance in [Chapelle et al., 2010; Weiss et al., 2016].

3.1.2.1 *Supervised learning*

Supervised learning is the most common type of machine learning problems. Given a dataset of input-output pairs (the experience E), the task T consists in predicting from the inputs x the output(s) y [Murphy, 2021]. In other words, supervised learning aims at automatically building a model $h(x)$ to approximate y.

Example: Image recognition, the ability of a computer to identify objects in images, is one example of a supervised learning problem. From a dataset of input-output pairs, where the inputs x correspond to the pixels of the images and the output y the object represented in each image, supervised learning algorithms allow to learn a function that will predict from the pixels of an image the represented object. If one adds more images in the dataset, the ability of the learning algorithm to identify the represented object of each image without error will improve.

3.1.2.2 *Unsupervised learning*

Unsupervised learning aims at gaining insight into the data, determining patterns, structure and extracting new information. Contrarily to supervised learning, there is no output to guide the learning. From a probabilistic point of view, we have a set of observations of a random vector X (the experience E) and we want to infer some properties of the system who generated the observations of this random vector (the task T) [Hastie et al., 2009].

Example: An example of unsupervised learning is clustering, which consists in splitting data into clusters, each cluster containing "similar" points [Murphy, 2021]. A popular algorithm for this task is the k-means algorithm, an iterative clustering method allowing to gather data into k clusters. Each observation belongs to the cluster with the nearest center, defined as the mean of the data samples in the cluster.

3.1.2.3 *Reinforcement learning*

Reinforcement learning is a type of learning problems that is goal-oriented and learns what actions to take according to some inputs x to maximize a numerical reward signal (the task T). Contrarily to supervised learning, the system learns by interacting with the environment and is not told which action is the best to take. Furthermore, it acts as a closed-loop system and the actions taken at a time step will influence the inputs of the system at later time steps [Sutton and Barto, 2018]. The experience E consists in the sequence of observations the system gathers as actions are taken, corresponding for each time step t of the sequence to the state of the system at time t, the action taken and an instantaneous reward signal.

Example: A common application of reinforcement learning is video games such as Atari video games, as in [Mnih et al., 2013]. In this example, the agent, representing the player in the game, has only access to the video input of the game (the state of the system) and the reward signal. It learns by taking actions and observing the impact on the video inputs and reward signals. The sequence of actions improves as the agent learns how to maximize the reward.

3.2 Characteristics of supervised learning algorithms

The type of problems addressed in this thesis being supervised learning, the remainder of this chapter focuses on it. In this section, we present the types of input and output variables and we describe supervised learning using a standard probabilistic formalization.

3.2.1 Types of inputs

The inputs x are also called *variables*, *attributes* or *features*. There are several types of variables and they can be qualitative or quantitative. A continuous variable is a numerical variable that can take values in a continuous interval, as for example the production of a wind farm in MW. A categorical or discrete variable is a variable that can take a finite number of values. They are often qualitative (for instance a *weather* variable with values 'sunny' or 'cloudy') but they can be quantitative. For instance, a *temperature* variable with possible values 'hot', 'mild' and 'cold' is an ordered categorical variable [Hastie et al., 2009].

3.2.2 Types of outputs

The (vector of) output(s) y is also called the *target*, or *label*. Depending on the nature of the output space, the supervised learning task is called differently. The two standard types are classification and regression.

3.2.2.1 *Classification*

When the nature of the output space is discrete, the task is named *classification*. It is the case for instance if one wants to predict the stability of a power system after the occurrence of a disturbance. The output can then take two values, 'stable' or 'unstable'. The different values of a discrete output y are called *classes*.

3.2.2.2 *Regression*

When the nature of the output space is continuous, the task is named *regression*. A regression task can be for instance predicting the cost of operation given the state of the system.

3.2.3 Probabilistic formalization

The introduction and two first subsections are strongly inspired by sections III.A and III.B in [Duchesne et al., 2020b].

Supervised learning can be formalized in a probabilistic framework. One then considers the inputs $x \in X$ and the outputs $y \in Y$ as realizations of two (vectors of) random variables, drawn from some joint distribution $P_{x,y}$ over $X \times Y$.

The goal of supervised learning is to find a model (or predictor) $h(x)$ among a set of candidate predictors, named the *hypothesis space* $\mathcal{H}$, that will predict y with as little error as possible. In order to measure the performance of a given model h for this task, one needs to define a real-valued *loss function* l. The function l depends on the application and the nature of the output space Y. For instance, a common loss function for classification is the 0-1 loss function, where $l(y, h(x)) = 1(h(x) \neq y)$ and a common loss function for regression is the squared-error loss $l(y, h(x)) = (y - h(x))^2$.

The inaccuracy of a predictor h, called the *expected loss* (or *average loss* or *risk*) $L(h)$, is then measured by:

$$L(h) = E\left\{l(y, h(x))\right\} = \int_{X \times Y} l(y, h(x)) dP_{x,y}. \tag{3.1}$$

Therefore, supervised learning aims to find the function $h \in \mathcal{H}$ that minimizes the expected loss $L(h)$.

3.2.3.1 *Bayes model*

The Bayes model h_B is the best possible model. Assuming the conditional probability distribution $P_{y|x}$ is known, it is defined in a point-wise way by

$$h_B(x) = \underset{y' \in Y}{\arg\min} \int_Y l(y, y') dP_{y|x}. \tag{3.2}$$

It corresponds to the model that minimizes the average loss $L(h)$.

In practice, the conditional probability distribution $P_{y|x}$ is rarely known and one should target a model as close as possible to this ideal model.

3.2.3.2 *Finite dataset and empirical risk minimization*

Instead of knowing the conditional probability distribution $P_{y|x}$, one usually has access to a finite number N of realizations x and y drawn from the joint probability distribution $P_{x,y}$, i.e. a dataset LS[1] of N input-output pairs $\{(x^i, y^i)\}_{i=1}^N$, where $x^i \in X$ is the vector of inputs of *object* i and $y^i \in Y$ the corresponding (vector of) output(s). Assuming that the N realizations are obtained by sampling identically and independently (i.i.d.) N times from $P_{x,y}$, the expected loss $L(h)$ of a predictor h is then approximated by the *empirical risk* or *training error* $\hat{L}(h)$, defined as:

$$\hat{L}(h) = \frac{1}{N} \sum_{i=1}^{N} l(y^i, h(x^i)). \tag{3.3}$$

Once a loss function l is defined, the next step is to choose the space of predictors $\mathcal{H}$. This space should ideally contain the Bayes model h_B, or at

[1] We call the dataset LS for Learning Set, the set used to learn a model $h(x)$.

least models sufficiently close to it in terms of the chosen loss function. This corresponds to selecting a supervised learning algorithm. Indeed, a supervised learning algorithm A is formally stated as a mapping

$$A : (X \times Y)^* \to \mathcal{H}, \tag{3.4}$$

from $(X \times Y)^* = \bigcup_{N=1}^{\infty} (X \times Y)^N$ (the set containing all the possible datasets sampled from $X \times Y$) into a hypothesis space $\mathcal{H}$.

Supervised learning then reduces to solving the following problem, which is called the *empirical risk minimization*:

$$A(\{(x^i, y^i)\}_{i=1}^N) = \arg\min_{h' \in \mathcal{H}} \hat{L}(h') = \arg\min_{h' \in \mathcal{H}} \frac{1}{N} \sum_{i=1}^{N} l(h'(x^i), y^i). \tag{3.5}$$

It is possible to show that if $\mathcal{H}$ is sufficiently small, it produces models whose loss L converges towards $\min_{h \in \mathcal{H}} L(h)$, when the sample size N increases. As far as accuracy is concerned, the supervised learning algorithm should use an as small as possible hypothesis space $\mathcal{H}$ containing good enough approximations of h_B.

Note that in practice, the empirical risk minimization is a very difficult problem to solve. Most practical algorithms are inspired by this principle but there exist various alternative optimization problems that do not necessarily seek to find the global minimum. Moreover, we will see later on that perfectly predicting the dataset LS might be counterproductive.

3.2.3.3 *Generalization and expected generalization errors*

Let us consider a dataset LS of size N, a loss function l and a model h_{LS} obtained by applying algorithm A to LS. For h_{LS} to be a good predictor, it should be able to also predict well data independent from the dataset LS, but drawn from the same distribution. This is measured by the *generalization error* (also called the *testing error*) of the model h_{LS}, which is given by:

$$L(h_{LS}) = E\{l(y, h_{LS}(x))\} = \int_{X \times Y} l(y, h_{LS}(x)) dP_{x,y}. \tag{3.6}$$

It corresponds to the expected loss and is used to guide the choice of a learning method and assess the performance of the chosen model.

The generalization error of the Bayes model $L(h_B)$ is called the *residual error*. Since the Bayes model is the ideal model, it gives a lower bound on the generalization error for all candidate models. It can be different from 0 when the relation between X and Y is not completely deterministic.

The generalization error $L(h_{LS})$ also depends on the random nature of the dataset LS used to train the model. One can therefore define an *expected generalization error* $E_{LS}\{L(h_{LS})\}$[2] over all datasets of size N:

$$E_{LS}\{L(h_{LS})\} = \int_{(X \times Y)^N} L(h_{LS}) dLS. \tag{3.7}$$

[2]To avoid confusion with the notation $E\{\cdot\}$ which refers in this section to the expectation over $X \times Y$, we use here the notation $E_{LS}\{\cdot\}$ to refer to the expectation over all possible LS of size N that can be drawn from the joint probability distribution $P_{x,y}$.

This error represents how good a learning algorithm is on average over all the possible learning sets of size N and not only for one given dataset. It is therefore useful to characterize a learning algorithm.

3.2.3.4 *Bias-variance trade-off*

To better understand the expected generalization error, let us consider a regression task, the squared-error loss $l(y, h(x)) = (y - h(x))^2$ and a learning algorithm A. In that case, the expected generalization error $E_{LS}\{L(h_{LS})\}$ of the learning algorithm can be decomposed into three terms (for the full development, see [Geurts, 2002, chap. 3])[3]:

$$\begin{aligned} E_{LS}\{L(h_{LS})\} = {} & E_x\left\{E_{y|x}\left\{(y - h_B(x))^2\right\}\right\} \\ & + E_x\left\{(h_B(x) - E_{LS}\{h_{LS}(x)\})^2\right\} \\ & + E_x\left\{E_{LS}\left\{(h_{LS}(x) - E_{LS}\{h_{LS}(x)\})^2\right\}\right\}. \end{aligned} \tag{3.8}$$

The first term corresponds to the residual squared error and is therefore the lower error bound achievable for all models, independently of the algorithm chosen or the dataset used to train the model. The two other terms characterize the learning algorithm. The first one is the average squared error between the average predictor and the Bayes model and thus corresponds to the squared *bias* of the algorithm, and the second one measures in average how much the model h_{LS} varies with the learning set and corresponds to the *variance* of the algorithm.

To minimize $E_{LS}\{L(h_{LS})\}$, one must therefore act on the variance and the bias, that are usually both functions of the *complexity* of the model. Most learning algorithms can work with different levels of complexity. We will see some examples in section 3.5. Generally, the more complex the model, the smaller the training error $\hat{L}(h_{LS})$.

On the one hand, the complexity of the model must be increased in order to decrease the bias of the algorithm. On the other hand, the complexity of the model must be decreased in order to reduce the variance. There is therefore a trade-off between bias and variance in supervised learning. It is illustrated in Figure 3.1.

An important aspect to monitor when learning a predictor h is *overfitting*. A model h_{LS} is said to overfit when it predicts well the data in LS (i.e. the empirical risk $\hat{L}(h_{LS})$ is low) but it is not able to predict as well independent data drawn from the same distribution $P_{x,y}$ (i.e. the generalization error $L(h_{LS})$ is high). Usually, such a model has a large variance and a low bias and is too complex to be able to generalize to new, unseen data. A possible solution to try to avoid overfitting is *regularization*. It reduces the size of the hypothesis space $\mathcal{H}$, for instance by adding to the loss function a penalty term taking into

[3]Similarly to footnote 2, the notations $E_x\{\cdot\}$ and $E_{y|x}\{\cdot\}$ refer respectively to the expectation over the marginal probability density P_x and the conditional probability density $P_{y|x}$.

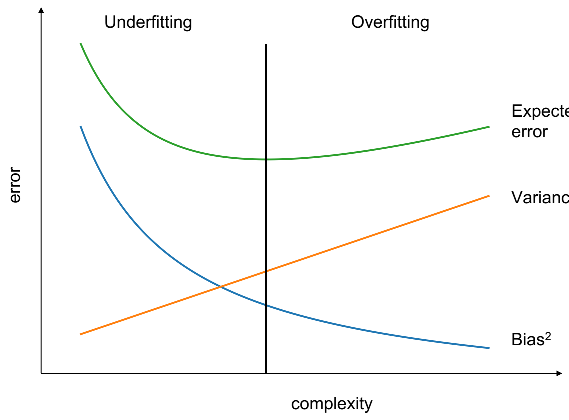


Figure 3.1: Illustration of the bias-variance trade-off.

account the model complexity. It reduces the variance but may increase the bias.

On the other hand, *underfitting* refers to a model too simple to learn the underlying relationship between the input x and the output y and is characterized by high empirical risk and generalization error. The link between the bias-variance trade-off, underfitting and overfitting is represented in Figure 3.1.

3.3 Dataset building and impact on the quality of the learning

In machine learning textbooks, it is often considered that a dataset is already available. In practice, it is not often the case. In power systems in particular, due to lack of historical data availability, simulations are generally used to build a dataset of input-output pairs and learn a supervised learning model. For critical applications such as power systems security, it allows to sample operating points close to the security boundary, which is typically not the case with historical data and real measurements, for which the very large majority of observations are reliable. Simulation methods and points sampled in LS must then be carefully chosen to learn useful predictors for real-life applications.

The quality and representativity of the dataset has indeed a major impact on the effectiveness of the machine learning approach. For instance, for a dataset based on simulation, if the representation of the problem is too simple, this could lead to a learnt model with very good performance on data generated with the same distribution as the dataset LS but bad performance when it is used in practice, in a real situation. Furthermore, the input variables must allow one to discriminate well the target output(s). If it is not the case, the problem may be hard to learn and one may overfit noise, which could lead to difficulties to obtain an efficient model. The dataset must also be large enough for the model to be able to capture all the subtleties of the studied problem, in order to obtain high generalization performances. Another important aspect for the person exploiting the database is to know the hypotheses used to generate it, if this is a database based on simulations, and to know the data collection process, if it is based on real-life data. In general, when the quality and/or representativity of the dataset is insufficient, the learnt models cannot be used in practice [Duchesne et al., 2020b].

Note that once one has built a dataset of input-output pairs (either by simulation or data collection), there are still some steps before learning a model from this dataset, such as data exploration and data pre-processing. Some examples of feature pre-processing are described in section 3.6.

3.4 Practical choice of a model

In this section, we first introduce three criteria that should be considered to select a learning algorithm or model, then we explain how to do model assessment and selection in practice, when only a dataset of input-output pairs is available. Finally, we describe the four loss metrics used in this manuscript.

3.4.1 Criteria to select a learning algorithm and a model

Once a dataset is built and a loss function is chosen, the first step to apply supervised learning is to define the set of candidate predictors $\mathcal{H}$ and a learning algorithm. Accuracy is not the only criterion to be considered when selecting a learning algorithm or a final model. There are commonly three considered criteria: accuracy, interpretability and computational efficiency.

Interpretability of the output of a machine learning algorithm is highly desirable, especially in applications such as medical care where human life is at stake [Sutera, 2019]. Understanding how the algorithm works is particularly important in these situations. Furthermore, interpretability can help the human experts to better understand the problem studied.

Computational efficiency is also often a concern. Solving a supervised learning problem is often computationally expensive, especially since the rise of deep learning[4], and one may be limited by the computational resources and time available. For some applications, there may be a need to keep the models as light as possible, such as for embedded applications.

The different supervised learning algorithms available today yield different compromises between these three criteria. The choice of the most suitable algorithm thus highly depends on the application, for several reasons [Duchesne et al., 2020b]:

(a) the application determines which one of these three criteria is the most important one in practice;

(b) the application determines the data-generating mechanism and loss function, hence the Bayes model h_B and thus how well different hypothesis spaces allow to approximate this ideal target predictor;

[4]Deep learning is introduced in section 3.5.4.2.

(c) the application domain conditions the size of the possibly available datasets, impacting both accuracy and computational efficiency of most algorithms, but in different ways.

This situation means that, typically, one will try out a subset of machine learning algorithms, analyze their behavior and results, and select the one which seems to fit in the best way the needs of the considered application.

3.4.2 Model assessment and selection

In practice, one has a dataset of input-output pairs but no further information about the joint probability distribution $P_{x,y}$. Finding the Bayes model or computing the exact generalization errors of candidate models are therefore rarely achievable. We explain how in practice, from a single dataset LS, one can try to find good candidate models and evaluate their generalization errors.

3.4.2.1 *Training, validation and test sets*

Because the empirical risk minimization chooses models minimizing the average loss over the learning set LS, the empirical risk is typically strongly biased in an optimistic way. Therefore, it is a bad estimate of the generalization error and using it is prone to lead to overfitting. A good practice in machine learning is therefore to use part of the dataset as an *independent test set*, that is not used to train the models but only to estimate the generalization error.

Ideally, one divides randomly the dataset in 3 parts, as is represented in Figure 3.2.

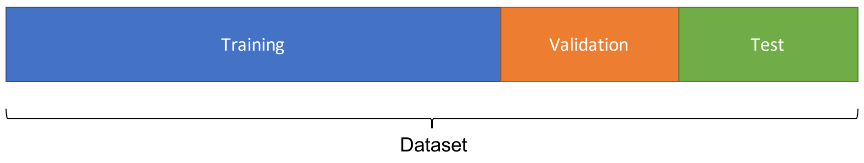


Figure 3.2: Training, validation and test sets.

The larger part is called the *training set* or equivalently the *learning set*. It is used to learn (or train or fit) the models.

The *validation set* is used to evaluate the generalization error of the trained models in order to select among several learning algorithms the one more suited to the studied problem, or to tune some algorithm's *meta-parameters*[5], or even select a good subset of relevant input features. It is particularly useful to avoid overfitting, allowing to tune adequately the complexity of the model to minimize the generalization error.

Finally, the *test set* is generally kept until the end and is used to assess the performance of the finally selected model on independent data.

[5]Many learning algorithms depend on meta-parameters that influence the computational complexity, smoothness, and most notably the accuracy of the learned models. Some examples are presented in section 3.5.

There is no general rule to decide how to split the dataset in three parts and which proportion of the observations allocate to each set. It depends on many parameters, such as the size of the dataset, the complexity of the problem studied, the noise, etc. Ideally, the training set must be as large as possible to obtain good predictors while keeping enough samples in the validation and test sets to correctly estimate the generalization error.

3.4.2.2 *Cross-validation*

When the dataset is small, dividing it in three parts can lead to a learning set too small to learn good predictors (regarding both the empirical risk and the generalization error). In order to avoid this, but still evaluate correctly the generalization error of the algorithm, one can use *k-fold cross-validation.*

First, the dataset is divided into k folds and each model is trained k times, each time with a different fold left out. The left out fold is used to evaluate the test error. The generalization error is then estimated by averaging the k errors computed with the k left out folds. Figure 3.3 represents a 5-fold cross-validation to illustrate this principle. At the extreme, k is equal to the size of the learning set N and all observations in the training set are used to train, except one. It is called the *leave-one-out* cross-validation. Cross-validation therefore allows to better exploit the data, at the price of higher computational requirements [Hastie et al., 2009].

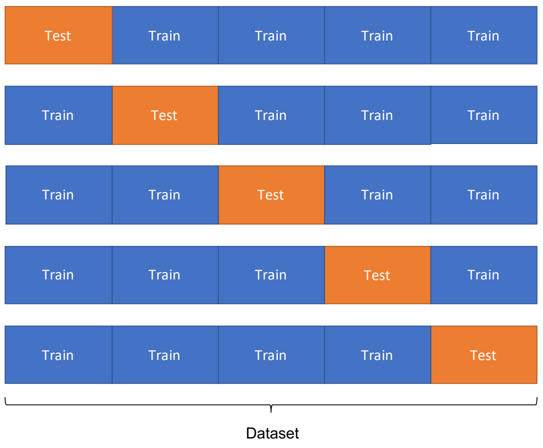


Figure 3.3: Example of a 5-fold cross-validation.

3.4.3 Loss metrics

We present in this section the four loss metrics, two for regression and two for classification, used along this manuscript to learn the models and/or measure the quality of the predictions. Other examples of loss functions can be found for instance in [Scikit-learn developers, nd].

We consider in this section that y is a single-valued output. To compute the loss in case of a multi-output problem where y is a vector, several methods exist, such as averaging the loss of each element in y or considering only the loss of one element.

3.4.3.1 *Mean Squared Error*

The Mean Squared Error (MSE) is a very common error used in regression. It is defined as:

$$MSE(y, \hat{y}) = \frac{1}{N} \sum_{i=1}^{N} (y_i - \hat{y}_i)^2, \tag{3.9}$$

where y_i is the true output value of object i and $\hat{y}_i$ is the one predicted by the model. This error is scale-dependent; the root MSE $\sqrt{MSE(y, \hat{y})}$ is also often used because it can facilitate the analysis of the error, having the same unit as y.

3.4.3.2 *R^2-score*

The R^2-score (or coefficient of determination) is a regression score. It is computed on the basis of N cases by [Scikit-learn developers, nd]:

$$R^2(y, \hat{y}) = 1 - \frac{\sum_{i=1}^{N} (y_i - \hat{y}_i)^2}{\sum_{i=1}^{N} (y_i - \bar{y})^2}, \tag{3.10}$$

where y_i is the true output of case i, $\hat{y}_i$ is the predicted output, and $\bar{y}$ is the mean of the N true values.

It can be (loosely) interpreted as the fraction of the variance of the target output variable that is explained by the model. The second term in the definition can be seen as a relative mean square error and should be as close as possible to 0.

The best possible score is therefore 1 and corresponds to a model that perfectly predicts all the target output values of the dataset used to estimate its value. The best constant model would systematically predict the output mean $\bar{y}$ and obtain an R^2-score of 0. It is possible to obtain negative scores, when the model is arbitrarily worse than the best constant model.

Contrarily to the MSE, the R^2-score is a normalized measure of accuracy, which interpretation is independent on the scaling of the target output variable.

3.4.3.3 *Accuracy*

The accuracy score is a classification measure computing the proportion of correctly classified observations in a dataset:

$$accuracy(y, \hat{y}) = \frac{1}{N} \sum_{i=1}^{N} 1(y_i = \hat{y}_i), \tag{3.11}$$

where y_i is the true class of object i and $\hat{y}_i$ is the one predicted by the model.

This score gives a general idea of how well a predictor performs, but it is often preferable to also study what is called a *confusion matrix*. The confusion matrix is a square matrix of size $C \times C$, with C the number of different classes in y, where element i, j is the number of samples of true class i that have been predicted as class j (or vice-versa). The diagonal elements correspond to the correctly classified samples. The confusion matrix is particularly useful when there is class imbalance, i.e. the proportion of samples of each class is not balanced. The confusion matrix, contrarily to the accuracy score, allows to easily analyze if the predictor is also able to predict well the samples of the minority classes.

3.4.3.4 *Cross-entropy loss*

The cross-entropy loss is a classification loss. A predictor h learnt on the basis of this loss function outputs a vector of scores of length C, the number of different classes in y. This loss maximizes the score s_y of the correct class and minimizes the score of the other classes and is defined for a sample as [PyTorch developers, nd]:

$$loss(s, y) = -\log\left(\frac{\exp(s_y)}{\sum_{i=0}^{C-1}\exp(s_i)}\right), \tag{3.12}$$

where $s = [s_0, s_1, \dots s_{C-1}]$ is the vector of outputs of the classifier and y is the true class of the sample. The class predicted by the classifier for a sample is then the one corresponding to the largest score.

3.5 Main supervised learning algorithms

This section is an adapted version of section 2.2 in [Duchesne, 2016], except for subsection 3.5.4 about neural networks, which is new.

The main supervised learning algorithms used in this manuscript are briefly introduced in the following subsections. In most of this work, the target outputs have numerical values and thus only regression models are used. We therefore limit the discussion to the regression counterpart of each learning algorithm, except for the neural network algorithm which is used for both classification and regression in this manuscript.

3.5.1 Linear regression

One of the simplest supervised learning model for regression is the linear regression. An example is shown in Figure 3.4. The model h has the form

$$h(x) = w^T x + b, \tag{3.13}$$

with $b \in \mathbb{R}$ and is linear in its parameters. Depending on the way the problem is formulated to obtain the coefficients w and the bias b, we have different algorithms. We present here the Ridge Regression (RR) and Support Vector Regression (SVR) algorithms.

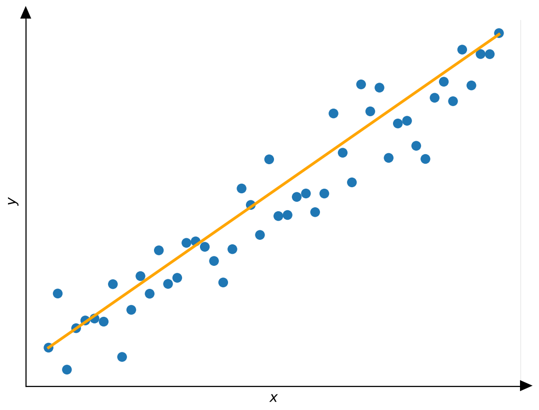


Figure 3.4: An example of linear regression. The points represent the data to fit and the line is the learnt linear model $h(x)$.

3.5.1.1 *Ridge regression*

RR is a least-square regression with an L2-norm normalization as regularization. The vector of regression coefficients w is shrunk towards 0 to reduce the variance and avoid overfitting.

The coefficients w can be obtained by solving the following problem:

$$w^*, b^* = \underset{w,b}{\arg\min} \left\{ \sum_{i=1}^{N} \left(y_i - w^T x_i - b \right)^2 + \lambda ||w||_2^2 \right\}, \tag{3.14}$$

where λ is the regularization meta-parameter. The greater the value of λ, the greater the shrinkage. When $\lambda = 0$, the ridge regression is equivalent to a classical least-square regression. Compared to a least-square regression, the ridge regression has a larger bias but a smaller variance that could result in a smaller prediction error [Hastie et al., 2009].

Note that RR is not scale-invariant. If the ranges of the features are too different, it is generally preferable to normalize them before applying the algorithm.

3.5.1.2 *Support vector regression*

Support vector machines are an important family of methods in machine learning. They usually perform well and their computational complexity is independent from the number of features.

This section focuses on SVR. The idea behind SVR is to learn a linear function in the feature space X that deviates from the learning outputs by at most ϵ_{SVR} and that is as flat as possible.

The problem is written as an optimization problem that minimizes the norm of the coefficient vector w under the constraints that all the samples should lie within a tube of radius ϵ_{SVR} around the predicted function. However, it may happen that it is impossible to find such a function. In that case, it is possible to introduce two slack variables ξ_i and ξ_i^* in the optimization problem so as to allow for deviations larger than ϵ_{SVR}. These larger deviations are penalized via the ϵ-insensitive loss function proposed by Vapnik and that is equal to 0 if $|y_i - h(x_i)| < \epsilon_{SVR}$, $i = 1, ..., N$, and is equal to $|y_i - h(x_i)| - \epsilon_{SVR}$ otherwise. An example of SVR is shown in Figure 3.5.

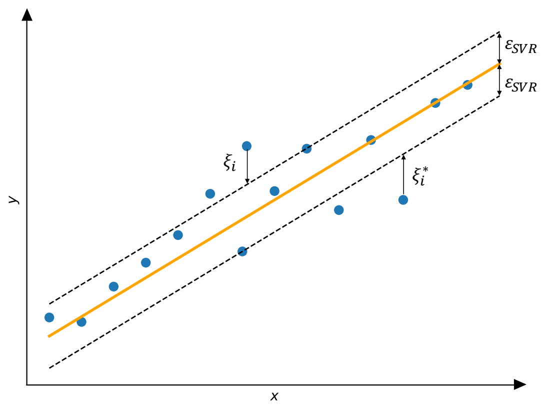


Figure 3.5: An example of SVR. The points represent the data to fit and the orange line is the learnt linear model $h(x) = w^T x + b$.

Formally, the SVR problem is a convex optimization problem that can be written such as [Smola and Schölkopf, 2004]:

$$\begin{array}{ll} \min\limits_{b,w,\xi,\xi^*} & \frac{1}{2}\|w\|_2^2 + C\sum_{i=1}^{N}(\xi_i + \xi_i^*) \\ \text{s.t.} & y_i - (w^T x_i + b) \leq \epsilon_{SVR} + \xi_i \quad \forall i = 1, 2, ..., N \\ & -\Big(y_i - (w^T x_i + b)\Big) \leq \epsilon_{SVR} + \xi_i^* \quad \forall i = 1, 2, ..., N \\ & \xi_i, \xi_i^* \geq 0 \end{array} \tag{3.15}$$

The constant $C > 0$ and ϵ_{SVR} are the two meta-parameters of the algorithm. C defines a compromise between the model's smoothness and the degree of tolerance for deviations greater than ϵ_{SVR}. The larger the C, the less deviations are tolerated but at a price of an increase in the model complexity.

One can show that the solution of the dual form of this problem is [Hastie et al., 2009]:

$$h(x) = \sum_{i=1}^{N}(\alpha_i^* - \alpha_i){x_i}^T x + b. \tag{3.16}$$

The term $(\alpha_i^* - \alpha_i)$ is equal to 0 for all points lying in the tube defined by ϵ_{SVR}. Similarly to RR, SVR is not scale-invariant nor shift-invariant.

3.5.2 Kernel methods

If a linear function is not suitable in the input space, it is possible to learn a linear function in a space induced by a positive kernel. Let's call it the kernel space. A kernel K is a symmetric function such that

$$K(\cdot, \cdot) : X \times X \to \mathbb{R}, \tag{3.17}$$

where X is the input space. The function K is a positive kernel if for any finite choice of inputs $x_1, \dots, x_m$ the $m \times m$ Gram matrix $[K(x_i, x_j)]$ is positive semi-definite.

Mercer's theorem states that if the kernel is positive, it is possible to express it as $K(x, x') = \phi(x)^T \phi(x')$, where $\phi(\cdot)$ is a (typically non-linear) mapping from the input space to the kernel space.

With this relation, instead of computing $\phi(x_i)$ and $\phi(x)$ and then replacing $x_i^T x$ by $\phi(x_i)^T \phi(x)$ in predictor equations such as eq. (3.16), it is possible to replace the dot product in the kernel space by the function $K(x_i, x)$. This is computationally more efficient [Murphy, 2021]. This is called the kernel trick and it can be used for many linear algorithms.

The two kernels used in this work are the polynomial kernel and the Gaussian Radial Basis Function (RBF) kernel. They are defined as:

- polynomial kernel: $K(x, x') = (\gamma x^T x' + 1)^d$,
- Gaussian RBF: $K(x, x') = e^{-\gamma \|x - x'\|_2^2}$.

γ is a kernel meta-parameter for both the polynomial and the RBF kernel and influences the range of actions of the kernel. The polynomial kernel has an extra parameter: the degree d of the polynomial.

3.5.2.1 *Kernel support vector regression*

Using the kernel trick with eq. (3.16), the SVR predictor becomes:

$$h(x) = \sum_{i=1}^{N} (\alpha_i^* - \alpha_i) K(x_i, x) + b. \tag{3.18}$$

This function is linear in the space of the kernel but not linear regarding the input space X, allowing to fit data in more complex space than linear space.

3.5.2.2 *Kernel ridge regression*

There is no clear dot product $x_i^T x_i$ with RR but it is possible to make it appear. The full development can be found for instance in [Murphy, 2021]. The predictor $h(x)$ is then expressed by:

$$h(x) = (w^*)^T x + b = \sum_{i=1}^{N} \alpha_i x_i^T x + b = \sum_{i=1}^{N} \alpha_i K(x_i, x) + b. \tag{3.19}$$

3.5.3 Tree-based methods

Tree-based methods are non-linear methods that divide the input space into regions and fit a simple function in each of these regions [Hastie et al., 2009]. We describe here the regression tree algorithm and two learning algorithms based on this regression tree: random forests and extremely randomized trees.

3.5.3.1 *Regression trees*

The decision tree algorithm iteratively divides the learning set into regions according to splitting criteria. Starting with the full learning set, a splitting criterion is defined by a splitting variable v_j and a split point s and divides the data into two regions R_1 and R_2 which are thus functions of (j, s).

For regression trees, the prediction for both parts of the dataset is their mean output value, that is

$$h(x) = c_1 = \frac{1}{|R_1|} \sum_{x \in R_1} y_i \text{ if } x \in R_1, \tag{3.20}$$

$$h(x) = c_2 = \frac{1}{|R_2|} \sum_{x \in R_2} y_i \text{ if } x \in R_2, \tag{3.21}$$

where $|R_1|$ and $|R_2|$ represent respectively the number of samples in R_1 and R_2. The splitting criterion is chosen in such a way that it minimizes the mean square error:

$$\min_{j,s} \left\{ \min_{c_1} \sum_{x \in R_1(j,s)} (y_i - c_1)^2 + \min_{c_2} \sum_{x \in R_2(j,s)} (y_i - c_2)^2 \right\}. \tag{3.22}$$

Once the dataset is divided into two distinct parts, all the search for a splitting variable and a cut-point is done independently for both parts. The same method is used until the tree is completely built, i.e. all the leaves of the tree contain samples of the learning set with identical output values. An example of a decision tree can be found in Figure 3.6.

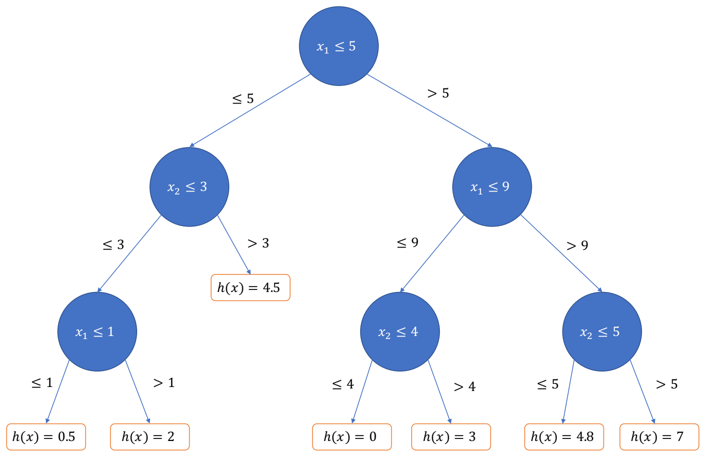


Figure 3.6: An example of a regression tree with two input variables x_1 and x_2.

In practice, fully developed trees are prone to overfitting. Therefore, it is common to stop the splitting process before the leaves are pure (pre-pruning), or to remove some branches after the tree has been built (post pruning). Pre-pruning can be done for instance by specifying a minimum number of samples n_{min} needed to split a node or by defining a maximum depth for the tree. A tree can be post-pruned by removing some subtrees until finding the minimum average prediction error over an independent validation set.

3.5.3.2 *Random forests*

In order to reduce the variance and/or the bias of a machine learning algorithm, it is possible to combine the predictions of several models. The methods performing this are called *ensemble methods*. There are two main families of ensemble methods: the averaging methods where the predictions of each model are averaged and the boosting methods that combine several weak learners to form a strong one.

The Random Forests (RFs) learning algorithms are ensemble methods aggregating decision tree predictors. For regression models, the predictions obtained by each regression tree are averaged to give the final result. For this method to be of value, some randomness must be introduced when building the decision trees in order to aggregate possibly different models. An illustration of a random forest of regression trees can be found in Figure 3.7.

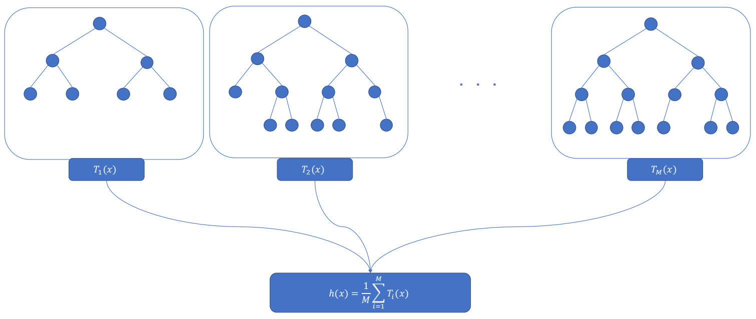


Figure 3.7: An illustration of a random forest of M regression trees. The predictions $T_i(x)$ obtained by each regression tree are averaged to give the final result. Image inspired from [Sutera, 2019].

Different types of random forests are reported in the literature. As an example, bagging (**b**ootstrap **agg**regat**ing**) was introduced by Breiman [1996] and consists in using bootstrap replicates of the learning set to build each tree and then aggregating the predictions. A bootstrap replicate is a subset of the learning set where all the samples have been drawn randomly with replacement, meaning that the same object can be found several times in a particular subset. Another example of random forest was introduced by Dietterich [2000]. In that case, the splitting criterion is chosen uniformly randomly amongst the K best splits.

In this work, we use the random forest algorithm developed by Breiman in [Breiman, 2001] for his experiments. It consists in using bagging associated to a random selection of features. When splitting a particular node, instead of considering all the features and then selecting the best one to split the data, a number K of features are randomly selected and the best split is chosen among those K attributes. One can modify this parameter K to introduce more or less randomness. When $K = 1$, the splitting variable is chosen completely at random whereas if K is equal to the number of features p, only bagging brings randomness. Note that there is no post-pruning of the grown trees in [Breiman, 2001].

This algorithm has thus three meta-parameters, namely $M \geq 1$ (the number of trees in the ensemble), $K \in \{1, 2, \ldots, p\}$ (the smaller the K, the stronger the randomization), and $n_{min} \geq 2$ (the larger the n_{min}, the shorter the individual trees). In terms of accuracy, the larger the M the better and so the value of M depends on the computational resources available, while the optimal values of K and n_{min} are problem dependent. In practice, all three meta-parameters need to be tuned to the problem at hand.

3.5.3.3 *Extremely randomized trees*

The Extremely Randomized Trees (ET) algorithm [Geurts et al., 2006] is also an ensemble method based on trees but it goes one step further in the randomness. As for the random forest, it selects randomly K features but the cut-point for each of them is also chosen at random. Then the best split is selected.

Contrarily to other random forest algorithms, the whole learning set is used to learn the trees. This intends to minimize the bias.

This algorithm thus has exactly the same meta-parameters M, k, and n_{min} as random forests. However, for a given problem, their suitable values may differ from those of the latter algorithm.

3.5.4 **Neural networks**

Neural Networks (NNs) are models mapping input to output through successive layers, each layer being a mathematical function of the previous layers, allowing to gradually increase the complexity of the model. The first NN algorithm, called the perceptron, was proposed by Rosenblatt already in 1958 [Rosenblatt, 1958] but it is only recently (since the 2010s) that, thanks to the progress in computational infrastructure, the increasing amount of data available and the improvement in training algorithms, they became the dominant algorithm in machine learning in general.

3.5.4.1 *Feed-forward neural network*

We describe here the classical fully connected feed-forward neural network algorithm (also called the multi-layer perceptron) since it is the neural network used in this dissertation. It is the simplest neural network algorithm and is composed of one or more hidden layers, each composed of one or several elements called neurons. The information goes from the input layer to the output layer without cycles. An illustration can be seen in Figure 3.8. In the rest of this dissertation, NN will refer to this particular type of network.

Considering a NN with k layers, such as represented in Figure 3.9, the output of layer i is recursively given by

$$z_i = g_{i-1}(W_{i-1} z_{i-1} + b_{i-1}), \; i = 1, \ldots, k, \tag{3.23}$$

where W_{i-1} and b_{i-1} are the parameters of the i^{th} layer of the neural network and g_{i-1} is the i^{th} activation function. We have $z_0 = x$ and the predictor

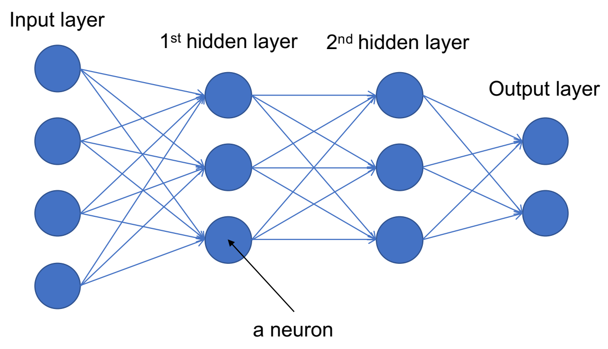


Figure 3.8: A feed-forward neural network with two hidden layers.

h is given by $h(x) = z_k$. For a regression task, $h(x)$ predicts directly the approximation $\hat{y}$, as for the other methods described in this chapter. However, for a classification task, the output $h(x)$ of the NN can take different forms, depending on the loss function chosen and the output layer activation function. For instance, with the cross-entropy loss and $g_{k-1} = W_{k-1}z_{k-1} + b_{k-1}$, $h(x)$ is a score vector of length C (the number of output classes) where each element corresponds to a class. The predicted class is then the one corresponding to the element with the largest score.

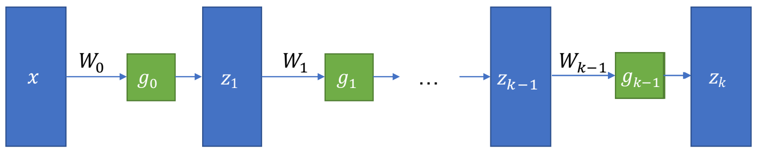


Figure 3.9: Representation of a k-layer NN, with activation functions g_i.

The activation function of the hidden layers used in this work is the commonly used Rectified Linear Unit (ReLU) activation function, which is defined as $ReLU(z) = \max(0, z)$. It is represented in Figure 3.10. This activation function offers good performance on many problems. Despite the non-differentiability at 0, it is easy to optimize and can bring sparsity in the model [Glorot et al., 2011].

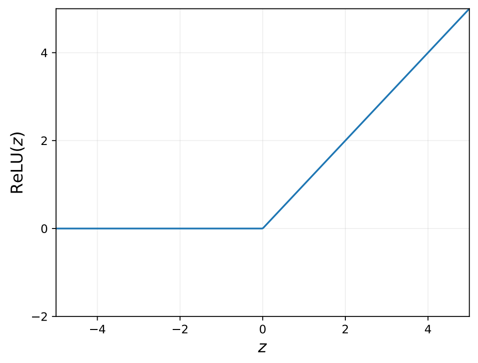


Figure 3.10: The ReLU function.

To optimize the parameters W_i and b_i of the neural network, one resorts to *backpropagation*. Backpropagation computes efficiently the gradient of the loss with respect to each parameter of the network using the chain rule, by computing the gradient one layer at a time, starting from the last layer and iterating towards the first one. The gradient is then passed to a gradient-based optimization method. Common methods to update the parameters of a neural network are based on stochastic gradient descent algorithms such as for instance the Adam optimizer [Kingma and Ba, 2014]. Usually, the learning set is divided into several subsets (called *batches*) and the parameters are updated after each batch, to speed up model training. A cycle through the full learning set is called an *epoch*.

NN, contrarily to tree-based methods, are not scale-invariant nor shift-invariant and it is preferable to standardize the data before learning the parameters of the model, especially when the ranges of value of the features are different, to avoid favoring some features over others.

Given the complexity of these models, regularization methods have been proposed in the literature such as the dropout method [Srivastava et al., 2014], which consists in dropping randomly some neurons during training, in order to reduce overfitting.

The meta-parameters of this method are the number of hidden layers and the number of neurons per hidden layer, which directly relate to the complexity of the method. Some meta-parameters depend on the optimizer chosen to find the optimal parameters of the network, such as the learning rate λ or the size of the batches; others depend on the regularization method (the probability for a neuron to be dropped out during training for instance). The number of epochs is often chosen as the one minimizing the validation set error.

3.5.4.2 *Deep learning*

Deep learning "allows the computer to build complex concepts out of simpler concepts" [Goodfellow et al., 2016]. The feed-forward neural network algorithm illustrates perfectly deep learning since it maps the inputs of a dataset to some output through layers. Each layer can be seen as another representation of the input data. The qualification deep depends on the number of layers of a learning algorithm but there is no clear consensus about the depth of an algorithm necessary to be qualified as deep learning.

The key with deep learning is that the learning algorithm is able to learn the underlying structure and hierarchy of the data and is therefore able to build the right representation of input data for the application studied. One can thus present very large amount of raw data to the algorithm, without needing to process it or add expert-knowledge to help the algorithm [LeCun et al., 2015]. Furthermore, some neural networks structures are particularly suited to some kind of input data. For instance, convolutional neural networks for images or recurrent neural networks for time series data allow to better understand the structure in the data and give impressive performances on complex problems that are hard to solve with other learning methods.

The various new ideas of these recent years such as generative adversarial networks [Goodfellow et al., 2014], graph neural networks [Wu et al., 2020], normalizing flows [Kobyzev et al., 2020], etc. have broadened the range of applications of these methods and offer new perspectives for current complex challenges in real-world applications.

3.6 Feature selection and feature engineering

It is possible to improve the accuracy and/or reduce the computational complexity of a model by modifying the inputs of the problem. Processing the data can improve the quality of the predictions, increase training speed and transform data in more meaningful representation to facilitate model training.

3.6.1 Feature engineering

Feature engineering (or feature pre-processing) consists in computing useful functions of the input variables based on the information provided in the dataset. It transforms the data to represent it in a more meaningful way, to facilitate the learning. Sometimes, it can be used for reducing the dimensionality of the data, for instance with methods such as Principal Component Analysis (PCA). More information about dimensionality reduction can be found in [Murphy, 2021].

3.6.2 Feature selection

Considering a given dataset of input-output pairs, it is possible that some of the inputs are actually irrelevant for the supervised learning task. Identifying these irrelevant variables in order to remove them can both increase the accuracy of the model (for instance by reducing overfitting) and decrease its complexity, as well as give a better understanding of the problem studied.

Feature selection is particularly relevant in power systems applications because many features are generally necessary to fully describe the system operating state, but most often only a subset of them are relevant for each sub-problem addressed with machine learning.

There exist several methods for selecting ‘optimal’ subsets of features. In this work, we exploit the so-called ‘feature importances’ that can be computed as a by-product of training models in the form of random forest or extremely randomized trees. They provide a non-parametric multi-variate ranking of the features and assess the amount of information provided by each feature to predict the output.

The measure we use to evaluate feature importances with tree-ensemble methods is the Mean Decrease Impurity (MDI), which is based on the reduction of impurity performed at each split. The decrease of an impurity measure is what is minimized at each split and corresponds to the variance for regression trees.

For a tree-ensemble method, the importance of a variable v_j for the single tree T, $FI_T(v_j)$, is computed for all trees and the results are then averaged to give :

$$FI_{forest}(v_j) = \frac{1}{M}\sum_T FI_T(v_j) = \frac{1}{M}\sum_T \sum_{t\in T:v(s_t)=v_j} p(t)\Delta i(s_t,t), \tag{3.24}$$

where M is the number of trees in the forest, $\Delta i(s_t,t)$ is the decrease of impurity at node t and for split s_t, $p(t)$ is the proportion of training samples reaching node t and $v(s_t)$ is the splitting variable at node t.

We refer the interested reader to reference [Louppe et al., 2013] for a study of the theoretical properties of these MDI measures and to [Guyon and Elisseeff, 2003] for other feature selection methods.

4

Recent works in machine learning for static reliability management

Overview

This chapter reviews recent works using machine learning for electric power systems reliability management. It treats mostly static security, since it is the focus of this dissertation. We start by providing some statistics about the publications of machine learning applied to reliability management between 2000 and 2019, then we describe a generic dataset used in static security to train the models. The main part of this review presents works published between January 2015 and March 2021 in static security. This part is organized considering the power systems tools to be enhanced or replaced by machine learning techniques, namely power flow analyses, OPF problems, and UC optimization. Finally, we discuss open problems and directions for future work for machine learning in power systems reliability management.

References: This chapter expands on sections IV, VI and VIII of the following publication:

Duchesne, L., Karangelos, E., and Wehenkel, L. (2020b). Recent developments in machine learning for energy systems reliability management. *Proceedings of the IEEE*, 108(9):1656–1676.

Compared to the selected sections of this publication, the text of the survey has been adapted for coherence with the rest of this dissertation and works published since October 2019 have been added for completeness.

The idea to exploit Machine Learning (ML) for reliability management is not new and dates back to the 1980s [Wehenkel, 2012]. But it is only recently that the number of publications started to increase significantly, as can be seen in Figure 4.1[1]. Most of these publications are applied to dynamic security, in particular for security assessment (see Figure 4.2). It is indeed a particularly suitable application for machine learning techniques, given the need for fast assessment and control and the computational burden inherent to classical

[1]We found (via Google Scholar and Scopus) 366 papers in the field of bulk power system reliability management that were published between January 2000 and October 2019 and we analyzed them to obtain Figures 4.1 and 4.2. More information can be found in [Duchesne et al., 2020b].

dynamic security studies such as time-domain simulations, which is typically much larger than for static security assessment.

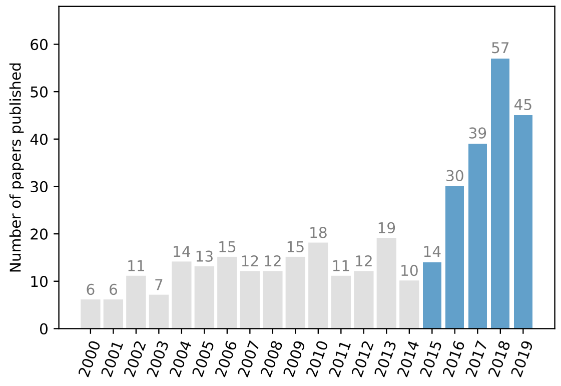


Figure 4.1: Yearly number of published papers we found on the topic of ML application to electric power systems reliability (both static and dynamic), between January 2000 and October 2019.

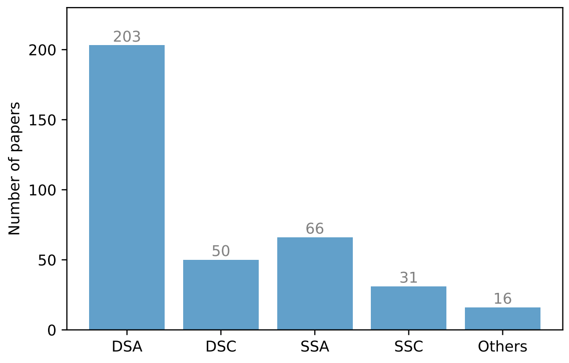


Figure 4.2: Number of ML papers (01/2000 - 10/2019) per reliability management problem (DSA = dynamic security assessment, DSC = dynamic security control, SSA = static security assessment, SSC = static security control).

Recently, machine learning approaches have also been applied to static security. In this chapter, we present recent works in this domain for transmission systems, published between January 2015 and March 2021. Note that some works that seemed less relevant for our discussion were not considered.

We start by describing briefly the datasets used in static security to train the models. The rest of this chapter is then organized considering the power systems tool that is considered to be replaced or enhanced with ML techniques[2]. In particular, we consider power flow computation, optimal power flow solving, and unit commitment optimization. This chapter ends with a discussion about open problems for using ML in the context of reliability management and directions for future works.

[2]Note that the machine learnt models do not necessarily need to be learnt from datasets built based on these power systems tools. They could also be learnt from observational data if these data are available.

4.1 Security database building

The first step to exploit ML is to build a dataset. For security assessment and control, due to lack of historical data availability, simulations are used in most papers to generate a security database. The database generation is usually done offline, given the extensive simulation cost to build it, while the application of the resulting model trained on the dataset can be done offline or online, depending on the application and the context.

In order to generate a dataset based on simulations, the first step is to generate representative operating states of the system. Given that it is impossible to sample all possible operating conditions, several techniques are used to sample a set of representative operating states. It is in most papers based on Monte-Carlo simulations but some papers use observational data from real systems. The input features describe the state of the system and depend on the context and available information within this context. They often consist of demand (active and reactive load value at each bus), renewable generation, change in system topology (e.g. if there is a line in outage) and control decisions (e.g. generator power outputs and voltage magnitude at the buses with generating units). For real-time applications, information available from Phasor Measurement Units (PMUs)[3] such as voltage magnitude and phase of the buses and current flowing through the lines can also be considered.

The outputs of the dataset are usually obtained by simulating the operation of the system and its physical behavior, for instance with power flow or optimal power flow models. For assessment purposes, the outputs can be reliability indicators (e.g. if the system can withstand all N-1 contingencies) or socio-economic costs of operation, while for control purposes, some models predict directly which decisions, such as generation redispatch, to apply. Figure 4.3 summarizes the dataset building process in the context of supervised learning tasks.

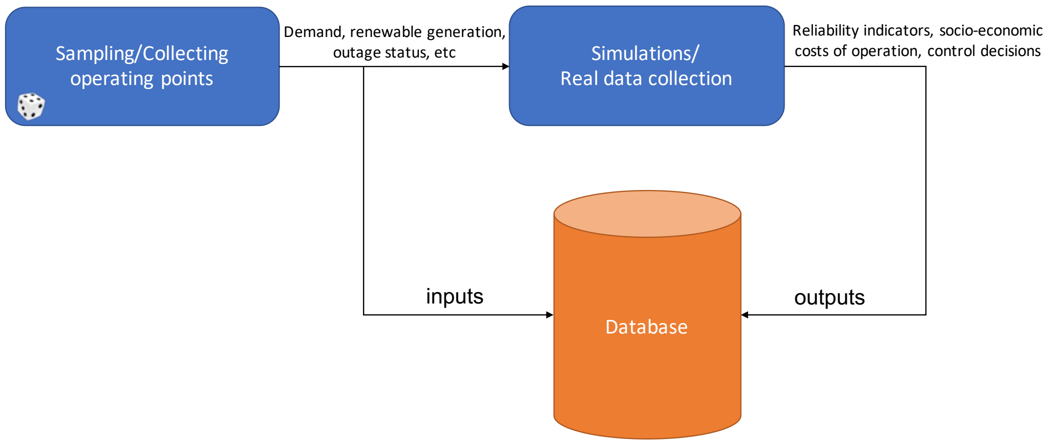

Sampling/Collecting operating points
Demand, renewable generation, outage status, etc
Simulations/ Real data collection
Reliability indicators, socio-economic costs of operation, control decisions
inputs
outputs
Database

Figure 4.3: Database building for static reliability management.

[3]The PMUs are devices allowing to monitor the power system state in real-time. They measure synchronously voltage phasors at buses where they are located and current phasors in the branches connected to these buses.

Generating a good dataset for security assessment and control, i.e. a dataset that correctly spans the range of possible operating states so that the learnt model will be able to generalize to future real operating conditions, is not an easy task. Indeed, the number of input variables is generally very large, they have complex marginal probability distributions and are highly correlated, with non-linear dependencies. Furthermore, security datasets are often imbalanced, with much more secure states than insecure ones, making it difficult for the machine learnt models to characterize correctly the security boundary. However, compared to datasets obtained from observational data, datasets from simulation can be larger, with more diversity and a better balance between secure and insecure samples.

Given the importance of the dataset generation to obtain good ML models, papers focusing mainly on building more effective databases for machine learning-based security assessment and control were published [Konstantelos et al., 2019; Venzke et al., 2019]. In [Konstantelos et al., 2019], the authors propose an approach based on Vine Copulas to capture the complex dependency between the input variables, in order to generate representative power systems states for machine learning tasks. On the other hand, Venzke et al. [2019] propose a method to generate datasets that characterize the security boundary and cover equally the secure and insecure regions. In particular, they introduce infeasibility certificates based on separating hyperplanes to identify, for large portions of the input space, the infeasible region.

Once a dataset is built, the data can be processed before being fed to a learning algorithm. This step may in some cases be mandatory to improve the quality of the predictions of learning algorithms and increase training speed. The pre-processing step often includes *feature selection* and *input data normalization*.

4.2 Approximation of results of power flow computations

A Power Flow (PF) computation (or *load flow*) allows to determine, from the network topology and loads and generators power injections, the bus complex voltages, and thus the power flows in transmission lines (the flows can indeed be computed from the complex voltages). It is mathematically modeled as a system of non-linear equations [Weedy et al., 2012]. It is often used in steady-state security analysis to determine if operational constraints such as voltage and power flow limits would be violated for given control decisions and for a given list of credible contingencies (generally the N-1 contingencies), and to infer from it security indices or the security status of the system. It can also be used to evaluate the voltage stability margin and system critical voltage stability areas.

Current solving methods are often based on the Newton-Raphson solver, but are quite slow, and it is why ML techniques have been applied to build proxies of PF computations or to learn features of the problem in order to enhance its solving.

Table 4.1 sets out the main target outputs of the ML methods used in the context of static security assessment, to replace or enhance PF computations, the exploited learning algorithms, as well as their corresponding references.

Table 4.1: Main ML approaches for PF problems and corresponding references published between January 2015 and March 2021.

Predicted quantities	Algorithms	References
Power flows or Complex voltages	Neural networks	Chen and Tate [2020] Donnot et al. [2018a,b,c, 2017] Donon et al. [2020a,b] Du et al. [2019b] Hu et al. [2020] Yang et al. [2019]
	Several algorithms	Schäfer et al. [2020]
(Composite) Security indices	Neural networks	Bhatt et al. [2017] Lekshmi and Nagaraj [2018] Paramathma et al. [2016]
	LASSO	Li et al. [2018b]
Security status	Tree-based methods	Gholami et al. [2016, 2015] Oliveira et al. [2017] Saeh et al. [2016] Sekhar and Mohanty [2016] Zhukov et al. [2017]
	SVM	Dhandhia et al. [2019] Thirugnanasambandam [2018]
	Neural Networks	Du et al. [2019a,b]
Voltage stability	Neural networks	Chakraborty and Saha [2016] Singh et al. [2016]
	SVM	Yun et al. [2017]
	Tree-based methods	Li et al. [2018a] Su and Liu [2018]
Critical voltage stability areas	Unsupervised learning	Colorado et al. [2016]
	SVM	Pérez-Londoño et al. [2017]
Power grid segmentation	Unsupervised learning	Marot et al. [2018]

4.2.1 Learning complex voltages and power flows

Several papers studied the possibility to replace the PF computation by a faster machine learnt proxy, either to predict directly the power flows in the lines or to predict the bus complex voltages. The main algorithm used for this application is the neural network algorithm.

For instance, Donnot et al. [2017] use a deep neural network to estimate power flows very quickly from the active and reactive load values, the active production and voltage setpoints and the topology corresponding to possible N-1 contingencies (encoded with a one-hot encoding method). The authors propose to exploit it to help operators in the control room to choose remedial actions such as network topology modification after a contingency. Improving their previous work, they introduced *guided dropout* to enable the estimation of flows for a range of power system topologies [Donnot et al., 2018c]. The guided dropout method uses some neurons that are only activated if the corresponding line/breaker is disconnected, and the authors show that with this approach, a proxy trained with only N-1 events can generalize to N-2 events. This proxy is fast, and can be used to rank (double) contingencies and estimate the risk of a grid state [Donnot et al., 2018a,b]. Finally, using observational input data collected by RTE (*Réseau de Transport d'Electricité*), the French TSO, a new type of neural network for predicting line flows in case of topology change is presented in [Donon et al., 2020b], where the authors introduce the *Latent Encoding of Atypical Perturbation* (LEAP net) architecture, so as to further improve the generalization to unseen topologies. The topology changes considered in this research are the splitting or merging of nodes at substations, which are more complex than line disconnection. In the training set, only unary changes compared to the reference topology are considered. However, the authors show that their models can generalize with good accuracy to two topology changes.

Schäfer et al. [2020] compare different learning algorithms to predict the line loadings and voltage magnitudes and show that the model with the lowest mean absolute error is the neural network model. They consider as inputs time series data corresponding to active power and voltage magnitude of buses with generators and active and reactive powers for other buses. They learn one model per N-1 case and so do not generalize to other topologies.

Recently, the physics of the problem has been incorporated in the learning process to improve the performances of the algorithms [Donon et al., 2020a; Hu et al., 2020; Yang et al., 2019]. For instance, using deep neural networks, Yang et al. [2019] build a model approximating PF calculations in the context of probabilistic PF, where PF problems must be solved for a large number of operating states, representing the uncertainty in load and renewable generation. They consider the physics of the problem in the learning process, by proposing a composite loss for training the neural network, based on the branch flows equations, and by adapting the gradients. The inputs of the model represent the injection power of all renewable energy sources and load demands. Hu et al. [2020] exploit a neural network in the context of multi-task learning.

The main task predicts the bus voltages, while the auxiliary tasks integrate the physical knowledge of the problem and act as regularizers, for instance by minimizing the power mismatches when exploiting the output of the main task to recompute the power injections and comparing them with the true power injections. Real-world load data are considered to train the model and the inputs are power injections of loads and generators, as well as voltage set points and voltage magnitude and phase of the reference bus. These two papers propose an approach that does not generalize to a change in topology.

Contrarily to the other papers, Donon et al. [2020a] propose a PF solver that does not try to imitate a classical PF solver, but learns by itself by minimizing the violations of Kirchhoff's law during training. It is based on graph neural networks. The inputs describe both the grid injections and grid information. For instance, a line is defined by its resistance, its reactance, etc. but also by the two bus indices connected by the line. The authors show that their solver can generalize to different power grid topologies, and even different grids without being retrained, and is faster than a classical Newton-Raphson solver.

Finally, Chen and Tate [2020]; Du et al. [2019b] propose to use deep convolutional neural networks to approximate the complex voltage values. In [Du et al., 2019b], a deep convolutional neural network, with the active and reactive power injections and a susceptance vector to indicate a change of topology during N-1 contingency as inputs, is used to both predict the complex bus voltages and the system security status (secure, alarm or insecure) for N-1 contingencies. Chen and Tate [2020] predict complex voltage values, from active and reactive power injections and DC power flow solutions. These voltages are then used as initial condition for the non-linear PF solver. Contrarily to the other papers in this section, the goal is not to replace the non-linear PF solver but accelerate the convergence by providing good initial conditions. This model needs to be retrained to consider other grid topologies such as N-1 contingencies.

4.2.2 Learning (composite) security indices

Load flow computations are also used to evaluate security indices, indicating violation of transmission constraints or voltage constraints, or to evaluate composite security indices, indicating violation of both bus voltage and power line transmission limits. Computing these indices in post-contingency states can help the operator to screen and rank the considered contingencies according to their severity. In [Paramathma et al., 2016], an artificial neural network is used to predict a security index for each considered contingency. In [Bhatt et al., 2017; Lekshmi and Nagaraj, 2018], RBF neural networks are used to predict a composite security index for each contingency, in order to rank them and identify the most severe ones. Finally, Li et al. [2018b] use a LASSO (Least Absolute Shrinkage and Selection Operator) method to estimate composite security indices.

The input variables of the proposed models describe the pre-contingency system state and the contingency (often with a binary variable per line indicating if the line is in outage). In all these papers, only the N-1 contingencies are tested, for variable load conditions.

4.2.3 Learning security status

The static security of the system can also be assessed with security labels. The labels are generally obtained from the values of (composite) security indices, computed with a PF. To speed up the evaluation of the static security state of the system, several publications propose to learn a model able to predict the system security state after the occurrence of a given contingency or to predict directly if the system is secure for all contingencies in a given set (for instance the set of N-1 contingencies).

The inputs of the proposed models are for most of these papers the pre-contingency state defined as active and reactive load values, generator output powers, power flows and voltage magnitude and phase at each bus. When it is not the case, the inputs of the models are explicitly mentioned.

Many papers propose to use tree-based methods. For instance Gholami et al. [2016, 2015] use decision trees and random forests for static security assessment, respectively to predict the N-0 status and the security status considering a set of single and double contingencies. The inputs of the models are only the active and reactive power injections at buses. Sekhar and Mohanty [2016] use decision trees and random forests to predict the N-1 security status and in [Saeh et al., 2016], the authors test four decision tree algorithms to classify the N-1 contingencies as secure or insecure. They also apply a sequential search algorithm for feature selection. In [Zhukov et al., 2017], a hybrid approach of random forests and tree boosting is used for the steady-state security analysis of contingencies. They consider four classes, normal, alarm, emergency correctable and emergency non-correctable. Finally, multi-way decision trees were exploited in [Oliveira et al., 2017] to determine if the system is secure or insecure regarding the violation of voltage magnitude limits, given the bus voltage magnitudes, the active and reactive loads and power generations, the network topology under operation and the considered N-1 contingency. Furthermore, a method called Stratified Random Sampling was used to obtain the same proportion of secure and insecure labels.

Some papers propose to use the SVM approach. For instance, Dhandhia et al. [2019] exploit the SVM approach to classify each N-1 contingency as secure, alarm or insecure and Thirugnanasambandam and Jain [2018] propose several ensemble classifiers (Adaboost) with SVMs as weak learners to evaluate the static security state of the system for a given N-1 contingency and then evaluate in case of insecure state the type of condition violation. Both feature selection and class imbalance correction are used to improve the performances of the method. In this paper, input variables indicating the contingency location are added to the inputs describing the pre-contingency state.

Regarding the neural network algorithm, Du et al. [2019a,b] use deep convolutional neural networks to classify the system security states between secure, alarm and insecure for various loads, renewable generations and N-1 line contingencies. The inputs are the power injections and a susceptance vector to indicate a change of topology during N-1 contingency.

For most of these papers, the proposed approach is able to generalize to load variation and N-1 contingencies when the model predicts the security status for a specific contingency. In [Oliveira et al., 2017] , the model can also generalize to different topologies and then to N-1 contingencies for these topologies.

4.2.4 Learning voltage stability status and margin

In the context of static voltage stability assessment, several load flows with varying load levels may be run to determine the voltage collapse point and thus the voltage stability status and voltage stability margin (distance to voltage collapse).

Fan et al. [2017] propose a method based on ensemble of feature selections and curve fittings to predict the voltage stability margin. In [Li et al., 2018a], the authors compare two feature selection methods to predict the class of static voltage security margin with a decision tree. They show that both methods are equivalent. They use operation variables such as bus voltages, reactive generation and reactive power flows as inputs. In [Su and Liu, 2018], the authors use a random forest model classifier with voltage magnitudes and phases as inputs for online voltage stability assessment and deal with the problem of frequent model update. Frequently updating the learning model is useful to take system changes (i.e. topology) into account but it can take time. To tackle this issue, the authors propose a random forest algorithm, where only part of the decision trees are updated each time instead of the whole model. They also perform feature selection based on feature importances, a by-product of random forest algorithms. Yun et al. [2017] exploit the SVR learning algorithm to estimate the severity of wind power output fluctuations, which is function of the stability margin, in order to estimate the static voltage security risk of wind power fluctuations. The input variables are the active powers of load and wind farm generation. Finally, Chakraborty and Saha [2016] use a neural network to predict a unified voltage stability indicator, based on a topology indicator and active and reactive power of some buses and Singh et al. [2016] exploit an RBF network to estimate a probabilistic insecurity index. The inputs are the line parameters (reactance, resistance, susceptance), the voltage at generator buses and the total system load, so that the model can generalize to a change in line parameters.

For most of these papers, the models are tested for various load conditions (and renewable generation for [Yun et al., 2017]) and N-1 contingencies.

4.2.5 Identifying critical voltage stability areas or segmenting the power grid

Machine learning approaches applied on data from PF computations can be used for detection of critical (or weak) voltage stability areas or to segment the power systems in several zones. For instance, in [Colorado et al., 2016], the authors exploit the unsupervised learning algorithm called k-means to identify voltage stability critical areas and Pérez-Londoño et al. [2017] combine the k-means algorithm with SVMs to first identify offline weak voltage stability areas and then determine online the weak area of each new operating condition. In both papers, the inputs of the clustering algorithms are the voltage phasors at each bus, the active and reactive power flowing through the lines and the voltage stability indices, while in [Pérez-Londoño et al., 2017], the inputs of the SVM classifier (to be used online) are only the voltages and power flows. In both papers, the method can generalize to load variations and N-1 contingencies, considering both generator and line outages.

Finally, Marot et al. [2018] propose to exploit an unsupervised learning technique to segment the power system into zones taking into account the real-time context, in order to help the operators managing the grid. The inputs of the unsupervised learning method are the real-time active power flows. The method is tested on several power grids of different sizes, from small grids of 14 buses to large grids such as the French power grid, and can be applied for different topologies.

4.3 Prediction of optimal power flow features and outcomes

A more recent application scope of ML for reliability management is to help out in solving OPF problems. The OPF and its extended version the SCOPF were introduced in sections 2.6.1 and 2.6.2. They are extensively used by the operators for operation planning, to find optimal decisions considering physical and operational constraints. They are also solved repeatedly during operation. However, these problems are non-linear and non-convex and generally large-scale. Furthermore, close to real-time, they must be solved within a short period of time. Solving OPF problems is thus still a computational challenge, that has been recently addressed with the help of ML techniques. In the literature, some papers approach this problem by building proxies predicting the decisions or related costs of an OPF while others try to learn features of the OPF to accelerate its resolution or learn security rules to enhance its formulation. To summarize the different approaches and the exploited learning algorithms, the references discussed below are sorted in Tables 4.2 and 4.3.

4.3.1 Learning optimal decisions

Most papers predicting directly the outputs of an OPF problem predict the optimal decisions given by the program, which are usually generator setpoints (generator active power and voltage magnitude). In the literature, the dominant approach for this task is to exploit *deep learning*. The main difficulty of this approach is to guarantee that the predicted solutions are feasible regarding the physical and operational constraints of the system. For this reason, a

Table 4.2: Main ML approaches to solve OPF problems and corresponding references published between January 2015 and March 2021 - part 1.

Predicted quantities	Algorithms	References
Decisions	Neural networks	Baker [2020a,b]
		Biagioni et al. [2020]
		Chatzos et al. [2020, 2021]
		Chen et al. [2020a]
		Diehl [2019]
		Dong et al. [2020]
		Falconer and Mones [2020]
		Guha et al. [2019]
		Huang et al. [2021]
		Mak et al. [2021]
		Owerko et al. [2020]
		Pan et al. [2020a, 2019, 2020b]
		Venzke et al. [2020a]
		Velloso and Van Hentenryck [2020]
		Zamzam and Baker [2019]
		Zhao et al. [2020]
	Extreme learning machine	Lei et al. [2020]
		Rahman et al. [2021]
	Tree-based methods	Baker [2019]
		Rahman et al. [2021]
	Reinforcement learning	Yan and Xu [2020]
		Zhou et al. [2020a,b]
Related costs	Several algorithms	Canyasse et al. [2017]
		Duchesne et al. [2017, 2018]
	Neural networks	Duchesne et al. [2020a]
Security status	Neural networks	Du et al. [2020]
		Sun et al. [2018]
		Urgun and Singh [2018]
	kNN	Urgun and Singh [2019]

large number of papers propose different methods to ensure the feasibility of the solution, either by adapting the training process or by post-processing the

Table 4.3: Main ML approaches to solve OPF problems and corresponding references published between January 2015 and March 2021 - part 2.

Predicted quantities	Algorithms	References
Features of the OPF problem	Neural networks	Ardakani and Bouffard [2018] Chen and Zhang [2020] Deka and Misra [2019] Falconer and Mones [2020] Robson et al. [2019] Zhang et al. [2020]
	Statistical learning	Ng et al. [2018] Misra et al. [2018]
	Linear regression	Mezghani et al. [2020]
	Polynomial regression	Hu et al. [2021]
	Tree-based methods	Prat and Chatzivasileiadis [2020]
Embedded security rules	Tree-based methods	Cremer et al. [2018, 2019] Halilbašić et al. [2018] Hou et al. [2020] Thams et al. [2017]
	Neural networks	Nguyen-Duc et al. [2017] Venzke et al. [2020b]
	SVM	Zhou et al. [2018b]

predicted solution. The rest of this section is organized regarding the purpose of the learnt models.

4.3.1.1 *Optimal decisions from AC-OPF or DC-OPF problems*

In order to obtain fast optimal decisions, deep learning is used in [Pan et al., 2019; Zhao et al., 2020] and in [Chatzos et al., 2020; Huang et al., 2021; Pan et al., 2020a; Zamzam and Baker, 2019] to predict the decisions of respectively a DC-OPF and an AC-OPF. In all these papers a method ensuring the feasibility of the solution is described. For instance, in [Pan et al., 2019], if the predicted solution is not feasible, the authors solve an optimization problem to find the feasible solution closest to the predicted one. In [Zhao et al., 2020], the initial constraints of the DC-OPF are calibrated (e.g. the line capacity limits are reduced) in such a way that despite prediction error, the solution remains feasible. In [Zamzam and Baker, 2019], the output of the deep learning model is constrained with a sigmoid function to adhere to active power generation and voltage magnitude constraints and a PF problem is then solved based on the predictions to compute other dependent variables such as the reactive output power of generators and the voltage phase at buses. Solving a PF is indeed faster than solving an OPF. Finally, in [Chatzos et al., 2020], the authors take advantage of deep learning and a dual Lagrangian method to obtain high fidelity approximations of OPF and enforce physical and operational constraints.

Graph neural networks can also be used to predict the generator output powers. Owerko et al. [2020] show that graph neural networks offer better scalability to larger networks than classical feed-forward neural networks and Falconer and Mones [2020] show that graph neural networks outperform feed-forward neural networks and convolutional neural networks for this task.

Deep neural networks models do not scale well with the size of the power system. Indeed, models for large power systems have a very large number of parameters, which slows down convergence and can impact the accuracy of the model. To obtain more scalable models, [Mak et al., 2021] propose to reduce the input feature dimension by learning a load embedding scheme to aggregate the loads on adjacent buses with an encoder. Another proposed approach to improve the scalability of these models and reduce the training time is to exploit a spatial network decomposition. In [Chatzos et al., 2021], the power system is divided into several regions and a two-stage approach is used to predict the solution of an AC-OPF: the first stage predicts the voltages and power flows in the nodes and lines coupling the regions and the second stage predicts the AC-OPF solution for each region. The advantage of small training time is that the model can be quickly retrained in case of topology change.

To improve the training speed and avoid dealing with hyper-parameters tuning of deep neural networks, Lei et al. [2020] propose to predict the generator active and reactive output powers with a three-stage stacked extreme learning machine. Furthermore, they pre-classify the samples based on their *active constraints* (constraints that are satisfied at the equality for the optimal solution) and learn one model per class in order to further improve the performance of the proposed approach. Rahman et al. [2021] also use extreme learning machine to predict the solution of an AC-OPF, more specifically the complex voltages. They compare it with tree-based methods and use the PF equations to compute the other operating variables to ensure compliance with physical constraints.

All the proposed models are trained for a given topology and most probably do not generalize well to changes in topology. This problem is considered in [Chen et al., 2020a], where the authors use a meta-learning approach to find an initialization point enabling fast training for different system configurations.

Finally, instead of using supervised learning to directly predict the decisions of an AC-OPF, Yan and Xu [2020]; Zhou et al. [2020a,b] use a deep reinforcement learning approach.

All the models in this section consider as inputs the active load demand at each node (for DC-OPF) and active and reactive load demand at each node (for AC-OPF). The datasets for training and testing are mostly built by sampling uniformly each load in an interval centered around a default value, except for [Chatzos et al., 2020; Lei et al., 2020; Rahman et al., 2021; Yan and Xu, 2020; Zamzam and Baker, 2019] that use more complex sampling methods. For instance, Zamzam and Baker [2019] use a truncated Gaussian distribution. Rahman et al. [2021] consider observational hourly load profiles of several years from the Texas system operator. This allows the model to be able to generalize to seasonal change of the load data. In [Lei et al., 2020; Zhou et al., 2020a], renewable generation is also considered, while in [Falconer and Mones, 2020], grid parameters such as line thermal ratings, reactance and resistance of transmission lines and maximum and minimum output power of generators are added to the inputs, so that the model can generalize to small changes in these parameters. Finally, in [Zhou et al., 2020a], the authors also consider as inputs elements of the admittance matrix so that their method can generalize to N-1 events such as line tripping.

4.3.1.2 *Optimal decisions from SCOPF problems*

Fewer papers addressed the SCOPF problem with ML. Velloso and Van Hentenryck [2020] propose a deep learning approach combined with robust optimization to predict the solution of a DC-SCOPF problem. Their SCOPF formulation considers the *automatic primary response* mechanisms after a contingency, that adjust the power output of synchronized generators to restore power balance in a few seconds. The contingencies considered are single generator outages. The authors predict from the load value at each bus the preventive generations and post contingency variables representing the automatic primary response. They ensure feasibility of the predicted solution first with a dual Lagrangian method and then by finding the closest feasible solution to the prediction. Also dealing with DC-SCOPF, but considering only preventive actions and the contingencies due to outage of any single line, Pan et al. [2020b] use a deep neural network to predict from the load inputs the (preventive) generations and then reconstruct the voltage angles with a linearized PF. They also propose a post-processing procedure to ensure the feasibility of the final solution.

4.3.1.3 *Emulating the OPF solver*

With a different approach, Baker [2020a,b] proposes to use deep learning models to emulate an iterative algorithm solving the AC-OPF problem. The neural network model takes as input the solution of the previous step and updates this solution. In [Baker, 2020b], when close to convergence, a PF is solved to obtain a feasible solution. This method is faster than traditional method solvers. However, it does not generalize to topology change.

4.3.1.4 *Warm-start*

While still predicting the optimal decisions of an OPF problem, several papers propose to use these solutions as initial conditions (warm-start points) for the iterative OPF solvers, in order to improve convergence. This guarantees the feasibility of the final solution.

For instance, Baker [2019] proposes to learn the outputs of an AC-OPF problem with a random forest and shows on some test systems that it leads to a faster convergence time compared to other warm-start methods. Dong et al. [2020] use neural networks, and Diehl [2019] uses graph neural networks to predict the initial conditions. In the context of distributed DC-OPF, where the network is decomposed into independently operated partitions, Biagioni et al. [2020] use recurrent neural networks to improve convergence.

All these papers consider the load value at each bus as inputs. These ones are sampled uniformly around default load values, except for [Diehl, 2019] that considers synthetic hourly load distributions over one year.

4.3.2 Learning socio-economic costs

Rather than predicting the decisions obtained from an OPF, some papers [Canyasse et al., 2017; Duchesne et al., 2020a, 2017, 2018] are interested in the (optimal) cost of operation related to the decisions. In three of these papers, several learning algorithms are tested and compared to predict the cost of real-time operation obtained from solving an AC-OPF [Canyasse et al., 2017] and a DC-SCOPF with N-1 security constraints [Duchesne et al., 2020a, 2017, 2018] problem. In [Canyasse et al., 2017], the inputs of the models are the load values and it is shown that the model can generalize to change in load along a day. The considered context in [Duchesne et al., 2020a, 2017, 2018] is the day-ahead context, so that the inputs of the models are, among others, the real-time load and wind generation realizations, the hour of the day and the generator output powers decided in day-ahead for the latter two. In [Duchesne et al., 2017], line, generator and transformer outages are considered as inputs so that the model can also generalize to a change of topology.

4.3.3 Learning security status

Another application of OPF problems is determining if the demand could be met at all buses for given operating conditions. Coupled with Monte-Carlo simulations to generate many operating states, it can help assess reliability of the system with reliability indices such as *loss of load probability*.

In [Urgun and Singh, 2018, 2019], the authors use a multi-label classifier to predict from the generation at each generation bus if the demand is met at each bus. The training and test states are generated with Monte-Carlo simulations, considering constant load values corresponding to the system load peak level, but varying generator outages, which are sampled based on their failure and repair rates. In [Urgun and Singh, 2019], the available capacity of transmission lines is also considered to model possible line failures in addition to the generator failures. The classifier allows them to estimate loss of load indices considering generator outages (and transmission outages for the latter) without solving an OPF problem for each state, accelerating the reliability assessment of the system. The method does not generalize to different load levels, new classifiers must be learnt in that case.

In [Sun et al., 2018], the authors propose a deep-learning based feature extraction framework based on deep autoencoders to automatically extract effective training features to be fed to a security classifier such as a decision tree, in order to assess the security of the system. The simulator exploited to build the training dataset is the DC-OPF, which computes the post-contingency security status of the system, for example by verifying that no loss of load is necessary to satisfy the operational constraints. The inputs of the feature extraction module correspond to the system state (active loads, power injections, generator power outputs, voltage angles and power flows), as well as information about the contingency one wants to assess, encoded with a one-hot encoding. The model is trained with observational data from the French transmission system over several months, enriched with samples obtained with an R-vine copulas method. The approach can generalize to various load and wind generation conditions, as well as N-1 line contingencies but not to other topologies.

The OPF can also be used to determine the state of the system when only the demand at each node and the topology of the system (given by the susceptance matrix) are known. In [Du et al., 2020], the authors propose to learn the complex bus voltages with a convolutional neural network model. These predicted state variables are then used as inputs of a feed-forward neural network to predict, given a topology change (due to contingency), a security index value. This approach is used for fast cascading outage screening and can generalize to different loading conditions and system topology changes.

Finally, Venzke et al. [2020a] introduce a framework based on mixed-integer linear reformulations of neural networks to compute worst-case guarantees of these models. They study the physical constraints violations, the distance between the prediction and the optimal decisions, and the sub-optimality when these models predict the security status from the solution of a preventive N-1 DC-SCOPF or a preventive N-1 and small-signal stability AC-SCOPF.

4.3.4 Learning features of OPF problems

Machine learning techniques have been proposed to reduce the computational burden related to OPF solving, by learning features of the problem instead of directly predicting its outcomes. The objective is to help the OPF solver in order to accelerate the resolution, ideally without compromising the accuracy of the solution.

Some papers propose to reduce the search space of the OPF problem and thus accelerate the resolution by identifying important constraints in the problem or important scenarios in the case of a stochastic OPF formulation, others to learn convex approximations of the equations.

4.3.4.1 *Active constraints*

Some papers propose to learn the set of active constraints, i.e. the constraints which are all satisfied with equality for a given solution, at the optimal solution. The optimal solution may then be obtained by solving the problem with this reduced set of constraints.

In [Ng et al., 2018] and [Misra et al., 2018], the authors propose a method to solve a stochastic DC-OPF considering uncertainties, that will adjust the generation in response to uncertainties. In this context, the DC-OPF must be solved within a short time period and one proposed approach is to pre-define a piecewise affine policy, referred to as the ensemble control policy in the papers, before the realization of uncertainties, in such a way that in real-time, one has to find the corresponding affine control policy for the realization of uncertainties, instead of solving the full OPF. Each affine policy can be derived from the set of active constraints at the optimal solution of a given uncertainty realization. To define the ensemble control policy, one can thus search for all the bases (i.e. the sets of active constraints) of the DC-OPF given the distribution of uncertainties, and then to each basis associate an affine policy. When a scenario is realized, one can look for the basis at the optimal solution of this scenario and then apply the corresponding affine policy.

In order to find an ensemble control policy more efficiently, Ng et al. [2018] and Misra et al. [2018] leverage statistical learning to identify the most important bases of the real-time DC-OPF, i.e. the bases with the largest empirical probabilities of occurrence given the distribution of uncertainties (active load at each bus in their case study). By computing affine control policies for only a subset of bases, the authors reduce the computational burden in real-time, allowing to solve these parametric programs more efficiently online.

Following these works, Deka and Misra [2019] exploit neural networks to learn a mapping between realized uncertainties and the corresponding basis at their optimal solution, among the ones previously found with the statistical method, allowing to enhance even more the computational efficiency of the approach.

In the context of bi-level optimization problems, Prat and Chatzivasileiadis [2020] use decision trees to learn set(s) of active constraints of the lower level problem (e.g. a DC-OPF) and thus reduce the size of the bi-level problem by considering only active constraints in the lower level. The context is a strategic producer optimizing its bidding price in the day-ahead market. The inputs of the model are the load per bus (and the price bid optimized in the upper level for one of the proposed approaches) and the total system load.

To find the active constraints in a linear network flow problem (e.g. a DC-OPF), Chen and Zhang [2020] do not use a classifier to determine for each constraint if they are binding at the optimal solution. Instead they propose to build a neural network model that predicts from a load vector the corresponding optimal objective value of the problem. The gradients of the neural network model with respect to the inputs are then used to predict the active constraints in the problem. The solution of the initial problem is finally obtained by solving a linear program with only the found active constraints. This approach is further improved in [Zhang et al., 2020]. The authors use a convex neural network to keep the convex relationship between the loads and the objective value (a cost) of the DC-OPF, and add the Karush-Kuhn-Tucker optimality conditions in the training loss in order to improve the generalization performance of the method.

Predicting the active set of constraints for AC-OPF problems, Falconer and Mones [2020] compare different neural networks architecture. Robson et al. [2019] also use multi-label neural networks, but to predict the active status of each constraint for both DC- and AC-OPF problems. They then solve the reduced OPF problem iteratively by adding at each iteration the constraints of the full problem that are violated with the optimal solution of the reduced formulation. The total computational time of the iterative method is incorporated in the loss function during training in order for the learnt model to predict the set of constraints that will minimize this computational time. They consider the close to real-time context for which OPF problems must be quickly solved. For both papers, the inputs are the load values, as well as grid parameters such as line resistance, line reactance, line thermal ratings, maximum and minimum generator output powers, so that the model can generalize to small variations (10%) in the grid parameters.

4.3.4.2 *Umbrella constraints*

The problem of reducing the search space is tackled differently by Ardakani and Bouffard [2018]. Instead of predicting the set of active constraints, they propose to predict the set of umbrella constraints, which corresponds to the set with the minimum number of constraints such that if one constraint is removed, the set of feasible solutions of the original problem is modified. Solving an OPF problem with only the umbrella set of constraints may reduce significantly the solution time. They use one neural network classifier (with the load value for each bus as inputs) per constraint to determine if the constraint is an umbrella constraint or not and they test their approach on a DC-SCOPF. The dataset used to train the classifier and test the approach spans a full operation year, so that the approach can generalize to different load profiles.

4.3.4.3 *Scenarios*

Mezghani et al. [2020] use the scenario-based approach to deal with short-term uncertainties when solving OPF, where power flow equations are solved for a number of scenarios sampled from the distribution of uncertainties. They consider an iterative approach, for which the scenarios used in the OPF problem are modified at each iteration, until the solution satisfies a security criterion (i.e. a sufficient number of independent scenarios are secure). They propose to exploit machine learning to reduce the number of scenarios needed in the OPF problem while keeping an accurate uncertainty quantification. In particular, they modify some of the load values of the selected scenarios so that they violate even more the constraints that they were already violating. For that, they use linear regression to model, for each violated constraint, the relation between the loads per bus and constraint violation, in order to identify the loads that should be modified, i.e. the ones that have more impact on the constraint violation.

4.3.4.4 *Convexification*

Instead of reducing the search space by identifying the important constraints or important scenarios, Hu et al. [2021] propose to learn a convex quadratic approximation of the three-phase AC power flow equations (in rectangular coordinates) with an ensemble method in order to convexify the AC-OPF problem. More precisely, they learn convex quadratic approximations of the relation between the bus voltages and the active and reactive power injections and line flows, where the voltages are the inputs of the learnt models and the power injections and flows the outputs. They compare their data-driven convex approximation approach with a semi-definite programming relaxation and show on several case studies of different sizes that their method outperforms the semi-definite programming method in computational efficiency.

4.3.5 Embedding machine learnt security rules in OPF problems

Another main approach to enhance OPF problems with ML consists in building models of dynamic (and/or static) security assessment and then extracting security rules from these models that can be embedded in OPF problems, to obtain useful control actions (decisions) considering security. Indeed, classifiers built with machine learning contain knowledge about stable and unstable regions.

Several papers propose to learn the security rules with a tree-based method. The learnt security rules can then be embedded in the OPF formulation as mixed integer linear equations. This results in a problem to solve being either a MILP or a Mixed Integer Non Linear Programming (MINLP) problem, depending if the initial OPF problem is a DC-OPF or an AC-OPF problem.

Cremer et al. [2018] exploit decision trees and embed the rules determining the output of the security classifier in a decision-making problem (i.e. an OPF problem). This allows to compute control decisions considering the stability boundary. In their paper, the authors present the challenges of this approach, which are the computational complexity to build the database and the accuracy of the learnt rules. This approach is further developed in [Cremer et al., 2019], where learnt condition-specific safety margins are proposed to be incorporated in a decision-making program. These margins allow, according to the authors, to improve the risk/cost balance. An ensemble of decision trees (Adaboost) is used to perform probabilistic security control. In the case study of these papers, the security classifier predicts from the pre-fault operation state (loads and generator powers) the security status of the system, secure corresponding to no loss of load for all the N-1 line outages, and is embedded in a DC-OPF problem.

Thams et al. [2017] propose to build with machine learning line flow constraints to be incorporated in a market clearing program (under the form of a SCOPF problem) to improve both small-signal stability and N-1 security. They use a decision tree-based classifier to extract knowledge. The decision tree rules consist in conditional line transfer limits, that can be embedded in the SCOPF formulation in order for the operator to take decisions already in line with the small-signal stability margin. An extension of this work to solve an AC-SCOPF instead of a DC-SCOPF is proposed in [Halilbašić et al., 2018], while still incorporating small-signal stability N-1 security with decision tree-learnt rules. The authors then relax the resulting MINLP problem with a second-order cone relaxation in order to solve a convex MIP problem. In these papers, the decision tree predicts from the loads and generator powers the small signal stability of the system, considering N-1 line outages and bus faults.

Finally, Hou et al. [2020] propose to embed N-1 security in an economic dispatch problem. They learn the rules with a sparse weighted oblique decision tree, which has as inputs the pre-contingency active load values, generator output powers, renewable generations and power flows. The training database is built based on observational renewable generations and load values.

Security rules obtained from neural network models can also be expressed exactly as mixed integer linear constraints if the activation functions are ReLU functions. Venzke et al. [2020b] incorporate N-1 security and small-signal stability in an AC-OPF formulation with rules learnt with a neural network algorithm. The inputs of the model are the pre-contingency control variables (e.g. generator active powers) and the database is built by discretizing the set of possible values of the generating output powers. They propose to iteratively linearize the non-linear nodal power balance equations in the AC-OPF to formulate the resulting optimization problem as MILP problems.

Using regression models predicting the critical clearing time (the maximum time available to clear a disturbance before the system becomes unstable) of a contingency instead of the security status, Nguyen-Duc et al. [2017] are solving an OPF considering transient stability constraints. For each considered contingency, the corresponding critical clearing time is approximated by a neural network model (with as inputs the pre-contingency generator active and reactive output powers) that is then linearized to be embedded in the OPF formulation. This approach helps to define preventive decisions that are such that the critical clearing time of all considered faults is greater than a defined minimum value.

Finally, Zhou et al. [2018b] build a two-stage SVM model to determine the transient stability region that is embedded in a decision-making program to determine preventive control actions. The outputs are the pre-contingency generator active and reactive powers, the generator voltages, the rotor angles and the power flows in the lines and there is one model per contingency. The final transient stability-constrained OPF being non-linear and non-convex, the authors propose to solve it with particle swarm optimization.

4.4 Prediction of unit commitment features and outcomes

The UC problem, already introduced in section 2.6.3, consists in deciding in advance which generating units should be on or off for the time horizon considered. It is a hard optimization problem to solve, given the large number of variables, the presence of discrete variables and the temporality of some constraints. Enhancing the solving or building proxies of the UC problem is therefore useful, especially for very large systems or in applications for which a large number of UC problems must be solved, for instance in the context of long-term planning where multiple scenarios are studied.

In order to summarize this section, Table 4.4 presents the ML approaches used in the context of UC problems as well as the corresponding references. Most of the found publications exploit ML to build models predicting directly the commitment of generating units but some papers also learn features of the problem to help its solving or learn security rules to enhance its formulation.

Table 4.4: Main ML approaches to solve UC problems and corresponding references published between January 2015 and March 2021.

Predicted quantities	Algorithms	References
Decisions	kNN	Dalal et al. [2018, 2019]
	Reinforcement learning	Dalal and Mannor [2015]
		Dalal et al. [2016]
		Jain et al. [2018]
		Wang et al. [2019]
		Zhou et al. [2018a]
Features of the UC	kNN	Pineda et al. [2020]
		Xavier et al. [2020]
	SVR	Xavier et al. [2020]
Embedded security rules	SVR	Singhal et al. [2018]
	Neural networks	Zhang et al. [2021]

4.4.1 Learning UC decisions

Building a fast proxy of the UC problem can be useful to model short-term operation. For instance, Dalal et al. [2018] need to quickly evaluate the outcome of short-term operation, for a (mid-term) outage scheduling purpose. They propose to use k Nearest Neighbors (kNN) as a proxy of short-term operation and thus the predicted unit commitment schedule is the schedule in the learning database with operating conditions closest to the one evaluated. The inputs of the proxy are the 24-hour day-ahead load and wind generation forecasts and the daily network topology (availability of the transmission lines). This proxy can be exploited for maintenance scheduling, by using short-term operation proxies to quickly evaluate the impact of a maintenance decision on power system operation [Dalal et al., 2019].

The UC problem can also be addressed with a reinforcement learning approach. In [Dalal et al., 2016; Dalal and Mannor, 2015; Jain et al., 2018], the UC problem is modeled as a Markov Decision Process (MDP). To model the complex dependencies between the different time scales of decision-making in power systems, Dalal et al. [2016] propose an approach based on interleaved MDPs. In the paper, one MDP represents the day-ahead context where the generators participating to next day generation are chosen and the other the real-time context, where the operator can take preventive actions such as generation redispatch to adapt to the real-time state. In day-ahead, the operator takes decision based on a 24-hour load and wind generation forecast, while in real-time, the state of the system is described by the realized load and wind generation for the time horizon considered (an hour) as well as by a vector indicating the current state of each transmission line (available or not). The real-time process is used as a proxy to assess the reliability of decisions taken in day-ahead process. In [Jain et al., 2018], this approach is also applied but to minimize operation cost or CO_2 emissions.

All the proposed approaches generalize to different load and wind forecasts of different days and to variation in grid topology.

The unit commitment problem can also be defined as a multi-objective problem, for example to both minimize operation costs and maximize system reliability [Zhou et al., 2018a] or minimize both operation costs and wind curtailment [Wang et al., 2019]. In both papers, uncertainties of load and wind generation are considered. To solve this difficult problem, they exploit reinforcement learning-based particle swarm optimization.

4.4.2 Learning features of the UC problem

The approach to help out OPF solving can also be applied for UC problems. For instance, Pineda et al. [2020] propose to reduce the dimensional complexity of a transmission-constrained UC problem by learning with the kNN algorithm the congestion status of transmission lines based on the net demand at each node from historical data and then disregarding lines that will not become congested, i.e. removing redundant or inactive constraints.

Going further in the reduction of the problem size, Xavier et al. [2020] use ML techniques to build three complementary predictors that will help the solving of the UC. First, they learn with a kNN algorithm which security constraints need to be enforced in the problem and which transmission constraints can be safely omitted. Then they learn from historical data with a kNN the best warm-start point (initial conditions for the solver) for the optimization problem and finally, they predict with an SVR model affine subspaces where the optimal solution is likely to lie. These affine subspaces are added as constraints to the problem formulation in order to reduce the search space. The uncertainties considered are the production and start-up costs, the peak total load, the temporal load profile and the distribution of the total load between buses.

In both works, changes of the network topology are not considered.

4.4.3 Embedding machine learnt rules in UC formulations

With the objective of facilitating real-time operation, Zhang et al. [2021] embed frequency constraints in the UC formulation to improve the frequency stability (and thus dynamic security) of the system. For that, two deep neural networks are used to learn the frequency response and are then incorporated in the UC formulation as a set of mixed-integer linear constraints. The inputs of these models are each generator commitment status, the magnitude and location of the largest generator that could be in outage to represent the worst-case N-1 generator contingency and, additionally for the second model the active power injections of each generator. These models can generalize to ranges of operating conditions (load, wind generation and synchronous generator output power). On the other hand, Singhal et al. [2018] use the linear SVR algorithm to model reserve response sets after generator contingencies, more precisely to predict from the amounts of active/deployed reserves the post-contingency line flows. These linear models are learnt considering multiple net load possible realizations that represent the uncertainties and are incorporated in the UC formulation to account for post-contingency congestion patterns and uncertainties and therefore improve reserve scheduling and allocation.

4.5 Discussion

In this chapter, we reviewed recent works tackling (most often) static reliability assessment and control problems with various ML techniques. Indeed, a large increase in the number of publications in that particular field was observed recently. Many papers propose new applications, such as using deep learning to predict power flows, predicting the outcomes of OPF problems with simplified models, learning features of OPF or UC problems, and embedding machine learnt security rules in their formulations.

4.5.1 Challenges for ML approaches

Despite the great potential of these techniques, some challenges still must be faced before ML becomes common practice for reliability assessment and control in the electric power systems industry. We detail hereafter some challenges that we believe should be addressed by further research and development activities.

A first challenge concerns the acceptance of these new methods by the human users (operators and planners) in the industry. The traditional approaches used in practice for reliability management are model-based and take explicitly into account well-known physical laws. Given the practical consequences of a failure in assessment or control, moving towards a data-driven approach is difficult. There is definitely a need to convince the field experts that these methods are actually efficient and reliable. First, approaches using ML should be used in parallel with the more traditional approaches, to allow the human experts to assess their accuracy and usability; subsequently both approaches could be used in symbiosis, where traditional approaches would only be applied when the machine learnt predictors are not confident enough in their predictions. Furthermore, we believe that anyhow interpretability of the machine-learnt models is something that could not be neglected in future research.

Another challenge comes with the fact that electric power systems are constantly changing, due for instance to topology changes (e.g. forced or planned outages of elements), the increasing penetration of renewable generation and the seasonality of the load profile. Therefore, ensuring the adaptability and proposing ways to maintain over time the quality of the machine-learnt models used for assessment and control is also a requirement for the practical acceptance of ML applications to reliability management of electric power systems.

With the rise of data-driven methods, vulnerability of machine learnt models against man-crafted adversarial data must also be considered [Chen et al., 2018b] and techniques to detect these adversarial examples should be developed. More broadly, an important aspect not to be neglected is the study of guarantees on the performances of ML algorithms, in particular to avoid unexpected or harmful behavior [Amodei et al., 2016; Dobbe et al., 2020]. This is necessary for system operators to trust new proposals of ML for reliability management.

4.5.2 Future research directions

Beyond these current practical challenges discussed hereabove, we also see many interesting future research directions in the field of ML for reliability management of large-scale systems of systems such as electric power systems.

As a first direction for future research, we believe that reliability databases and evaluation protocols should be built and made publicly available. It would allow to build models with more representative datasets that better cover the range of operating conditions and to more easily compare the various methods proposed in the literature. This would help researchers in that field to advance more rapidly. Similarly, data shared by the industrial system operators would also help the research community, but other types of problems such as privacy, safety, and commercial sensitivity, are major obstacles for letting this happen in a near future.

Another direction of research, quite new, that we expect to be more developed in a recent future, is the use of machine learnt *proxies* to model the behavior of other parts of the overall multi-energy system. These can be smaller subsystems or other large-scale systems interacting with the managed one, such as distribution grids, other interconnected transmission systems, gas transportation systems, electric vehicle charging infrastructures, district heating systems, etc. One can also consider the integration of different time scales for sequential decision-making, and then use for instance proxies modeling the real-time operation in order to enhance the decision-making process in short-term operation planning (as is proposed in this thesis) or proxies of post-contingency behavior of emergency control systems to be exploited in the context of preventive mode dynamic security assessment.

To conclude, we think that the proposed methods are of great potential to improve reliability assessment and control and we expect more and more applications, both in research and in industry, to be developed while exploiting ML techniques. Finally, we believe that the methods mentioned and the identified challenges and future directions of research are relevant for many other large-scale systems and infrastructures, even beyond energy systems, such as distribution grids, micro-grids and multi-energy systems.

Part II

Machine learnt proxies for reliability management in operation planning

5

Machine learnt proxies of real-time operation

Overview

In this chapter we present a methodology based on supervised machine learning to build simplified models of Real-Time (RT) reliability management response to the realization of uncertainties. Our response models predict in particular the real-time operation costs and the resulting reliability level of the system. We test our methodology on the IEEE-RTS96 benchmark. Furthermore, we show how feature 'importances' computed by tree-based ensemble methods can be used to extract the most relevant variables to predict the response of real-time reliability management, and thus obtain a better understanding of the system properties.

References: This chapter is an adapted version of the following publication:

Duchesne, L., Karangelos, E., and Wehenkel, L. (2017). Machine learning of real-time power systems reliability management response. In *2017 IEEE Manchester PowerTech*, pages 1–6. IEEE.

Terminology and notations have been slightly adjusted for the sake of consistency with the rest of this manuscript. The text has also been processed to minimize overlap with respect to previous chapters.

5.1 Introduction

With the increasing levels of uncertainties in the context of short-term operation planning, the traditional approach for determining operation planning decisions, based on a single 'most likely' forecast along the considered look-ahead horizon, is not appropriate anymore. Indeed, the observed real-time realizations are farther and farther from this forecasted trajectory, making compliance with the reliability target more and more difficult. To progress, one possibility is to plan operation over a representative set of possible future operating conditions while modeling the way the real-time operator would respond along these trajectories. The purpose would be to choose operation planning decisions making the compliance with real-time reliability targets feasible with high enough probability while minimizing the expectation of operating costs.

For this approach, it is necessary to model in a suitable way the real-time reliability management strategy over many time steps and many look-ahead scenarios, which implies a challenging computational burden. To make this tractable, we propose to apply machine learning in order to automatically build *proxies* of the outcome of real-time reliability management; these proxies should be orders of magnitude faster to compute, and at the same time sufficiently accurate, so that they can be used instead of the detailed SCOPF type of computational models in the context of operation planning reliability management. Reliability management in real-time can indeed be modeled suitably by a SCOPF and the N-1 criterion.

Considering this, the contribution of this chapter is twofold. First we investigate the use of machine learning (and in particular supervised learning) to predict some outputs of real-time reliability management such as the costs of real-time recourse decisions and the level of system reliability they induce. In particular different supervised learning algorithms are tested in order to evaluate which ones are the most appropriate for this problem and the time gain obtained with the machine learnt proxies with respect to the full SCOPF computation is evaluated. Then we show that with the by-products of some supervised learning algorithms (tree-based ensemble methods) we are able to analyze the relevance of features to predict the studied outputs.

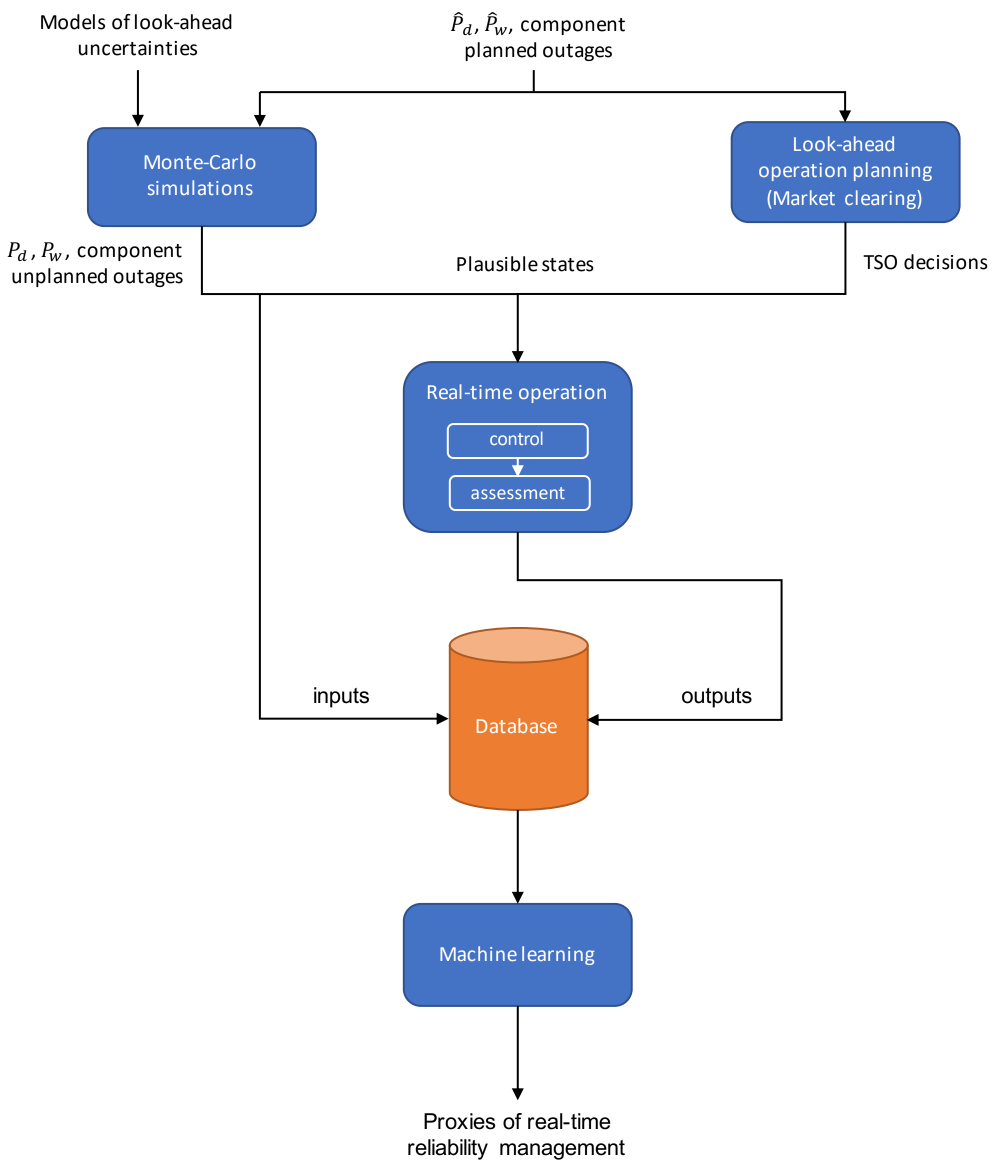
This chapter is organized as follows. Section 5.2 describes the proposed methodology for using (supervised) machine learning for the construction of proxies of real-time reliability management. Section 5.3 presents an empirical study of this proposal on the IEEE-RTS96 benchmark. Finally Section 5.5 concludes and outlines directions for future research.

5.2 RT-operation proxy building methodology

We start by modeling RT-operation in the form of a SCOPF program, followed by the assessment of the resulting reliability level gotten via a cascade simulator. Then, in order to use supervised learning to predict the outcome of RT operation, we build a database whose inputs describe the RT operating conditions and whose outputs are the outputs of this SCOPF program and of this reliability assessment program. Figure 5.1 depicts the whole methodology. Section 5.2.1 describes how the database may be generated with the help of Monte-Carlo simulations and in a look-ahead context, while section 5.2.2 describes how supervised learning algorithms can be used to build the proxies and how these algorithms should be evaluated.

5.2.1 Database generation

Let us, for the sake of explanation, consider the day-ahead operation planning context. Our database in such case would include a look-ahead horizon spanning the full 24 hours of the upcoming day and several plausible states for each hour. Such states would be generated by combining the day-ahead market clearing outcome with chosen TSO day-ahead decisions (if any) and stochastic models of day-ahead uncertainties. Particularly, stochastic models of load,



Models of look-ahead uncertainties
$\hat{P}_d$, $\hat{P}_w$, component planned outages
Monte-Carlo simulations
Look-ahead operation planning (Market clearing)
P_d, P_w, component unplanned outages
Plausible states
TSO decisions
Real-time operation
control
assessment
inputs
outputs
Database
Machine learning
Proxies of real-time reliability management

Figure 5.1: Methodology used to build the proxies. $\hat{P}_d$ and $\hat{P}_w$ are the forecast load and wind generation while P_d and P_w are the realizations.

wind generation and component forced outage (generators, lines, phase-shifting transformers, etc.) would be needed to capture the difference between the anticipated state of the system at the day-ahead and the realized state of the system in real-time.

Once a dataset of possible real-time states is generated, we apply to each state a simulator of RT reliability control followed by an assessment of its outcome. The purpose of RT control is to adapt to the uncertainty realization (different from the forecast due to the unavoidable forecast errors) to ensure that the RT reliability target is met. The outputs of RT control are in particular the RT control costs and decisions. Then the assessment stage computes a measure of the system reliability level.

5.2.2 Use of supervised machine learning

In order to predict continuous outputs, we compared the following regression algorithms:

- RF with 500 trees,
- ET, also with 500 trees,
- RR and Kernel Ridge Regression (KRR),
- SVR,
- fully connected NN with 3 hidden layers, 100 neurons per layer and ReLU as activation functions.

For the KRR and SVR algorithms, we tested three different kernels: linear, polynomial and gaussian. Note that the KRR algorithm with a linear kernel actually corresponds to the RR algorithm.

In order to train and then evaluate the accuracy of the resulting proxies, we divided the dataset into two subsets: the learning set which contains 80% of the samples and the test set which contains the remaining 20%. To assess the performance of a model, we used the R^2-score (coefficient of determination).

In order to find the best meta-parameters for each estimator, we defined a range of values for each meta-parameter and then determined with a 5-fold cross-validation over the learning set the cross-validation score for each combination. We kept the combination of meta-parameters leading to the best cross-validation score for each algorithm.

Note that for the RR, KRR, SVR and NN methods, since these algorithms are not scale-invariant, the inputs and outputs are standardized with the mean values and standard deviations of the learning set, such that they have a zero mean and a unit variance.

5.3 Case study on the IEEE-RTS96 benchmark

All the experiments are run on a Toshiba Satellite computer with the processor Intel®Core™ i7-3610QM @ 2.3GHz and 6GB of RAM. The assessment and the control problems are implemented respectively with MATLAB [nd] and with GAMS [2012]. Finally, the supervised learning library used in this work is Scikit-learn [Pedregosa et al., 2011].

We test our methodology on a modified version of the IEEE-RTS96 single area network [Grigg et al., 1999]. We add nine wind farms similar to those specified in [Pandzic et al., nd] with a capacity of 300MW. The one-line diagram of the system is depicted in Figure 5.2.

5.3.1 Database generation in a day-ahead context

We choose to generate our database in a day-ahead context. We consider one particular day for which we have load and wind generation forecasts and

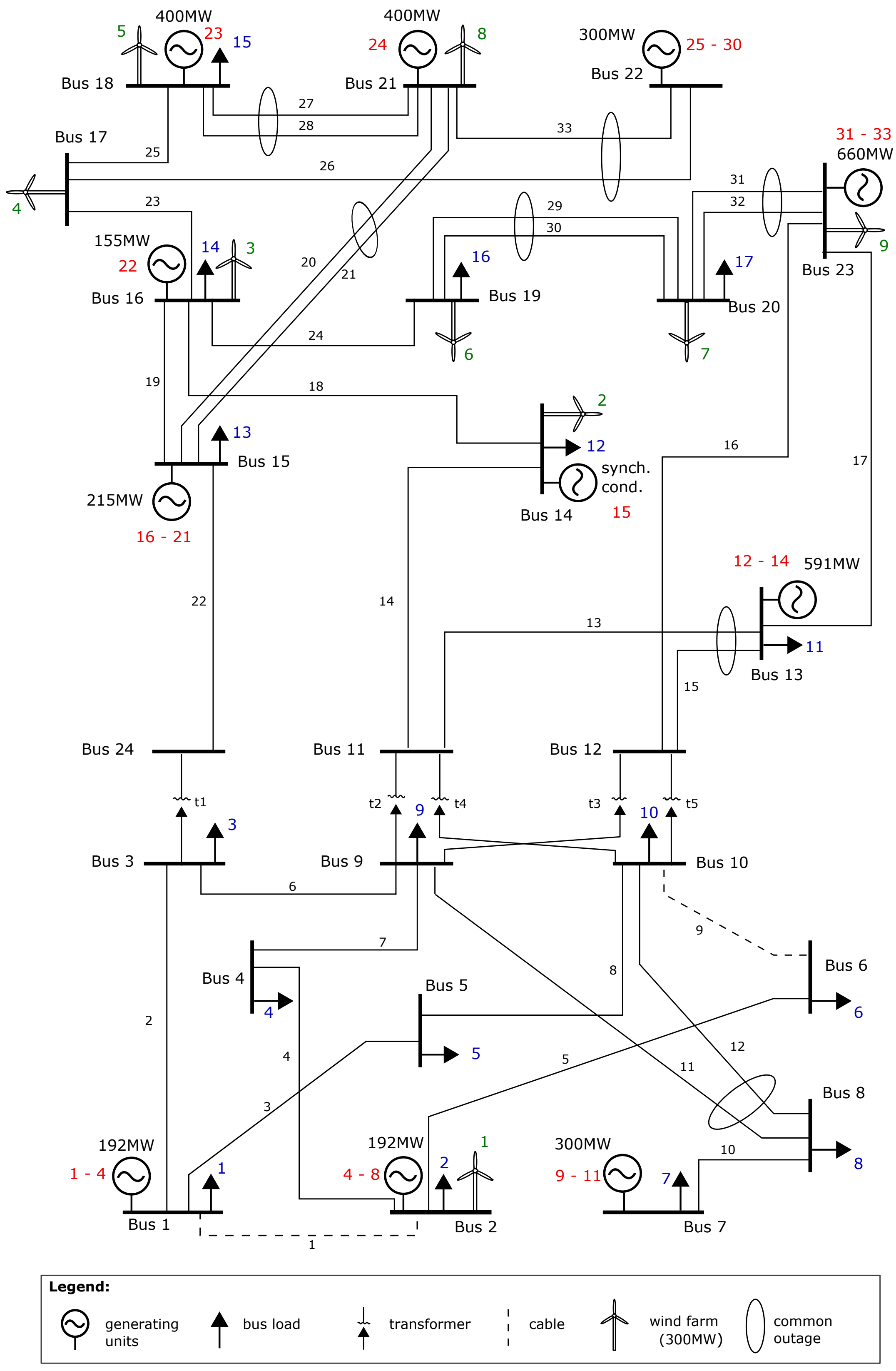


Figure 5.2: Modified version of the one-area IEEE-RTS96 network.

no planned outage. In order to generate multiple plausible scenarios for this day, we consider the realizations of load and wind generation based on day-ahead forecast error and also possible unplanned outages of transmission lines, transformers and generating units. More details on the uncertainty models used to generate plausible realizations can be found in appendix B.

The day-ahead market clearing is simulated with a UC problem imposing N-0 network constraints modeled with DC power flow equations and based on the forecast of load and wind generation. The upward and downward reserves have a capacity at least as large as the largest market output power among the generating units in service, in order for the system to be able to withstand the loss of any thermal generating unit.

For real-time control, we use the N-1 criterion but we only look at lines and transformers contingencies. We consider only preventive generation rescheduling and if needed preventive Wind Curtailment (WC) and Load Shedding (LSh). These preventive actions are determined by a preventive SCOPF and the DC network-model is used to express the network constraints. Note that for the sake of simplicity, each hour of the day is considered independently and therefore there is no temporal coupling between each hourly optimization problem. The set of constraints comprises maximum and minimum generation for each generator, ramp-up and ramp-down constraints, nodal power balance and transmission line thermal ratings. We allow continuous load shedding and wind curtailment. The objective function minimizes load shedding and wind curtailment costs, so that load shedding and wind curtailment are avoided as much as possible. The load shedding cost C^{p}_{LSh} is defined as:

$$C^{p}_{LSh} = \sum_{d \in \mathcal{D}} avg\_voll * LS_d, \tag{5.1}$$

where $\mathcal{D}$ is the set of loads, LS_d is the amount of load d shed and avg_voll is the average value of lost load (which estimates the cost of service interruptions) in €/MWh, here equal to the average of the coefficients published in [Fotuhi-Firuzabad and Billinton, 2000], assuming a 1 hour supply interruption duration. The wind curtailment cost C^{p}_{WC} is defined as :

$$C^{p}_{WC} = \sum_{w \in \mathcal{W}} wind\_penalty * WC_w, \tag{5.2}$$

where $\mathcal{W}$ is the set of wind farms, $wind\_penalty$ a wind penalty equal to 300€/MWh and WC_w is the amount of wind curtailed at wind farm w. Finally, the ReDispatch (RD) cost is defined as

$$C^{p}_{RD} = \sum_{g \in \mathcal{G}} C^{p}_{g}(P^{up}_{g} + P^{down}_{g}), \tag{5.3}$$

where $\mathcal{G}$ is the set of generating units, C^{p}_{g} is the redispatch cost of generating unit g per MW and P^{up}_{g} and P^{down}_{g} are respectively the amount of preventive ramp-up and ramp-down of g. Whenever the SCOPF program is infeasible for a sample (despite load shedding and wind curtailment), we simply discard it. Note that at the day-ahead stage, generating units 9 and 10 on bus 7 were forced to be on in order to avoid load shedding. Indeed, in case line 10 is in outage, if at least two of the generating units are not on, load 7 must be shed and since we only defined preventive actions, we prefer to prevent that by forcing these two units to be in service.

The assessment program is a more detailed cascade simulator based on [Yan et al., 2015b], allowing us to compute the expected amount of load shed over the trajectories of the system induced by a set of possible contingencies, which in addition to N-1 events also comprises some common mode double outages (circled in Figure 5.2). In particular, it computes the *risk*, which is defined as [Karangelos and Wehenkel, 2016]

$$\text{risk} = \sum_{c \in \mathcal{C}} \pi_c(w_0) \sum_{d \in \mathcal{D}} voll_d * LS_{c,d}, \tag{5.4}$$

where $\mathcal{C}$ is the set of considered contingencies, $\mathcal{D}$ is the set of loads, π_c is the probability of contingency c and depends on the weather w_0, $voll_d$[1] is the value of lost load for load d and $LS_{c,d}$ is the amount of load d shed at the end of the cascade of phenomena following contingency c. The values of lost load correspond to the coefficients published in [Fotuhi-Firuzabad and Billinton, 2000] and the method to compute the probabilities of contingencies is detailed in appendix B.

Our dataset contains 4000 samples. For each sample, it takes in average 0.04s to solve the real-time control problem and then 0.89s to solve the real-time assessment problem.

We study in particular 5 outputs: total cost of preventive actions, cost of preventive generation rescheduling C^p_{RD}, cost of preventive load shedding C^p_{LSh} and cost of preventive wind curtailment C^p_{WC} as outputs of the reliability control program and risk as output of the reliability assessment program. The total cost of preventive actions or total cost (also called the total preventive control cost in the next chapter) is defined as the sum of the costs of all preventive actions:

$$C^p_{tot} = C^p_{RD} + C^p_{LSh} + C^p_{WC}. \tag{5.5}$$

The considered input variables (also called 'features' in this manuscript) describe the realized states of the system. They include the realizations of nodal load and wind generations as well as the availability of each component of the system (generating units, lines and transformers), the total load, the total wind generation and the load net of wind (net load). Furthermore, to model that the outage probabilities of transmission lines depend on the weather conditions, we defined two weather states, namely normal and adverse [Billinton and Allan, 1996]. Therefore the weather status is part of the input variables. Concerning the TSO day-ahead decisions, the input variable per hour is the market dispatch of each generator, pre-computed in day-ahead mode. Finally, when we build the proxy predicting the system's risk, we consider in addition probabilities of contingencies. For the RR, KRR, SVR and NN methods the inputs and outputs have been standardized with the means and standard deviations of the learning set.

[1]Note that we use a *voll* depending on the load d because we want the risk to be a precise estimate of the level of reliability of the system. In contrast, we define the load shedding cost of the control program with an average *voll* to avoid discriminating by this factor in situations where load shedding is necessary.

5.3.2 Evaluation of the machine learnt proxies

Table 5.1 presents the test and learning scores of the different algorithms for each output. To determine them, the model is learnt with the learning set and then used to predict the test set $\mathcal{T}$ or the learning set $\mathcal{L}$. Recall that the score chosen is an R^2-score, for which the best possible value is 1.

Table 5.1: Test scores and learning scores for each estimator.

Algorithms	Set	Total cost	RD cost	LSh cost	WC cost	Risk
ET	$\mathcal{T}$	0.879	**0.944**	0.884	**0.883**	**0.754**
	$\mathcal{L}$	1	0.998	1	0.996	0.925
RF	$\mathcal{T}$	0.842	0.941	0.832	0.863	0.730
	$\mathcal{L}$	0.979	0.992	0.952	0.979	0.958
RR	$\mathcal{T}$	0.821	0.898	0.819	0.619	0.640
	$\mathcal{L}$	0.818	0.908	0.816	0.656	0.684
Gaussian KRR	$\mathcal{T}$	0.781	0.891	0.778	0.703	0.723
	$\mathcal{L}$	0.797	0.908	0.795	0.842	0.778
Poly KRR	$\mathcal{T}$	0.926	0.936	0.925	0.858	0.749
	$\mathcal{L}$	0.990	0.983	0.990	0.990	0.854
Linear SVR	$\mathcal{T}$	0.788	0.820	0.783	0.598	0.552
	$\mathcal{L}$	0.734	0.824	0.770	0.613	0.559
Gaussian SVR	$\mathcal{T}$	0.921	0.900	0.919	0.800	0.706
	$\mathcal{L}$	0.982	0.999	0.981	0.910	0.930
Poly SVR	$\mathcal{T}$	0.824	0.887	0.823	0.678	0.694
	$\mathcal{L}$	0.981	0.972	0.981	0.957	0.863
NN	$\mathcal{T}$	**0.964**	0.943	**0.962**	0.861	0.707
	$\mathcal{L}$	0.998	0.994	0.997	0.996	0.870

One can see in Table 5.1 that for the total cost and the load shedding cost the best method is NN. For the other outputs, ET are better. We also observe that the risk is the most difficult variable to predict. Finally, we observe that the two linear models (RR and linear SVR) are clearly outperformed by the non-linear models, especially for the prediction of the wind curtailment cost and the risk.

To give an idea about the quality of the prediction, Figure 5.3 shows the scatter plots representing the true values of the test set against the values predicted with the best proxies (in orange in Table 5.1). It can be seen that most data points seem to follow the line $y = \hat{y}$, showing that they are well predicted. It is clear in this figure that the predictions are not as good for the risk as for the other outputs.

The time needed to predict the risk with our proxies is 0.035ms per state in average, which is a great gain with respect to the 0.89s needed by the model

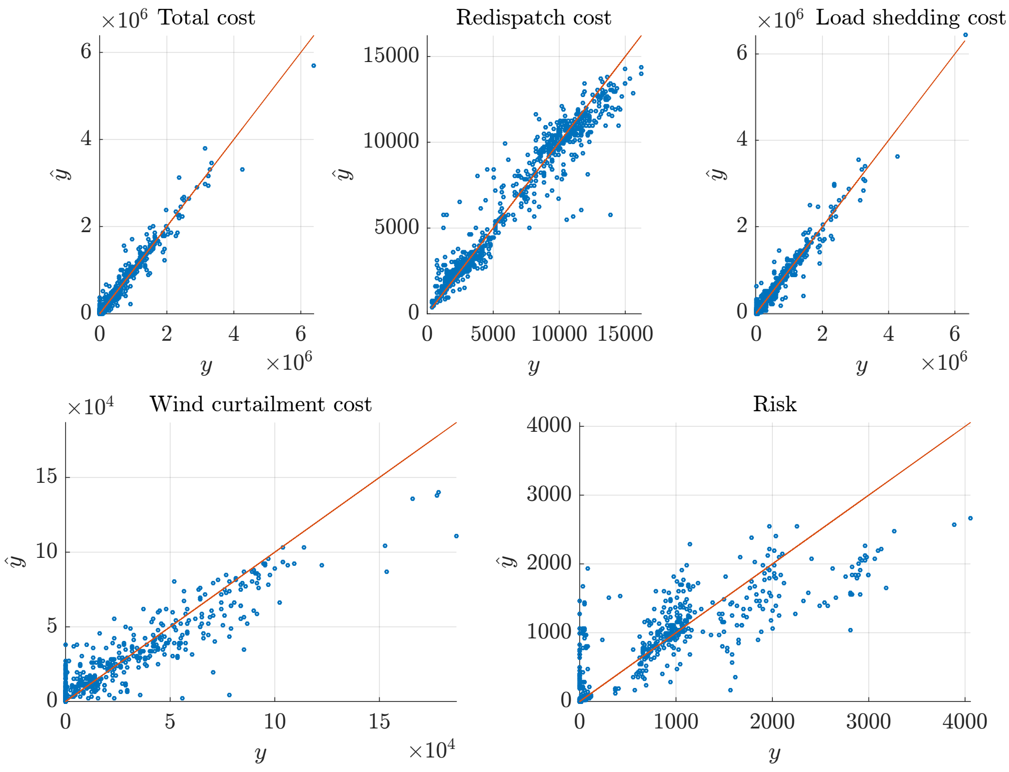


Figure 5.3: Scatter plots showing the true values (y) of the test set against the values predicted by the best proxies ($\hat{y}$) for each output.

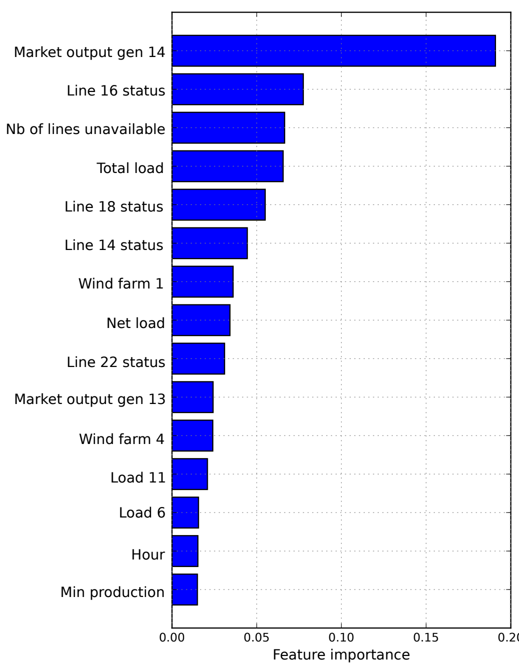
used to generate the dataset. Furthermore, it takes 0.04ms in average to predict each control output, while the SCOPF implementation used to generate the dataset required in average 0.04s per state. Therefore the gain in time achieved by our proxies is at least in the order of 10^3.

5.3.3 Study of the relevance of input features

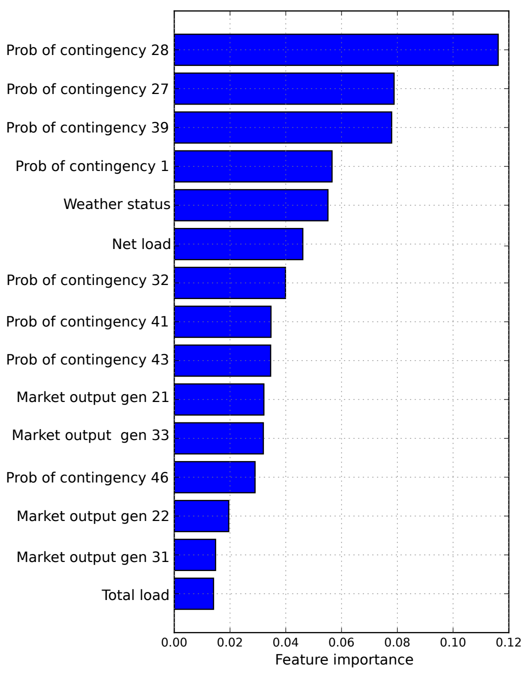
Supervised machine learning can also be used to find the most important input variables to predict an output, in order to have a better understanding of the problem. Input variable 'importances' can be computed as a by-product of training models in the form of random forests or extremely randomized trees.

To illustrate this kind of analysis, we show in Figure 5.4 the 15 most important variables and their relative importance computed by the ET method for two of the five studied outputs.

Let's first analyze Figure 5.4(a), concerning the total cost (the sum of the preventive redispatch, preventive load shedding, and preventive wind curtailment costs). We see that the most important input is the market dispatch of generator 14, which is on bus 13. If one looks at the total amount of up and down ramping per generator over the database, it can be noticed that generator 14 is one of the generating units for which the amount of redispatch is the



(a) Total cost



(b) Risk

Figure 5.4: Input variable importance ranking for (a) total cost and (b) risk level. Only the 15 most important features are shown in each plot. The horizontal axis measures the contribution (as a fraction $\in [0; 1]$) of the information provided by each input variable, normalized so that the total over all inputs is equal to 1.

largest. Furthermore, it is an oil/steam unit and one of the most expensive ones. This can explain the importance of this generator for the total cost. Note that similar remarks can be made for the market dispatch of generator 13, which also appears in the feature ranking. Indeed the total amount of redispatch of generator 13 is slightly smaller than the one of generator 14 and the unit is as expensive.

The second most important variable is the availability status of line 16. Its presence in this ranking indicates that this line is important for the preventive control. Indeed line 16 connects the area of the network where most generators are concentrated to the area where most loads can be found and thus the outage of this line may significantly increase the stress on the network and require more preventive actions. The same observations can be made concerning lines 14, 18 and 22, the availability statuses of which also appear in the 15 most important features.

The other important features are the number of lines in outage, the total and net load, some loads and wind farm generations and the minimum stable generation limit of the dispatchable units that are online. Load 11 is the load for which the total amount of load shed is the largest while wind farm 1 is the wind farm that is most often curtailed. These elements can explain why these variables are important. As could be expected, we can find in these 15 features elements impacting generation rescheduling, load shedding and wind curtailment.

If we now look at the most important features for the risk (Figure 5.4b), we see that there are many probabilities of contingencies. One could have expected that only the double outages would appear (contingency numbers greater than 39), given that single outages are covered by the N-1 criterion. However, it is not the case. One can even notice that the probability of having no outage (defined as probability of contingency 1) is the fourth most important feature to predict the risk. In fact all contingency probabilities carry information on the weather state and simultaneous component outages. Indeed all contingency probabilities change with the weather state and with the concurrent occurrence of outages. Therefore, one can deduce that the weather clearly impacts the risk, as well as the real-time topology. Note that the weather status is also among the 15 most important features.

Regarding the definition of risk (5.4), it is also not surprising to see that the total load is an important feature. The net load is an indication of the level of stress on the network and can therefore directly impact risk. Concerning the market dispatch of generators 21, 31 and 33, these generators are located on buses connected to other buses with double lines. Since the system is not operated to withstand the simultaneous loss of these double lines, when there is a common mode double outage coupled with a non-zero generation on the bus of concern, the risk can increase. The importance of the market dispatch of generator 22 can be explained similarly, given that most buses connected to bus 16 are linked to the rest of the system with a double line.

5.4 Related work

As we saw in chapter 4, our work is certainly not the first work on the use of machine learning in the context of power systems reliability management. As a matter of fact, a large body of work has been carried out in this context during the 1980'es and 1990'es, with the goal of building classifiers for fast assessment of transient stability and voltage stability (see [Wehenkel et al., 1989] for some early work, and [Wehenkel, 2012] for a more comprehensive bibliography). More recently, the European project iTesla [Vasconcelos et al., 2016] has developed an industrial software platform to apply these techniques for real-time security assessment. In these works, machine learning is used to build security rules in the form of decision trees or neural networks, to predict the dynamic response in real-time of the power system, on a per-contingency basis, so as to speed up real-time dynamic security assessment.

Furthermore, several papers have already studied the possibility to use machine learning in multi-stage decision-making programs to build proxies of shorter-term stages. For instance, we refer the reader to [Dalal et al., 2019] and [Canyasse et al., 2017], in which the authors have built proxies of respectively day-ahead unit commitment and real-time AC-OPF for a mid-term to long-term planning purpose. In [Dalal et al., 2019] the nearest neighbor algorithm is used to predict the costs and decisions of a day-ahead unit commitment program while in [Canyasse et al., 2017], several supervised learning algorithms are tested to predict the cost and feasibility of an AC-OPF problem.

In contrast, in the present work we propose to use machine learning to predict the response of the real-time reliability management process, so as to speed-up its simulation during the day-ahead reliability management process, and thus allow one to more effectively take into account uncertainties in this latter process. Hence, our models predict the outcome of the real-time reliability management process responding to day-ahead forecast errors, sudden changes in weather conditions, and/or forced network component outages in terms of real-time costs and risk. Furthermore, we decompose the total cost between redispatch cost, load shedding cost and wind curtailment cost in order to be able to analyze each of them separately. It will allow us to know for the predicted samples when load shedding and/or wind curtailment is necessary. This way, in a further step we will be able to select a day-ahead decision minimizing load shedding rather than the total cost for example, depending on the objective of the TSO.

5.5 Conclusions

In this chapter, we presented a methodology to build machine learnt proxies able to predict the outcome of real-time reliability management response in a short-term operation planning context. We tested several supervised learning algorithms and used them to predict in particular the reliability management costs and risk. Furthermore, we used the extremely randomized trees algorithm to rank the features and know which variables have more effect on the studied outputs. The purpose is to have better insight into the problem.

Applying the methodology to the IEEE-RTS 96 network in a day-ahead context, we showed that it is effectively possible to use supervised machine learning to build the proxies. The results are good with test scores close to 0.9 for most studied outputs. Furthermore the time gain is significant, with an order of magnitude of 10^3. The results given by the different supervised learning algorithms are close to each other but proxies learnt with extremely randomized trees, neural networks and ridge regression with a polynomial kernel are the most accurate ones.

Concerning the feature importance analysis, we noticed that probabilities of contingencies, weather conditions and net load are important to predict the risk while variables representing market generation of most re-dispatchable units, availability of important lines, loads that are most shed and wind farm generations that are most curtailed are ranked first to predict the total preventive cost.

However, there is still room for improvement. One possibility to improve the prediction is to increase the size of the learning set. Another possibility is to use feature selection to learn with only the most important features and especially remove non-relevant features. Finally, given that non-linear methods give the best results, it would be interesting to investigate the use of deep learning to predict the outcome of the real-time reliability management.

Possible future research directions are the improvement of the prediction for the studied outputs and especially for risk and the development of methods to predict the SCOPF feasibility of a particular sample as well as the real-time control decisions.

Another important future work, addressed in the next chapter in the context of reliability assessment in operation planning, is to import the machine learnt proxies in a suitable way into the look-ahead reliability management problems.

Finally, it is important to note at this point that the method we proposed can easily be extended to AC-SCOPF programs and more detailed contingency response simulators. It is in fact not limited to a particular choice in the modeling of the operator behavior and power system response. In the case of AC-SCOPF simulators, we can even expect the computing time gain of the proxies to be better compared to DC-SCOPF simulators, assuming that the computing time of the proxies will not change significantly, while the computing time of an AC-SCOPF is in practice much larger than that of a DC-SCOPF.

6

Probabilistic look-ahead reliability assessment with proxies

Overview

In this chapter, we address the problem of probabilistic reliability assessment in operation planning. We propose an approach combining Monte-Carlo simulation, variance reduction techniques such as control variates, and the machine learnt proxies of real-time operation built with the methodology presented in the previous chapter. The objective is to speed-up the Crude Monte-Carlo (CMC) approach, which would entail a very large number of heavy computations. We provide an extensive case study testing this approach on the three-area IEEE-RTS96 benchmark, in the context of day-ahead operation planning while using a SCOPF model to simulate real-time operation according to the N-1 criterion.

References: This chapter is an adapted version of the following publication:

Duchesne, L., Karangelos, E., and Wehenkel, L. (2018). Using machine learning to enable probabilistic reliability assessment in operation planning. In *2018 Power Systems Computation Conference (PSCC)*, pages 1–8. IEEE.

Terminology and notations have been slightly adjusted for the sake of consistency with the rest of this manuscript. The text has also been processed to minimize overlap with respect to previous chapters.

6.1 Introduction

In the context of operation planning, probabilistic reliability assessment essentially boils down to predicting, efficiently and with sufficient accuracy, various economic and reliability indicators reflecting the expected performance of the system over a certain look-ahead horizon, so as to guide the operation planner in his decision-making.

In this chapter we address the day-ahead operation planning problem as a 'template', and focus on reliability assessment while adopting a probabilistic approach. Specifically, we aim at evaluating over the 24 hours of the next day the expected costs associated to operating the system. Our approach relies on two fundamental components, namely a probabilistic model of the exogenous uncertainties and a computational model of real-time operation. The former describes the possible operating conditions that could be encountered the next day by the operator. The latter is a SCOPF model, formalizing the operator's choice of preventive and/or corrective controls in real-time and the resulting costs.

Using these two components, it is in principle possible to solve the operational planning reliability assessment problem by adopting a Crude Monte-Carlo (CMC) approach: sample a suitable number of scenarios of next-day operating conditions according to the probabilistic model, run the SCOPF model to gather for each hour of each scenario values of operating costs, average these quantities over the set of simulated scenarios to yield an estimate of the expected operating costs. An obvious drawback of this approach is however its computational burden, due to the very large number of SCOPF computations that would typically be necessary.

6.1.1 Proposal & experimental setup

To make this approach more tractable, we propose to work along two complementary directions, namely i) speeding up the individual computations by leveraging ML to replace the SCOPF computations by a much faster proxy of real-time operator response, and ii) instead of using the CMC approach, leveraging variance reduction techniques (more specifically, control variates approaches), so as to reach the same accuracy while relying on fewer SCOPF computations. The proposed approach works as follows:

- During a first stage, we sample a number of scenarios and solve them with the SCOPF model in order to compute values of the cost function. It yields a dataset of input-output pairs which is then exploited according to the machine learning methodology presented in chapter 5 in order to build *proxies* of real-time operation.

- During a second stage, we exploit the learnt proxies together with the *control variates approaches*, in order to estimate the expected values of the concerned cost components, while also exploiting a second (independent) sample of scenarios solved with the SCOPF model.

Comparing the accuracy obtained by the proposed approach to that of the CMC approach with an identical total budget of SCOPF computations, allows one to infer the potential computational speed-up of the proposed approach for a given target accuracy.

We test this approach on the three-area IEEE-RTS96, while modeling real-time operation according to the N-1 criterion, and in order to estimate the expected value of different components of the real-time operating cost, such as preventive control costs, and corrective control costs.

6.1.2 Chapter organization

The rest of the chapter is organized in four further sections: section 6.2 introduces the problem statement, the CMC and the control variates approaches, and describes the proposed methods, section 6.3 presents the case study, section 6.4 describes some related works, and section 6.5 concludes and presents some further research directions.

6.2 Problem statement, background, and methods

6.2.1 Problem statement

In day-ahead (da), the operation planner needs to assess how the operation of the system would turn out during the next day (nd), and if necessary takes some decision to ensure that real-time operation will turn out in a suitable way. While doing this, it faces uncertainties ξ_{nd} about the exogenous factors (renewable generation, weather conditions, demand, etc.) that will influence the outcome of reliability management during the next day, and needs to anticipate how the control-room operators will react to them in real-time.

In this paper, we model the behavior of the real-time control-room operators by a SCOPF model, aiming at meeting the N-1 criterion at the least cost.

Furthermore, in day-ahead conditions, we suppose that the operation planning engineer disposes of a generative model allowing to sample scenarios of next-day conditions $\{\xi^1_{nd}, \xi^2_{nd}, \dots\}$ and to plug them into a SCOPF computational module that will reveal the response of real-time operation based on the N-1 criterion[1]. We suppose that the planning engineer wants to evaluate the consequence of a given day-ahead decision, by estimating the resulting *mathematical expectation*, also known as the expected value, of real-time operation cost components.

We thus focus on the following computational problem: *given a day-ahead decision, a generative model of exogenous uncertainties over a horizon of* $T = 24$ *next-day hourly time-steps, and a software solving a sequence of SCOPF problems over a next-day scenario (or trajectory), how to minimize the number of SCOPF calls to obtain a sufficiently accurate estimate of the expected next-day operating costs.*

6.2.2 Background

6.2.2.1 *Crude Monte-Carlo approach*

The CMC approach [Rubinstein and Kroese, 2016] uses the generative model in order to sample next-day scenarios and then runs the SCOPF model on

[1]In our case study, these next-day scenarios are defined over a horizon of 24 hours via 24 time steps of one hour, and the SCOPF module is then actually applied for each such scenario in a sequential way over the 24 time steps, to reveal the outcome of real-time operation along such a scenario.

each one of them to compute the costs and constraint violation indicators. It averages such costs over a number n of scenarios until the accuracy is sufficient. Denoting by $y(\xi_{nd})$ an output of the SCOPF model of real-time operation[2] and by $\{\xi_{nd}^1, \xi_{nd}^2, \ldots, \xi_{nd}^n\}$ an *i.i.d* sample[3] of exogenous scenarios of next day conditions, the CMC approach estimates the mathematical expectation μ_y of y by

$$\hat{\mu}_y = \frac{1}{n} \sum_{i=1}^{n} y(\xi_{nd}^i), \tag{6.1}$$

and the standard deviation σ_y of y by

$$\hat{\sigma}_y = \sqrt{\frac{1}{n} \sum_{i=1}^{n} \left(y(\xi_{nd}^i) - \hat{\mu}_y \right)^2}. \tag{6.2}$$

It is well known that the CMC estimator $\hat{\mu}_y$ is an unbiased estimator of μ_y [Rubinstein and Kroese, 2016]. Further, the standard error of $\hat{\mu}_y$ depends on the value of σ_y and on the sample size in the following way

$$\sigma_{\hat{\mu}_y} = \sqrt{\frac{1}{n}} \sigma_y. \tag{6.3}$$

Thus, in the CMC approach, one classically determines the required number of scenarios by checking the ratio $\sqrt{\frac{1}{n}} \hat{\sigma}_y / \hat{\mu}_y$. Typically the number n of scenarios is determined so that this quantity is smaller than 1%. In our case study, the number of SCOPF computations is therefore equal to $T \times n$, with $T = 24$.

6.2.2.2 *Variance reduction by using the control variates approach*

The CMC approach requires the computation of $y(\xi_{nd})$ over a set of next day scenarios that is large if the variance of y is large. Each one of these computations implies solving a sequence of SCOPF problems over the 24 hourly time-steps of the next day along a realization ξ_{nd}^i of exogenous variables. The total number of required observations n to reach a required level of accuracy is thus larger if the uncertainties in ξ_{nd} induce higher variabilities of costs and other indicators that need to be computed. In order to reduce the computational burden, various approaches have been proposed in the literature to reduce the variance of Monte-Carlo methods. The one that we investigate in this work is called the ‘control variates’ approach and is based on the following rationale.

Suppose that we dispose of a proxy y_p allowing us to compute at a very cheap computational cost an approximation $y_p(\xi_{nd})$ of the function $y(\xi_{nd})$. Let us suppose that $y_p(\xi_{nd})$ is so cheap to compute that we can estimate its mean μ_{y_p} by crude Monte-Carlo, using a very large number of observations at

[2]In our case, this value y would be obtained as the sum of hourly values of some term of the SCOPF objective function computed for the 24 successive hourly time-steps composing a scenario ξ_{nd}.

[3]Note that it is not required that the successive *time-steps* in a given scenario are *i.i.d.* Rather the n different *scenarios* are assumed i.i.d.

negligible cost. Since $\mu_y = \mu_{y_p} + \mu_{y-y_p}$, we can thus reframe the estimation of μ_y by estimating separately μ_{y_p} and μ_{y-y_p}. If at the same time $x(\xi_{nd}) = y(\xi_{nd}) - y_p(\xi_{nd})$ has a small variance, we then can estimate $\mu_x = \mu_{y-y_p}$ by using the CMC approach, at sufficient accuracy, with a number of observations that will be relatively small (in the extreme case, $x(\xi_{nd})$ is almost constant, and its expectation could be estimated by using a very small number of observations).

Thus, the control variates approach consists of seeking a good proxy y_p ('good' meaning that σ_x is much smaller than σ_y) so that using the estimate:

$$\hat{\mu}_y^{y_p} = \mu_{y_p} + \frac{1}{m}\sum_{i=1}^{m} x(\xi^i_{nd}) \tag{6.4}$$

would need a much smaller number m of observations than the CMC approach applied directly to y, while obtaining the same level of accuracy.

Indeed, the control variates estimator is by construction also an unbiased estimator of μ_y, since $\mu_x = \mu_y - \mu_{y_p}$ and since the second term of eq. (6.4) is itself an unbiased estimator of μ_x. On the other hand its standard error is given by

$$\sigma_{\hat{\mu}_y^{y_p}} = \sqrt{\frac{1}{m}}\sigma_x, \tag{6.5}$$

so that for a same level of accuracy than the crude Monte-Carlo approach this method would need a sample size m equal to

$$m = n\frac{\sigma_x^2}{\sigma_y^2}, \tag{6.6}$$

which will be (much) smaller than n if σ_x^2 is (much) smaller than σ_y^2. Notice that we do not suppose in any case that $\mu_{y_p} = \mu_y$; indeed if this could be guaranteed we could say in advance that $\mu_x = 0$, so that we would not need any additional observations to estimate this quantity.

6.2.3 Proposed methods

6.2.3.1 *Building control variates by supervised machine learning*

In some cases, a suitable proxy of the quantity y to estimate can be hand-crafted. In our context, it is unlikely that this can be done, given the complexity of the relationship between next day conditions ξ_{nd} and the considered variables y (cost indicators reporting the outcome of real-time operation).

We thus propose to use the methodology presented in chapter 5 in order to build, from a sample of pairs $(\xi^i_{nd}, y^i = y(\xi^i_{nd}))^{i=1,\dots,k}$, a proxy y_p, and then use the inferred proxy within the control variates approach. Given a total budget n of next-day possible scenarios for which we can compute the exact value of y, we investigate how to split them in two parts in the best way, the first k being used to learn a proxy, and the remaining $m = n - k$ being used in the control variates approach. We will also consider different settings for applying the machine learning approach, so as to build for a given learning sample size k the most accurate proxy y_p of y.

6.2.3.2 *Stacked Monte-Carlo approach*

In the Stacked Monte-Carlo (SMC) approach [Tracey and Wolpert, 2016] a sample of pairs $(\xi^i_{nd}, y^i)^{i=1,\ldots,n}$ is used in a more intensive way to build an estimate of μ_y. Similarly to the cross-validation method described in chapter 3, the sample is first split into V folds of $k_v \approx \frac{n}{V}$ pairs, then for each fold $v \in \{1, \ldots, V\}$ a proxy y^v_p is built by using machine learning applied to the union of all other $V-1$ folds, and then used to predict the value $x^v = y - y^v_p$ over the held-out fold only. Also, for each fold, the value of $\mu_{y^v_p}$ is estimated separately with high accuracy. The final SMC estimate is computed as follows

$$\hat{\mu}^{SMC}_y = \frac{1}{V} \sum_{v=1}^{V} \left(\mu_{y^v_p} + \frac{1}{k_v} \sum_{j=1}^{k_v} x^v(\xi^{v_j}) \right) , \tag{6.7}$$

where v^j denotes the index of the j-th sample of fold v.

Rather than splitting the whole sample once in two folds, one for training a proxy and the rest for use in the control variates approach, this approach uses all available observations both for training and for the control variates estimate, while however avoiding to use any observation both in a certain training sample and in the corresponding control variates estimate. It is therefore likely to lead to an even better variance reduction, while still being an unbiased estimator of μ_y.

6.2.4 Overall proposed study approach

In order to study the effectiveness of the above approaches, we first evaluate the baseline, i.e. the CMC approach, in terms of the number n of observations required to yield a reasonable target precision, based on empirical simulations with a given test system and target quantity y to be evaluated in day-ahead conditions.

Next, we investigate how to exploit in the best way the set of n observations by studying different settings of the approaches proposed: i.e. different ways of splitting into k and $n-k$ observations, different machine learning methods used to build the proxies, and different ways of stating these machine learning problems. The goal of this study is to determine the gain in accuracy that can be obtained with the proposed approaches with respect to the CMC approach, while using a same budget for the detailed simulations (and SCOPF computations) of day-ahead conditions.

Finally, we translate the gain in accuracy in gain in computational requirements, by determining how many observations n' would be needed to obtain the same level of accuracy than our approaches, while using the CMC approach.

6.3 Case study

6.3.1 Test system, uncertainties, and real-time operation model

We explore the applicability of this approach on the 3-area version of the IEEE RTS-96 [Grigg et al., 1999], as modernized with the addition of 19 wind power generators by Pandzic et al. [nd]. Our studies refer to the 1st day of the year, for a peak demand of 3135 MW, per area and 'favorable' wind. The line thermal ratings are reduced to 80% of their value and the capacities of the wind farms are those suggested in [Pandzic et al., nd].

6.3.1.1 *Horizon, uncertainties & temporal resolution*

We place ourselves 12 hours before the start of the day under consideration (that is, at noon of the previous day) and assume that, at this point in time, the yet unresolved uncertainties restrict to the forecast errors of wind power injections and load demand. Contrarily to the previous chapter, we neglect the uncertainties related to component unexpected outages for simplicity. Our modeling approach for exogenous uncertainties is based on the uncertainty models detailed in appendix B. We consider the spatial correlation between the forecast errors concerning power injections/demands located in the same area of the 3-area system. To do so, we assume that the forecast error of each power injection/demand is composed by a global and a local term. The global term is common for all wind power generators/loads in the same area while the local term is distinctive per each individual power injection/demand. Adopting a one-hour interval as the time step for real-time operation, we exploit this model to generate via Monte-Carlo simulations scenarios of 24 forecast error realizations per load demand and wind power generator.

6.3.1.2 *Day-ahead planning model*

In order to simulate day-ahead decision making, we consider the commitment status and economic dispatch of all dispatchable generators as well as the provisional curtailment of wind power generation, as decision variables. To fix such decisions, we solve a multi-period security-constrained UC problem in anticipation of the demand values from the original system description [Grigg et al., 1999] and the wind power 'favorable' forecast from [Pandzic et al., nd]. More specifically, we use the DC power flow approximation while taking into account the N-1 criterion concerning all transmission system components (i.e., transmission lines, cables, and transformers). Further, as a preventive measure to address the wind/load uncertainties we impose 300 MW up-ward and down-ward spinning reserve capacity constraints. Compared to the UC formulation of the previous chapter, we ensure that our day-ahead decisions satisfy the N-1 criterion for the forecast scenario, to reduce the need of generation redispatch in real-time and ensure coherence between the day-ahead and real-time contexts, and we consider the provisional curtailment of wind power generation as an additional decision variable to help satisfying this criterion. The full mathematical formulation is available in appendix A.2.

6.3.1.3 *Real-time operation model*

To model the system trajectory within the day under consideration, we sequentially go through the 24 single period real-time operation instances. That is, for any such instance, we i) generate a set of wind power & load demand realizations, and, ii) re-compute real time preventive and/or corrective (alternatively, pre- and/or post-contingency) control actions to maintain conformity with the N-1 criterion[4]. These decisions should adjust to the most recent forecast error realizations, and are taken with a single-hour horizon. For the sake of simplicity, the real-time control decisions already applied within the trajectory do not constrain the candidate decision space. It is only constrained by day-ahead decisions. Nevertheless, the existence of control actions to achieve the N-1 criterion is certainly not guaranteed at this stage. Rather, the decision-maker would attempt a 'best-effort' approach avoiding to the extent possible load shedding and wind curtailment to maintain the system operational under any postulated contingency within the considered set of contingencies. We again resort to a DC-SCOPF to model such a 'best-effort' approach. More specifically, we consider the preventive and/or corrective redispatch of each generating unit as the available actions of first priority and, in our objective, take into account the respective marginal redispatch costs. Further, we model pre- and post-contingency load shedding and wind curtailment and treat these actions as options of last resort by means of appropriately high penalty cost coefficients in the objective function of the resulting optimization problem. Compared to the previous chapter, the preventive redispatch costs are also minimized in the objective function and the real-time operation model is improved by modeling corrective control actions. The detailed mathematical formulation of this problem, notably including ramping constraints between any preventive redispatch action and the day-ahead dispatch for the forthcoming period for every generating unit is available to the reader in appendix A.3.

6.3.1.4 *Operational cost assessment*

Finally, given the decisions for the solution of this real-time SCOPF problem, we evaluate the respective hourly costs of operation. At this stage we adopt a penalty for wind-curtailment of 300€/MWh for any wind power generator and, consider an identical value of lost load for each load demand[5], which is an average of the coefficients published in [Fotuhi-Firuzabad and Billinton, 2000], assuming a 1 hour supply interruption duration. Using this process, we build a database of 2400 trajectories. Figure 6.1 schematically summarizes the database generation.

[4]Notice that, at this stage we exclude from consideration the outages of the single lines feeding nodes 207 and 307; such outages would lead to the islanding of these nodes.

[5]In this study, we are only interested in the amount of load shedding and so we do not want to discriminate by the value of lost load in case load shedding is necessary.

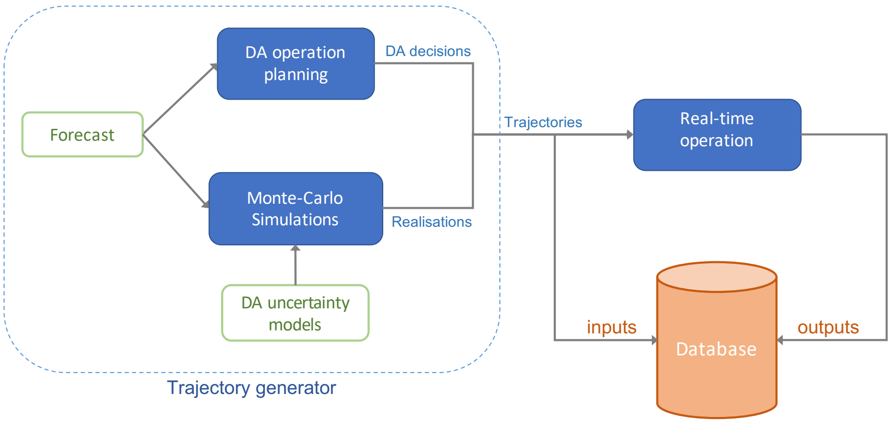


Figure 6.1: Methodology applied to the database used to learn the proxies y_p.

6.3.2 Machine learning settings and predictors

Once the database is generated, we can use it to learn the proxies. We study two classes of regression predictors: ET and NN, as they were identified as the most accurate proxies in chapter 5. Both ET and NN are two well-known non-parametric methods to solve regression problems and are both non-linear. Furthermore they have complementary characteristics. A value predicted by the NN predictor is not bounded by the values seen in the training database, contrarily to the ET predictor. On the other hand, the ET predictor is smoother. Furthermore, with a random forest algorithm such as ET, we can exploit the feature importances to have a better understanding of the system studied.

6.3.2.1 *Designing the predictors*

In order to build the predictors, we divide our dataset into two sets: a learning and a test set. The proxies are built with the learning set and then used to predict the target output over the test set.

Both methods depend on meta-parameters that can be tuned to improve the performance of the predictors. For the ET algorithm, we tested the following parameters: $k = 1, p/3, p/2, p$, where p is the total number of features and $n_{min} = 2, 4, 6, 8, 10, 20$. The number of trees was set to 1000, which is a good trade-off between performance and time needed to train and predict. The NN used in this chapter is a multi-layer perceptron with ReLU activation functions. We tried the following configurations: two or three hidden layers with 10, 50 or 100 neurons per layer. In order to find the best meta-parameters but still avoid overfitting, we use 5-fold cross-validation, and the R^2-score to assess the performances. The best meta-parameters vary in function of the output we want to predict or the setting used.

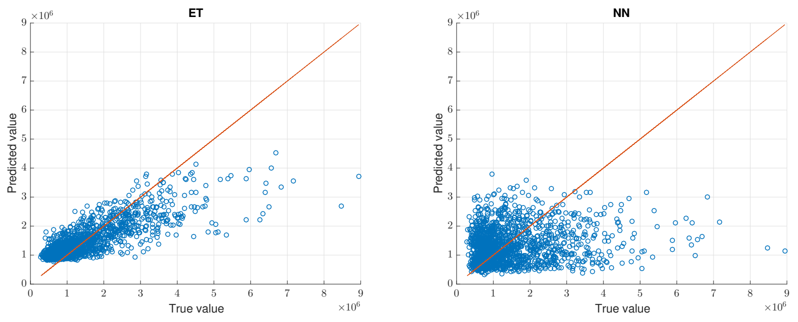
6.3.2.2 *Choice of the setting for machine learning*

In order to learn the proxies, we have investigated two different settings of applying machine learning to a dataset of trajectories solved with the SCOPF model.

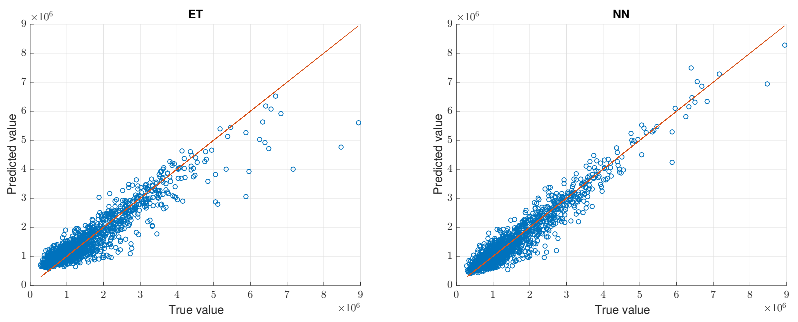
Setting 1 was to use a dataset of (say k) trajectories, each one described by a set of features describing the uncertainty realizations for each of the 24 hours (as well as the generation schedule decided the day ahead), and as output the sum of hourly costs along that trajectory.

Setting 2, on the other hand, consisted of splitting each trajectory into 24 hourly snapshots, inferring on the basis of these $k \times 24$ hours a proxy of the hourly costs, and predicting then for a certain trajectory the total cost by the sum of its 24 hourly proxy predictions.

In a preliminary study, we applied these two approaches with both the ET and NN predictors, different cost terms and different training sample sizes k. By comparing the performances of these two settings on independent test samples, in terms of the overall accuracy of their predictions, we made two observations, namely i.) Setting 2 very clearly outperforms Setting 1[6], in all cases, and ii.) the NN predictor is often significantly more accurate than the ET predictor, but it needs a more careful tuning of its meta-parameters. As an example, Figure 6.2 shows four scatter plots comparing one of the terms of the real-time operating costs (the total preventive control cost, in euro) against its predicted value over the test set for both settings and learning algorithms in the case of $k = 850$ learning trajectories (each one of 24 hours).



(a) Setting 1



(b) Setting 2

Figure 6.2: Scatter plots showing true vs predicted values for the total preventive control cost over the test set, in case of (a) direct prediction of the trajectory cost and (b) sum of the 24 hourly proxy predictions. $k = 850$.

[6] Note that this observation may not hold for other modeling choices of real-time operation, for instance if temporal coupling is considered between hourly instances of a trajectory.

Hence in the rest of this chapter we only report our results obtained with Setting 2, and we provide comparisons of the ET and NN predictors, when of interest.

6.3.3 Results

In our study we have investigated different terms of the real-time operating costs, namely total preventive control cost (*i.e.* the sum of preventive generation redispatch, load shedding and wind curtailment costs), expected corrective control cost (also composed, per contingency, of generation redispatch, load shedding and wind curtailment cost sub-components), preventive load shedding and wind curtailment costs.

6.3.3.1 *Comparison of the approaches*

To compare the different approaches, we first apply them to estimate the 'Total preventive control cost' (C^p_{tot}), defined in equation (5.5).

Computational budget To start, Figure 6.3 shows the convergence of the CMC approach when applied to the estimation of the expected value $\mu_{C^p_{tot}}$ of the total preventive control cost, as a function of the sample size n. With a total number $n = 2400$ of sampled scenarios, the standard error of this CMC approach is slightly above 2×10^4, i.e. about 1.4% of the estimated value of $\mu_{C^p_{tot}} \approx 1.44 \times 10^6$.

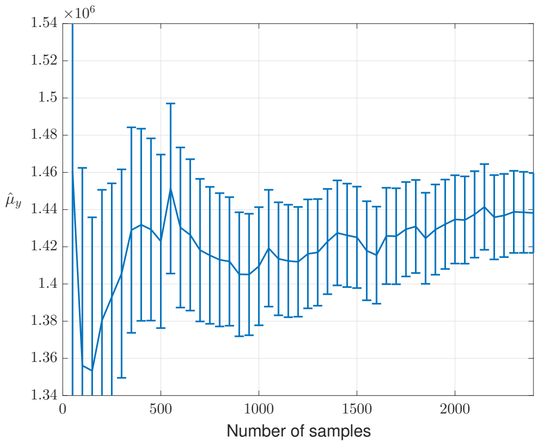


Figure 6.3: Convergence of the CMC approach for C^p_{tot} with a growing number n (up to 2400) of trajectories (error bars show the $\pm\hat{\sigma}/\sqrt{n}$ interval, where $\hat{\sigma}$ is the sample estimate of the standard deviation of C^p_{tot}).

Next, we apply the control variates approach while using the first 850 scenarios to build our proxies (with Setting 2), and both ET and NN predictors. Figure 6.4 shows that in both cases this approach leads to a reduction of the standard error of about a factor 2, the NN predictor being more accurate than the ET predictor. With a total number of trajectories of $n = 2400$ ($k = 850, m = 1550$), the standard errors of the control variates approaches are smaller than 1×10^4 (0.7% in relative terms).

Thus, while all estimators seem to converge to the same final value, with the CMC approach, we would need about 10,000 trajectories (i.e. about 240,000 SCOPF computations at the hourly basis) to reach the same level of accuracy than the control variates approaches using only about 60,000 SCOPF computations.

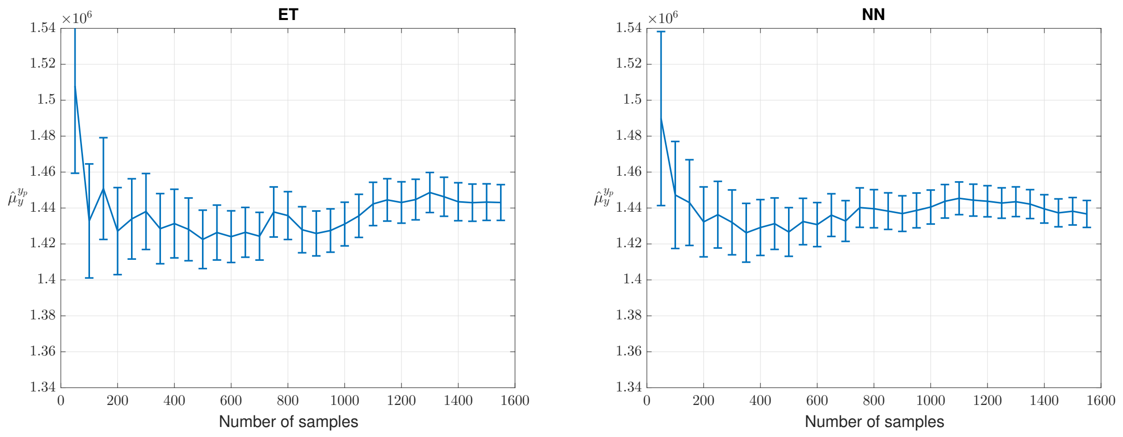


Figure 6.4: Convergence of the control variate approach applied to C_{tot}^{p} for a learning set size of $k = 850$, and up to $n - k = 1550$ additional observations.

In order to construct the above estimators, we have used the CMC approach to estimate μ_{y_p} of the proxy predictions on a sample of 20,000 trajectories (without any SCOPF computation, of course). Figure 6.5 shows, for both predictors, how this side computation converges with the number of (side) observations. We also see from this figure that using these proxies alone, even with a much larger number of additional side scenarios (say about 10,000), would lead to an estimator that would be quite biased while also having a higher variance than the two estimators using the control variates approach. Indeed, the NN proxy underestimates the actual value of the expected total preventive control cost around 1.42×10^6 and with the ET method we obtain a quite similar curve, with the main difference that in this case the bias is positive and even a bit larger (with a mean μ_{y_p} converging towards 1.47×10^6).

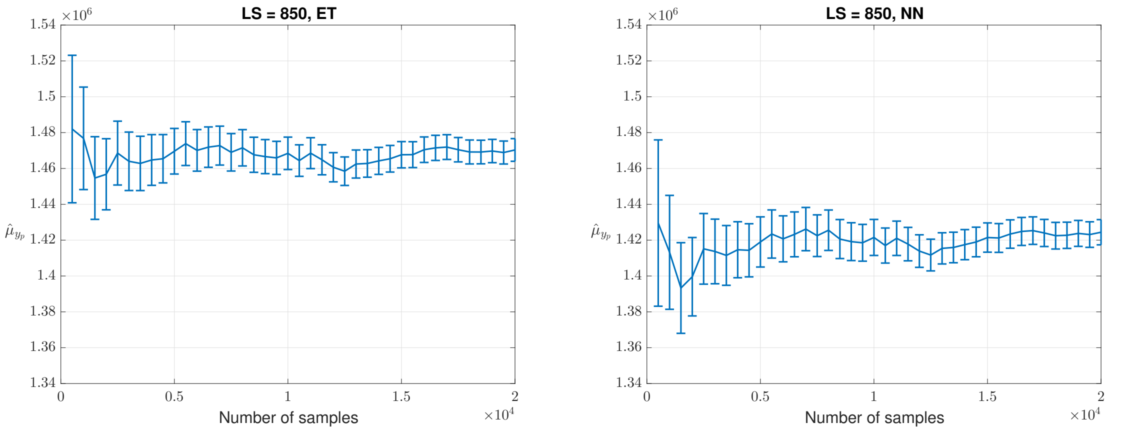


Figure 6.5: Estimation by CMC of $\hat{\mu}_{y_p}$ for both ET and NN predictors, from side observations (here the proxies were built using 850 training trajectories).

From these results, we observe that for a same budget of 2400 scenarios solved with the SCOPF computations, we can significantly improve the CMC approach by using part of the sample to build a control variate by machine learning, and the rest to reduce the bias of these proxies in a systematic way.

Size of the learning sample Of course, the budget of trajectories could possibly be used in different ways. To investigate this aspect, we have made further analyses. Table 6.1 reports numerical results. The first line gives the average and its standard error as obtained with the CMC approach for a total budget of 2400 scenarios solved with the SCOPF model, each one with 24 hourly time steps; the next lines show the results of the proposed approach, for different ways of splitting this budget into learning sample (k), the $2400 - k$ being used for the control variates approach. For each value of k and for both ET and NN predictors, we indicate the value of their mean ($\hat{\mu}_{y_p}$) estimated from 20,000 side observations (and its standard error), and the mean and standard error of using it in the control variates approach. We observe from the values in this table that, in spite of the biased values of $\hat{\mu}_{y_p}$ and for all settings, the control variates approach is yielding a non biased estimator with a factor two improvement of the standard error with respect to the CMC approach. We also see that the most accurate setting seems to be the NN approach with $k = 750$ (with a standard error of about 8×10^3), but the variation of accuracy with k and the predictor used is actually not very strong, the worst setting (ET with $k = 1000$) and the best one (NN with $k = 750$) yielding a difference in standard error of less than 20% (both are highlighted in orange in the table).

Table 6.1: Accuracies of different settings of the control variates approach to estimate the expected value of C^p_{tot}

Method	Mean	Std Err
CMC - $n = 2400$	1.438e+06	21.45e+03
$\hat{\mu}_{yp}$(ET) - $k = 250$	1.524e+06	5.87e+03
$\hat{\mu}_{yp}$(NN) - $k = 250$	1.416e+06	7.14e+03
MC with control variate (ET) - $k = 250$	1.440e+06	9.57e+03
MC with control variate (NN) - $k = 250$	1.436e+06	9.03e+03
$\hat{\mu}_{yp}$(ET) - $k = 500$	1.499e+06	6.23e+03
$\hat{\mu}_{yp}$(NN) - $k = 500$	1.426e+06	7.13e+03
MC with control variate (ET) - $k = 500$	1.437e+06	8.93e+03
MC with control variate (NN) - $k = 500$	1.431e+06	8.97e+03
$\hat{\mu}_{yp}$(ET) - $k = 750$	1.478e+06	6.33e+03
$\hat{\mu}_{yp}$(NN) -$k = 750$	1.428e+06	7.09e+03
MC with control variate (ET) - $k = 750$	1.440e+06	9.68e+03
MC with control variate (NN) - $k = 750$	1.434e+06	**8.07e+03**
$\hat{\mu}_{yp}$(ET)- $k = 1000$	1.474e+06	6.41e+03
$\hat{\mu}_{yp}$(NN) -$k = 1000$	1.419e+06	7.01e+03
MC with control variate (ET) - $k = 1000$	1.441e+06	**9.85e+03**
MC with control variate (NN) - $k = 1000$	1.431e+06	8.19e+03

The Stacked Monte-Carlo approach In order to conclude our analysis of variance reduction methods, let us report some preliminary experiments using the SMC approach. The results of Figure 6.6 were obtained in the following way: for a given number of trajectories $n \in [250; 2400]$, we used the SMC method with the NN predictor and $V = 10$ folds. We can observe from this graph that with a budget of only $n = 1000$ trajectories, we already obtain a non biased estimate of $\mu_{C^p_{tot}}$ with a standard error comparable to those obtained with the above control variate approaches when they were exploiting a budget of more than 2000 trajectories.

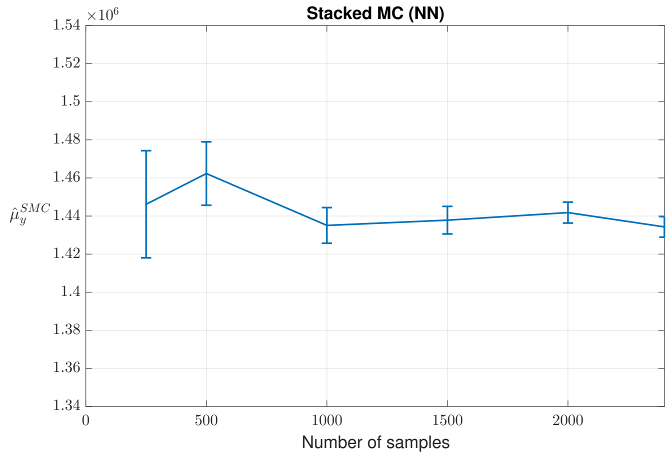


Figure 6.6: Convergence of the expected value of C^p_{tot} by the SMC method, with 10 folds and NN as learning algorithm

6.3.3.2 *Application to the estimation of other terms of the cost function*

For the sake of completeness we also briefly report here on results obtained from the application of these approaches to the sub-components of the real-time cost function. More specifically, we present results on estimating the cost of preventive load shedding, preventive wind curtailment as well as expected corrective cost (that is, sum of expected corrective generation redispatch, load shedding and wind curtailment) respectively.

For each one of these sub-components, Tables 6.2 – 6.4 present the performance of the CMC approach (first row), with the control variate (third row) while using 850 trajectories to construct proxies via the NN predictor, which as reported earlier was found to outperform the ET predictor. The middle row per sub-component presents the mean value and standard error of the NN predictor, once again estimated using 20,000 side-observations.

Tables 6.2 – 6.4 verify in general the applicability of the proposed approach in estimating the various sub-components of the real-time cost function. It is of interest to comment here that such sub-components are also indicative of the type of problems to be anticipated in real-time operation. For instance, increased preventive load shedding & wind curtailment costs are indicative of the fact that the network may be inadequate to securely accommodate the range of potential wind power and demand injections, that is a lack in transmission

Table 6.2: Accuracies of estimating the expected value of preventive load shedding cost – C^p_{LSh}

Method	Mean	Std Err
CMC - $n = 2400$	8.6157e+05	2.1374e+04
$\hat{\mu}_{y_p}$(NN) - $k = 850$	8.7661e+05	7.1728e+03
MC with control variate (NN) - $k = 850$	8.5968e+05	7.6484e+03

Table 6.3: Accuracies of estimating the expected value of preventive wind curtailment cost – C^p_{WC}

Method	Mean	Std Err
CMC - $n = 2400$	4.5849e+05	2.9559e+03
$\hat{\mu}_{y_p}$(NN) - $k = 850$	4.5644e+05	1.0053e+03
MC with control variate (NN) - $k = 850$	4.5714e+05	1.1766e+03

Table 6.4: Accuracies of estimating the expected value of expected total corrective control cost – $\hat{C}^c_{tot}$

Method	Mean	Std Err
CMC - $n = 2400$	2.8057e+03	4.3067
$\hat{\mu}_{y_p}$(NN) - $k = 850$	2.8037e+03	1.4425
MC with control variate (NN) - $k = 850$	2.8158e+03	2.9275

capacity and operational flexibility. These indicators can thus be exploited to point towards the type of necessary operational planning decisions. Likewise, increased expected corrective control costs may indicate that exogenous factors (e.g., weather conditions) and/or the system loading conditions increase the likelihood and/or potential impact of contingencies. From a planning perspective such finding can be exploited, for instance, by considering to make more preventive real-time flexibility resources available in order to reduce reliance on corrective control, taking into account that in practice it may turn out not to perform as expected.

Last but not least, these preliminary results on the sub-components establish the potential for further investigation on estimating the broad range of indicators that can be relevant to describe the operability of the system in real-time, such as load shedding and wind curtailment costs referring to specific loads and wind generators of interest, the achievability of the considered reliability criterion, etc.

6.4 Related works and contribution

In this section, we position our contribution with respect to the literature of related works.

On the one hand, we refer the reader to the text-book [Rubinstein and Kroese, 2016] for an up to date introduction to Monte-Carlo methods, and the huge body of variance reduction techniques that have been proposed over the years in this context. We also refer to [Hastie et al., 2009] for an explanation of the prominent role of the bias-variance tradeoff in the design of modern machine learning algorithms. In direct relation to the methods developed in this paper, we refer to [Oates et al., 2017; Tracey and Wolpert, 2016] which are studying from a theoretical point of view the use of machine learning to build control variates for the Monte-Carlo approach. In particular, Oates et al. [2017] show that by using a suitable class of non-parametric regression methods (such as the ET and NN predictors used in our work), convergence faster than 'root-n' (i.e. the convergence of classical Monte-Carlo methods $\mathcal{O}(n^{-1/2})$) may be achieved for a very large class of complex Monte-Carlo integration problems.

The more specific idea of using machine learning to build proxies of shorter-term decision-making contexts to be used when solving longer-term reliability assessment problems has been proposed and studied only recently, as we saw in chapters 4 and 5. Within this context, the method presented in this paper, using machine learning to build control variates to speed up the Monte-Carlo approach, is to our best knowledge entirely novel.

6.5 Conclusion and further research

In this chapter we have explored the use of machine learning in order to speed up the Monte-Carlo simulation approach in the context of uncertainty aware reliability assessment in operation planning.

The Monte-Carlo simulation approach has two nice features in this context: i) it lends itself almost trivially to massive parallel computing architectures, ii) it is free of strong assumptions about the uncertainty and real-time operation models and hence very generally applicable. The crude Monte-Carlo approach is nevertheless highly compute-intensive, and in order to scale it to real-life use, speeding it up is therefore highly desirable.

The approach investigated in this paper has solid theoretical guarantees: it is unbiased and may yield faster than 'root-n' convergence. On the basis of a systematic case study on the three-area RTS96 benchmark, we found that when estimating the expected value of various costs terms of next-day real-time operation it allows one to reduce the standard error of the crude Monte-Carlo approach by a factor of about 3 to 4, while using the same number of sampled trajectories of next-day operation conditions. In computational terms, this means a speed-up of a factor 9 to 16, for a given target accuracy.

Another advantage of the approach is that the scope of the proposed idea is not limited to the day-ahead operation planning context, with the specific assumptions used in the presented case study. It can also be applied to other modeling choices of uncertainties, operator behavior and power system response, and to other reliability management contexts. In fact, this method of combining the control variates approach with a machine learnt proxy could be applied in all power systems applications for which Monte-Carlo simulations are leveraged.

Furthermore, it is not necessary for the proxy y_p to perfectly predict the quantity y but only to be such that the variance of $x = y - y_p$ is (sufficiently) smaller than the variance of y. As a consequence, a proxy y_p built for a specific day and a specific look-ahead decision could be exploited during several days, and for other look-ahead decisions, as long as $\sigma_x < \sigma_y$. This can significantly reduce the computational burden linked to learning a new proxy each day for each candidate day-ahead decision to be assessed in day-ahead operation planning.

There are many possible directions of future research to broaden the potential of the proposed approach. Among them we mention the following ones:

- How to adapt the approach in order to estimate the probabilities of extreme situations that could occur in the next day, e.g. leading to the inoperability of the system?

- How to adapt the approach to estimate the gradient of the expected value of the different cost terms or of extreme situations' probabilities with respect to the calibration of day-ahead decisions, in order to provide further hints for optimal day-ahead decision making?

- How to enhance the approach to rank different possible day-ahead decisions, rather than just estimating the cost incurred by one of them, in order to more directly support a planner in his decision-making?

This last bullet point is addressed in the next chapter. In addition, there is also room for further improving the proposed approach by exploiting various existing machine learning algorithms (such as deep learning methods).

7

Ranking candidate look-ahead decisions with proxies

Overview

In this chapter, we extend the methodology presented in the previous two chapters to estimate the expected induced costs of real-time operation for a list of candidate look-ahead decisions, in order to rank them accordingly and identify 'good' operation planning decisions for the operation planner. More specifically, we propose to exploit Monte-Carlo simulation and machine learning to predict operation costs for various day-ahead unit commitment and economic dispatch decisions and a range of realizations of uncertain loads and renewable generations over the next day. We describe how to generate a database, how to apply supervised machine learning to it, and how to use the learnt proxies to rank candidate day-ahead decisions in terms of the expected operating cost they induce over the next day. We illustrate the approach on the IEEE-RTS96 benchmark.
References: This chapter is an adapted version of the following publication:

Duchesne, L., Karangelos, E., Sutera, A., and Wehenkel, L. (2020a). Machine learning for ranking day-ahead decisions in the context of short-term operation planning. *Electric Power Systems Research*, 189:106548.

Terminology and notations have been slightly adjusted for the sake of consistency with the rest of this manuscript. The text has also been processed to minimize overlap with respect to previous chapters and new results relative to an improved version of the validation protocol have been provided and are discussed.

7.1 Introduction

In this chapter we address the problem of ranking various candidate day-ahead planning decisions in terms of the *expected* next-day operating cost they induce, while considering exogenous uncertainties such as load or renewable generation. The final purpose of this would be to help selecting a day-ahead decision among a given set of candidate such decisions, while considering the impact of uncertainties on operation. More specifically, we propose a method to evaluate, for a list of candidate day-ahead decisions, their corresponding *expected* cost of real-time operation and then use this evaluation to rank the decisions to, *in fine*, ease the identification of a 'good' day-ahead decision.

We propose a generalization of the approach presented in chapter 6, to further help choosing among day-ahead decisions. We thus address the problem of building a *proxy* predicting operation costs with acceptable performances for several candidate day-ahead decisions, even unseen ones. This proxy could replace the real-time decision-making simulator in the CMC approach, allowing one to much more rapidly assess the expected cost of next-day operation for various candidate day-ahead decisions. We propose a methodology to automatically build a database combining different candidate day-ahead decisions with a sample of scenarios representing the expected range of possible next-day conditions, and show how to exploit such a database to learn and validate a proxy of real-time operation. In particular we investigate the use of neural networks and multitask learning [Caruana, 1997] to assess different day-ahead decisions and to rank them in terms of their induced next-day operating cost. We test this approach on the three-area IEEE-RTS96 system, while using the DC power flow model and the N-1 security criterion in order to simulate real-time operation as the resolution of a sequence of DC-SCOPF problems. In our case study, we consider as candidate day-ahead decisions several unit commitments and economic dispatches, as well as provisional wind curtailment, and we focus on the estimation of the expected value of the real-time total preventive control cost.

7.1.1 Chapter organization

The rest of this chapter is organized as follows. Section 7.2 states the problem studied and presents methodologies to generate automatically a database of real-time operation trajectories, to build proxies with supervised learning and to rank candidate day-ahead decisions with the help of these proxies. Section 7.3 reports the case study on the IEEE-RTS96 system, where we analyze the generalization capability of the proxies and we exploit them to rank candidate day-ahead decisions. Section 7.4 compares the proposed approach with analytical two-stage optimization and section 7.5 concludes and suggests future works.

7.2 Problem statement and proposals

7.2.1 Problem statement

In this work, we consider the standard setting wherein in day-ahead (da) the operation planner seeks to enable N-1 secure operation over every time period of the forthcoming day, and subject to uncertainty on renewable power generation and demand. In particular, the mission of the operation planner is to postpone outages scheduled for maintenance if necessary and to select in advance (i) the commitment and dispatch of generating units, and, (ii) provisional curtailment of wind power generation. To help in its decision-making, we assume that it has access to a simulator of real-time operation along the next day (nd), modeling both the physical behavior of the power system and the decision-making of control room operators in response to the resolution of uncertainties in real-time. We use a sequence of 24 SCOPF computations following the N-1 criterion while minimizing the costs of real-time operation to this end.

We also assume that the operation planner has at its disposal a generative model of day-ahead uncertainties, allowing to sample scenarios of next-day load and renewable generations, denoted $\{\xi^1_{nd}, \xi^2_{nd}, ...\}$. For a given candidate decision δ^i_{da} and a given scenario ξ^j_{nd}, the application of the real-time operation simulator allows the planner to anticipate real-time operation along the next day and in particular to evaluate the costs $y^{i,j}$ of real-time operation. Furthermore, we suppose that the operation planner is interested in evaluating, for any given day-ahead decision, the consequence over next-day operation by the expected value of the cost of operating the system the next day.

The problem addressed in this chapter then amounts to *screening candidate day-ahead decisions to select a good day-ahead decision in terms of its expected impact on next-day operating costs, while exploiting the available generative model and real-time operation simulator.*

7.2.2 Generating a database of day-ahead decisions and next-day scenarios

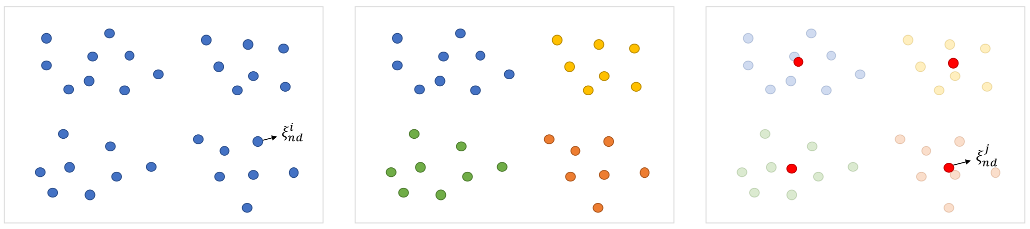
We extend the methodology described in chapter 6 to generate both a set of day-ahead decisions and a set of next-day scenarios. In this approach, m next-day scenarios $\{\xi^1_{nd}, \xi^2_{nd}, ..., \xi^m_{nd}\}$ are sampled with the generative model of day-ahead uncertainties available to the operation planner. They are then combined with k day-ahead decisions $\{\delta^1_{da}, \delta^2_{da}, ..., \delta^k_{da}\}$ generated as described in the following subsection.

7.2.2.1 *Generating k day-ahead decisions*

Here we assume that the planner has a day-ahead decision-making support software, e.g. in the form of a deterministic multi-period UC and ED program.

To generate the k day-ahead decisions with such a tool, we first generate a large sample ($n \gg k$) of next-day scenarios by using our generative model of uncertainties, and then apply to them the *k-means* clustering algorithm [Hastie et al., 2009], such that we obtain at the end k next-day scenarios $\{\xi^1_{nd}, \xi^2_{nd}, ..., \xi^k_{nd}\}$[1]. This is illustrated in Figure 7.1. We then compute k day-ahead decisions $\{\delta^1_{da}, \delta^2_{da}, ..., \delta^k_{da}\}$ from these k next-day scenarios $\{\xi^1_{nd}, \xi^2_{nd}, ..., \xi^k_{nd}\}$ by applying to each scenario the available day-ahead decision-making support software.

[1]If n is sufficiently large, the resulting k real-time scenarios will cover as well as possible the uncertainty space, for a given budget k.



(a) n next-day scenarios (b) k clusters (c) Resulting k next-day scenarios

Figure 7.1: Illustration of the k-means algorithm with $k = 4$ to obtain k next-day scenarios from n scenarios generated with a generative model of uncertainties.

7.2.2.2 *Simulating real-time operation*

For a given pair $(\delta^i_{da}, \xi^j_{nd})$ combining a day-ahead decision and a next-day scenario (we call this combination the *trajectory* $\tau^{i,j}$), we apply the real-time operation simulator to compute the corresponding next-day operation costs, denoted by $y^{i,j}$. This gives us a database $\{(\tau^{i,j}, y^{i,j})\}^N$ of size $N = k \times m$, with features describing day-ahead decisions and next-day scenarios as inputs and real-time operation costs as outputs, that we can exploit to learn proxies of the real-time operation simulator.

7.2.3 Learning and generalizing the proxies

The proxies we use here are simplified models of real-time operation, allowing to predict the real-time operation costs $y_{i,j}$ for a given day-ahead decision δ^i_{da} and a given next-day scenario ξ^j_{nd}. We use the methodology described in chapter 5 to build them with supervised learning.

7.2.3.1 *Splitting the database in training and test sets*

We are interested in evaluating the accuracy of our proxies to unseen day-ahead decisions, to unseen next-day scenarios, and to combinations of both. For that, we need to split the database into a learning set $\mathcal{L}$ and a validation set $\mathcal{V}$ used both to train the proxies, and three test sets $\mathcal{T}$ used to evaluate their generalization capabilities. Figure 7.2 shows this database decomposition in a graphical way.

7.2.3.2 *Assessing the accuracy of proxies*

In order to assess the accuracy of a proxy $h_p(\cdot)$ based on a sample $\mathcal{S}$ of size $|\mathcal{S}|$, we consider the square-loss $\ell(\hat{y}, y) = (\hat{y} - y)^2$ and compute the *empirical loss* by

$$\hat{L}(h_p, \mathcal{S}) = \frac{1}{|\mathcal{S}|} \sum_{(x^i, y^i) \in \mathcal{S}} \left(h_p(x^i) - y^i\right)^2 . \tag{7.1}$$

From there, we say that $h_p(\cdot)$ generalizes well to unseen day-ahead decisions if $\hat{L}(h_p, \mathcal{T}_{\Delta_u, \Xi_u}) \approx \hat{L}(h_p, \mathcal{T}_{\Delta_s, \Xi_u})$.

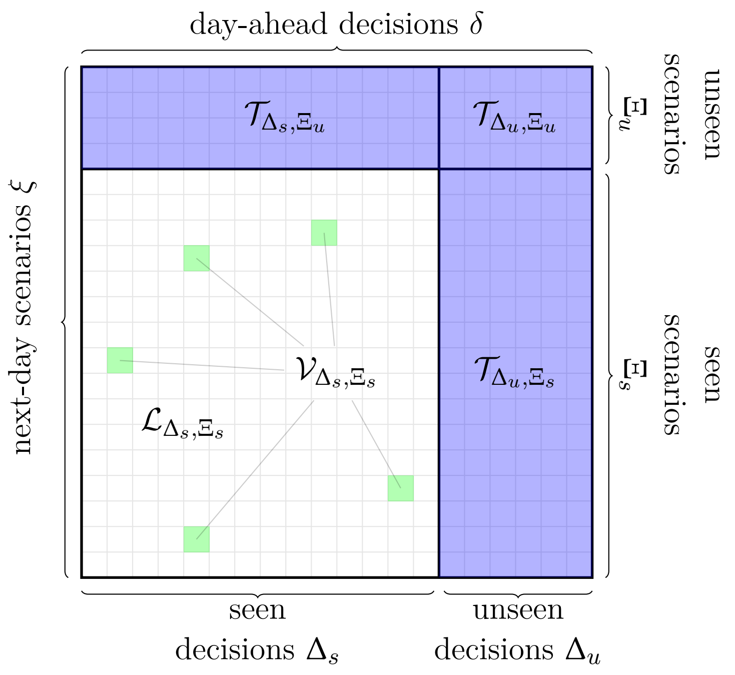

day-ahead decisions δ
$\mathcal{T}_{\Delta_s,\Xi_u}$
$\mathcal{T}_{\Delta_u,\Xi_u}$
$\mathcal{V}_{\Delta_s,\Xi_s}$
$\mathcal{L}_{\Delta_s,\Xi_s}$
$\mathcal{T}_{\Delta_u,\Xi_s}$
next-day scenarios ξ
Ξ_u unseen scenarios
Ξ_s seen scenarios
seen decisions Δ_s
unseen decisions Δ_u

Figure 7.2: Schematic representation of the database partition between train ($\mathcal{L}$, in white), validation ($\mathcal{V}$, in green) and test ($\mathcal{T}$, in blue) sets. Rows and columns respectively represent scenarios and decisions.

7.2.4 Using the proxies for ranking day-ahead decisions

Once one has shown that the proxies generalize well to unseen decisions, they could used in order to identify a good day-ahead decision. To do so, we propose to first select (randomly) a subset Δ_s of candidate day-ahead decisions that will be used to build a proxy of the real-time operation simulator. This subset should be large enough for the proxy to be able to generalize well to unseen decisions but small enough to avoid as much as possible the computational burden stemming from the use of the heavy real-time simulator. For all $\delta^i_{da} \in \Delta_s$ and $\xi^j_{nd} \in \{\xi^1_{nd}, \xi^2_{nd}, ..., \xi^m_{nd}\}$, we compute the next-day operation costs $y^{i,j}$ with the real-time simulator and then we build the proxy with supervised learning. Then we apply the proxy to predict the next-day operation cost $y^{i,j}$ for all $\delta^i_{da} \in \Delta_u$ and $\xi^j_{nd} \in \{\xi^1_{nd}, \xi^2_{nd}, ..., \xi^l_{nd}\}$[2]. After that we average the predictions over the l scenarios to have an estimate $\hat{\mu}^i_{y_p}$ of the expected next-day operation cost for each decision in $\delta^i_{da} \in \Delta_u$ and we average the $y^{i,j}$ over the m scenarios to have an estimate $\hat{\mu}^i_y$ for each decision in $\delta^i_{da} \in \Delta_s$. Finally, these estimates are used to rank the candidate decisions according to their expected next-day operation cost.

[2]Note that that these l scenarios can be different from the m scenarios generated for learning the proxy.

7.3 Case study on the IEEE-RTS96 benchmark

In order to test our proposed methodology, we place ourselves in the context of day-ahead operation planning, when the operation planner has to select a unit commitment and economic dispatch for the next day, as well as provisional wind curtailment.

Our real-time operation simulator as well as the scenario generator are implemented in JULIA. We use Python for learning: *scikit-learn* for the clustering algorithms and *Pytorch* as the deep learning framework.

We consider the same test system as in chapter 6, i.e. a modified version of the 3-area IEEE-RTS96 benchmark [Grigg et al., 1999]. The models of uncertainties, day-ahead decision-making and real-time operation simulator are also identical. They are described in section 6.3.1.

The results of this section are complemented by additional results in appendix C.

7.3.1 Description of the database

For our study, we generated $k = 20$ candidate day-ahead decisions, applying the methodology of Section 7.2.2 with $n = 20,000$ next-day scenarios. We combined further $m = 600$ next-day scenarios with each one of these $k = 20$ day-ahead decisions to yield $12,000$ trajectories of 24 hourly time-steps.

In our database, the input-features per hourly snapshot of each trajectory are the following:

- demand realizations for each load,
- wind generation realizations for each wind farm minus the day-ahead provisional wind curtailment,
- difference between the real-time scenario and the forecast scenario used to generate the corresponding day-ahead decision (in MW and in %),
- total demand, total wind generation and net load,
- maximum and minimum total generation capacity,
- hour of the day.

During a preliminary study, we found that it was preferable not to use the active power levels as features. The candidate decisions are thus only described with the difference between the real-time and forecast scenarios as well as the maximum and minimum total generation capacity. We provide further results in appendix C, comparing the performances of the proxies with and without these active power level features.

Concerning the outputs of the database, we focus here on the total cost of preventive actions, which is the sum of the preventive generation redispatch cost, the preventive load shedding cost and the preventive wind curtailment cost along a next-day scenario.

7.3.2 Analysis of the 20 candidate day-ahead decisions

We begin our analysis with the comparison of the unit commitments (on-off statuses) and economic dispatches (active power levels) of the 20 candidate day-ahead decisions. For that, we represent each decision as a vector containing as elements either the unit commitment or day-ahead dispatch of each one of the 96 generating units for each one of the 24 hours of the day. To compare the vectors pairwise, we use the *Hamming distance*, expressed as the number of components for which the two vectors differ. The results are presented in the form of heat maps and can be seen in Figure 7.3.

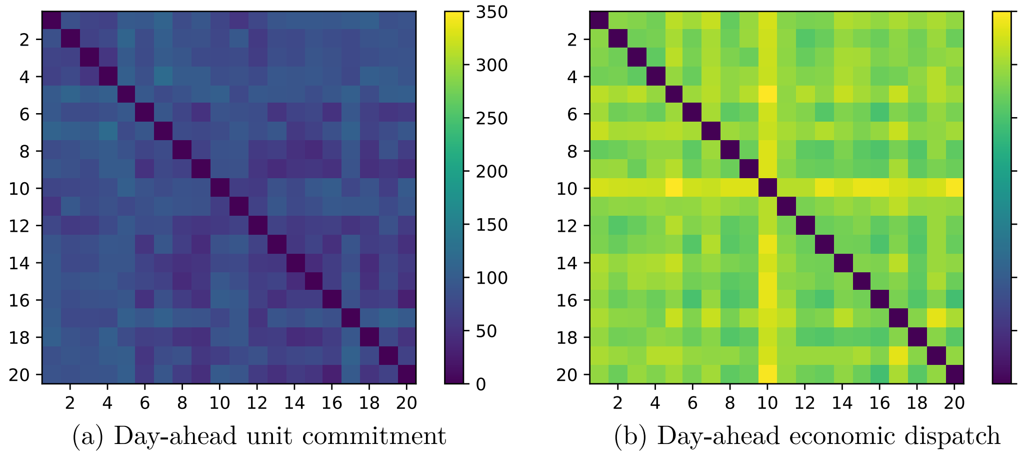


Figure 7.3: Hamming distance between each candidate day-ahead decision for respectively the unit commitment and day-ahead dispatch.

We observe that the decisions are different, and that the distances between them are of same order of magnitude, both in the unit-commitment space and the space of economic dispatches. On average, the Hamming distance for unit commitments is equal to 3% of the vector components. This proportion rises to 12% for the economic dispatches.

Another analysis we can make to compare candidate day-ahead decisions is to look at their impact on next-day operation costs. Since we used the same 600 next-day scenarios with each candidate day-ahead decision, the results are directly comparable. We look at some statistics of the total preventive control cost over the 600 scenarios, such as the average value $\hat{\mu}_y^i$, the standard error which is defined as $\frac{\hat{\sigma}_y}{\sqrt{m}}$, with $m = 600$ and $\hat{\sigma}_y$ the standard deviation of $y^{i,j}$ over the 600 scenarios, and the minimum and maximum values of $y^{i,j}$. The results can be seen in Table 7.1, which is sorted in increasing order of the mean total preventive control cost.

When analyzing the mean total preventive control cost, we see that there is clear difference between decisions. For instance decision 17, the most costly decision, is in average 300,000€ more expensive in real-time than decision 2. If one analyzes the components of the total preventive control cost, it can be seen that this decision leads to the largest load shedding cost in average over next day. Note that given the standard error values, we cannot guarantee that

Table 7.1: Mean, standard error, minimum and maximum value of the total preventive control cost for the $k = 20$ studied day-ahead decisions computed over $m = 600$ next-day scenarios. Sorted in increasing order of mean.

δ^i_{da}	Mean	Standard error	Min	Max
	$(\times 10^6)$	$(\times 10^4)$	$(\times 10^5)$	$(\times 10^6)$
2	1.641	3.439	5.278	6.339
19	1.656	3.163	4.912	7.024
1	1.661	3.350	5.072	6.672
3	1.676	3.263	5.382	7.404
11	1.689	3.384	5.136	7.466
8	1.691	3.411	5.970	7.091
6	1.692	3.150	5.546	8.139
7	1.719	3.741	5.151	7.680
10	1.728	3.615	5.153	7.664
9	1.731	3.705	4.788	7.182
4	1.732	3.264	6.359	6.215
20	1.740	3.560	5.835	7.119
16	1.746	3.412	5.270	6.781
14	1.748	3.518	5.944	7.608
18	1.788	3.700	5.061	7.287
12	1.797	4.046	5.248	8.192
5	1.798	3.485	6.345	6.317
13	1.855	3.631	5.928	6.293
15	1.872	3.793	5.412	7.547
17	1.917	3.962	6.546	7.912

decision 2 is effectively the candidate decision with the smallest expected next-day total preventive control cost, but the less expensive decision on average should nevertheless be among the first few decisions of Table 7.1.

7.3.3 Machine learning protocol

To predict the total preventive control cost along a next-day trajectory we divide this trajectory in 24 hourly snapshots, predict the total preventive control cost for each hour (denoted as hourly prediction in chapter 6) and then sum the 24 predictions. Note that we provide information about the hour of the day in two ways, one using one single input with values ranging from 1 to 24 and another one using 24 binary inputs with a one-hot-encoding. This leaves us with 240 hourly input features and one single output variable.

As supervised learning algorithm, we chose the NN algorithm and used a grid search (i.e. we tested all possible combinations of meta-parameters, given their selected range of values) to find a suitable configuration of meta-parameters among the following candidate values: 3, 4, 5 or 6 layers and 50, 100 and 200 neurons per layer. Furthermore, we tested 5 different initializations of the network weights. This procedure is repeated for each experiment and the best combination of meta-parameters may differ from one experiment to another. The other meta-parameters were kept constant. In particular, we used a batch size of 200, a learning rate of 10^{-3}, the Adam optimizer [Kingma and Ba, 2014] and a weight decay of 10^{-4}. The maximum number of epochs is 200, and we keep the model corresponding to the epoch minimizing the loss on the validation set.

In reference to Figure 7.2, 100 next-day scenarios are always kept out in order to yield the test sets. As concerns the splitting along the day-ahead decisions, we investigate different settings in the sequel. In any case, the validation set $\mathcal{V}$ corresponds to 5% of the trajectories not used in the test sets; it is used to select the meta-parameters of the learning algorithms[3], while the remaining 95% provide the learning set $\mathcal{L}$ used only to tune the parameters of the neural network predictors.

To evaluate the proxies, we always use the R^2-score.

7.3.4 Generalization over day-ahead decisions

7.3.4.1 *Leave-one-decision-out*

For this experiment, we use trajectories from $k-1$ decisions to learn a proxy and we test it with data from the k^{th} (unseen) decision applied on unseen scenarios. We redo this experiment k times, with each time a different unseen decision in the test set. We then average the k test scores obtained. If this score is high, the proxy is able to well generalize to unseen decisions. Figure 7.4 illustrates the principle of the leave-one-decision-out experiment.

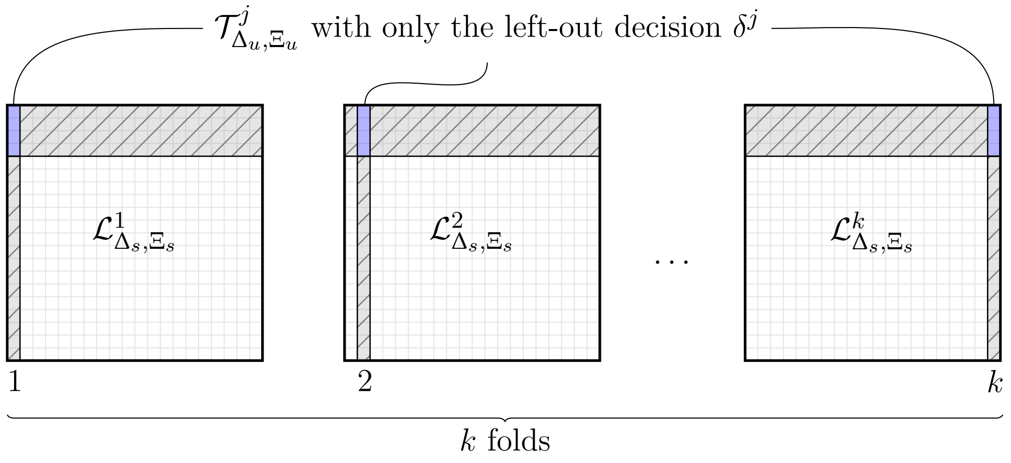

$\mathcal{T}^j_{\Delta_u,\Xi_u}$ with only the left-out decision δ^j
$\mathcal{L}^1_{\Delta_s,\Xi_s}$
$\mathcal{L}^2_{\Delta_s,\Xi_s}$
$\dots$
$\mathcal{L}^k_{\Delta_s,\Xi_s}$
1
2
k
k folds

Figure 7.4: Principle of the leave-one-decision-out experiment (unused data are in grey).

[3]We keep the combination of meta-parameters minimizing the loss on the validation set.

To select the meta-parameters, we performed a grid search and kept for each fold the model with the combination of meta-parameters leading to the maximum score on the validation set. We thus have 20 proxies, each with a different day-ahead decision left out. The statistics of the test scores of these 20 proxies on the different test sets can be seen in Table 7.2.

We observe that the scores of unseen decisions are almost equal to those of seen decisions. Therefore the proxy is able to generalize to unseen decisions. Furthermore, the scores are quite good, meaning that the proxy is able to predict the total preventive control cost with acceptable performances. The trajectory scores are generally a bit smaller than the hourly scores but are still good. We also see that when the scenarios have already been seen by the neural network, the score is close to 1, even for unseen decisions.

Table 7.2: Statistics of the hourly (H.) and trajectory-wise (T.) R^2-scores obtained over the 20 folds of the leave-one-decision-out experiment.

	H. train score Seen dec. Seen scen.	H. test score Seen dec. Unseen scen.	H. test score Unseen dec. Unseen scen.	H. test score Unseen dec. Seen scen.
Mean	0.9876	0.9262	0.9207	0.9820
Std	0.0004	0.0074	0.0144	0.0062
Min	0.9863	0.9070	0.8984	0.9648
Max	0.9883	0.9360	0.9434	0.9889
	T. train score Seen dec. Seen scen.	**T. test score Seen dec. Unseen scen.**	**T. test score Unseen dec. Unseen scen.**	**T. test score Unseen dec. Seen scen.**
Mean	0.9852	0.9069	0.8987	0.9791
Std	0.0015	0.0101	0.0196	0.0057
Min	0.9813	0.8798	0.8703	0.9614
Max	0.9870	0.9200	0.9395	0.9866

7.3.4.2 *Machine learning improvement: multitask learning*

Instead of predicting only the target output with a proxy, we can try to simultaneously predict a vector of outputs; this corresponds to multitask learning [Caruana, 1997]. In our case, this means predicting at the same time the target (total preventive control cost) and auxiliary outputs as the preventive redispatch cost, the preventive load shedding cost and the preventive wind curtailment cost. The main advantage of this method is that the model can benefit from extra knowledge brought by the additional auxiliary outputs, so as to improve the performances of the proxy on the main target output.

We repeated the leave-one-decision-out experiment while exploiting this multitask learning approach. We performed again a grid search analysis with the same meta-parameters as before, but this time predicting a vector of outputs. We keep the network configuration maximizing the validation score for the total preventive control cost, since it is the target of interest, for each fold. Looking only at the predictions of the total preventive control cost, we obtain the results presented in Table 7.3.

We see that with multitask learning, we can improve the test scores of the proxy by 2%, which is quite interesting given that the scores were already close to the maximum score.

Table 7.3: Statistics of the hourly (H.) and trajectory-wise (T.) R^2-scores obtained over the 20 folds of the leave-one-decision-out experiment, with multitask learning.

	H. train score Seen dec. Seen scen.	H. test score Seen dec. Unseen scen.	H. test score Unseen dec. Unseen scen.	H. test score Unseen dec. Seen scen.
Mean	0.9868	0.9444	0.9387	0.9817
Std	0.0005	0.0057	0.0134	0.0052
Min	0.9860	0.9286	0.9049	0.9704
Max	0.9874	0.9523	0.9595	0.9884
	T. train score Seen dec. Seen scen.	T. test score Seen dec. Unseen scen.	T. test score Unseen dec. Unseen scen.	T. test score Unseen dec. Seen scen.
Mean	0.9845	0.9257	0.9157	0.9784
Std	0.0017	0.0080	0.0240	0.0066
Min	0.9797	0.9021	0.8634	0.9620
Max	0.9861	0.9351	0.9519	0.9864

7.3.4.3 *Impact of the number of training day-ahead decisions*

In this experiment, we first selected randomly $l < k$ candidate decisions that we consider as our test decisions. Then we learn $k - l$ times a proxy, each time adding a new decision (different from the l test decisions) in the training set. At the end, we compare the $k - l$ test scores obtained both on the test set with seen decisions and unseen scenarios and the test set with unseen decisions and unseen scenarios and check when the test score corresponding to unseen decisions is similar to the one corresponding to seen decisions. This would indicate that enough decisions have been taken into account for a proxy to be able to generalize well to unseen decisions. Figure 7.5 illustrates the principle of this experiment.

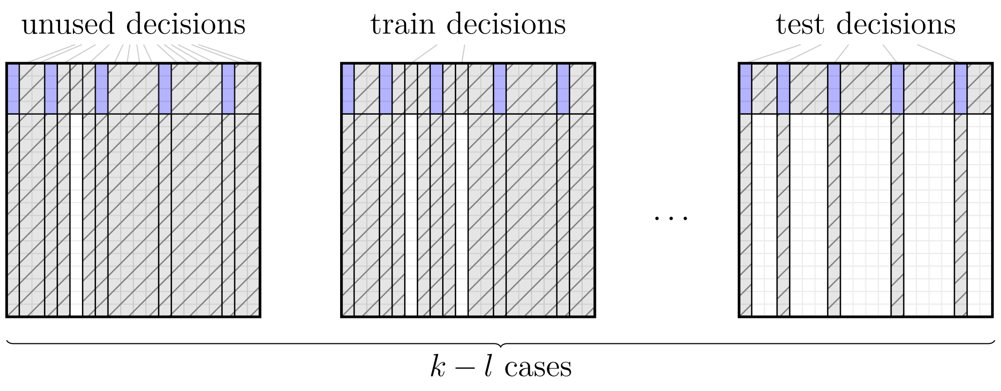
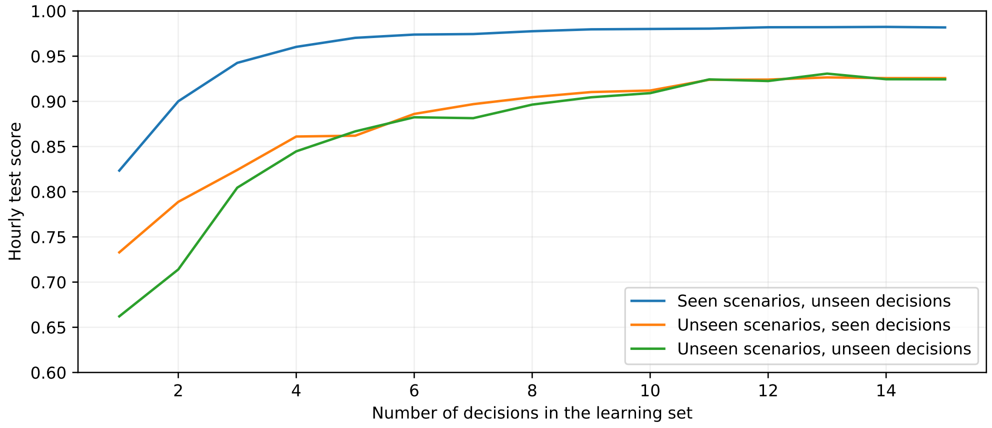
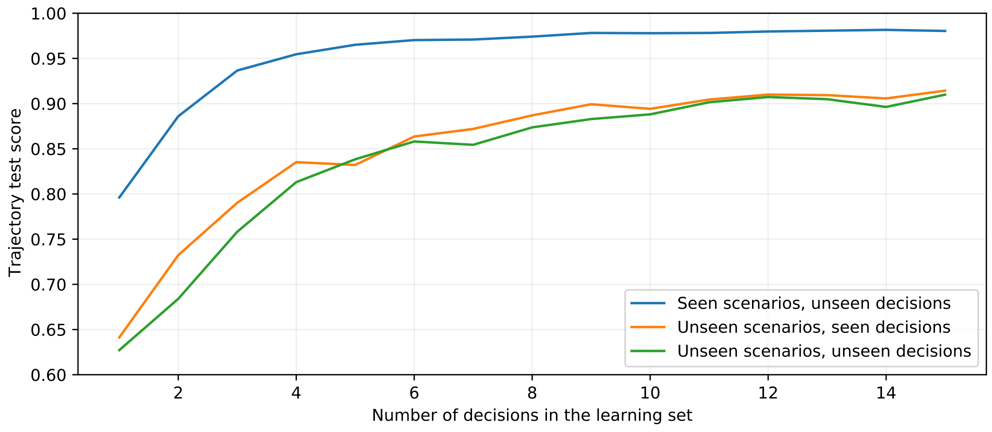


Figure 7.5: Principle of the gradual increase of the learning set $\mathcal{L}$ size experiment.

Here we report the results obtained with $l = 5$. We use the best configuration of neural network on average on the validation score from the previous experiment (5 layers and 50 neurons per layer) and we repeated each experiment 5 times, with a different initialization of the neural network weights. The results for the best initialization (based on the validation score) are presented in Figure 7.6. We see that the proxy is able to generalize well with only 5 decisions in the training set and that the scores are already quite good. With 10 decisions we get scores close to those reported for 19 decisions in Table 7.2.



(a) Hourly scores



(b) Trajectory scores

Figure 7.6: Mean (a) hourly and (b) trajectory R^2-scores as a function of the number of day-ahead decisions in the learning set.

7.3.5 Using the proxies for ranking day-ahead decisions

In this subsection we consider the use of the learnt proxies in order to rank a set of unseen candidate decisions according to the expected next-day operating cost they would induce. Since we have seen that the proxies generalize well to unseen decisions, we conjecture that they could be used in order to identify a good day-ahead decision, while avoiding as much as possible to resort to heavy SCOPF computations over large samples of next-day scenarios combined with each candidate day-ahead decision.

We compare two methods exploiting the proxies to estimate the expected total preventive control cost μ_y^i associated to a day-ahead decision δ_{da}^i. For each method, we first select a subset Δ_s of candidate day-ahead decisions that we exploit to build a proxy. Since we noticed in Figure 7.6 that only 5 decisions are needed in the learning set for the proxies to generalize well on unseen decisions, we selected randomly 5 decisions and assigned them to the learning set. We performed this operation 5 times, each time with a different set Δ_s of size 5. We also realized this experiment with 10 decisions in the learning set, for comparison.

7.3.5.1 *Method 1*

For this method, we first generate 2000 next-day scenarios. We then apply the proxy to predict the total preventive control cost $y^{i,j}$ for all $\delta_{da}^i \in \Delta_u$ and $\xi_{nd}^j \in \{\xi_{nd}^1, \xi_{nd}^2, ..., \xi_{nd}^{2000}\}$. Finally we average the predictions over the 2000 scenarios to have an estimate $\hat{\mu}_{y_p}^i$ of the expected total preventive control cost for each decision in $\delta_{da}^i \in \Delta_u$.

7.3.5.2 *Method 2*

This method extends the previous one by using the control variates approach presented in chapter 6 to correct a possible bias in the estimation $\hat{\mu}_{y_p}^i$ performed with the proxies. For that, we compute the estimated total preventive control cost $\hat{\mu}_{y_p}^{y,i}$ of decision $\delta_{da}^i \in \Delta_u$ as follows:

$$\hat{\mu}_{y_p}^{y,i} = \hat{\mu}_{y_p}^i + \sum_{j=1}^{100} (y^{i,j} - y_p^{i,j}), \tag{7.2}$$

where the 100 scenarios belongs to the set Ξ_u.

7.3.5.3 *Results*

We consider as ground truth the average value of y presented in Table 7.1 and computed with 600 scenarios per decision. Note that with both methods, for each $\delta_{da}^i \in \Delta_s$, the estimated expected value of total preventive control cost is the one presented in Table 7.1, given that the SCOPF calls had to be made to build the proxy.

To analyze the quality of the ranking, we use the Kendall's tau coefficient and the Spearman's rank correlation coefficient. The Kendall's tau coefficient τ is defined as $\tau = \frac{C-D}{C+D}$, where C is the number of concordant pairs and D the number of discordant pairs. The Spearman's rank correlation coefficient ρ is defined as $\rho = \frac{cov(rg_X, rg_Y)}{\sigma_{rg_X}\sigma_{rg_Y}}$, where $cov(rg_X, rg_Y)$ is the covariance of the rank variables and σ_{rg_X} and σ_{rg_Y} are the standard deviations of the rank variables.

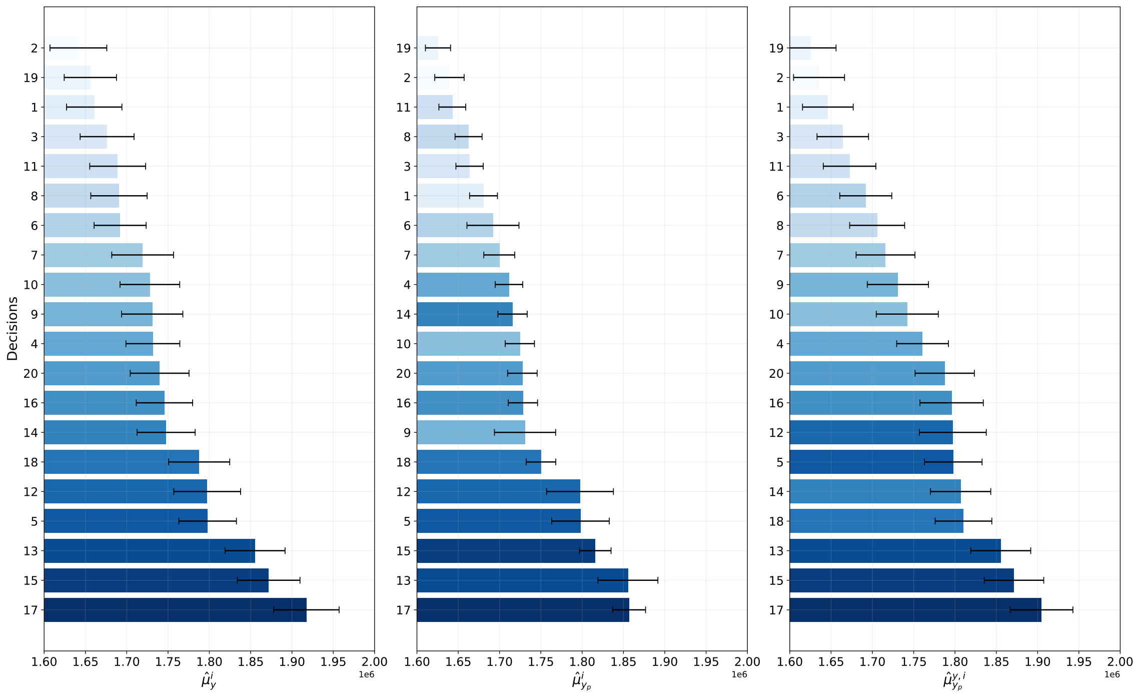
Both metrics minimum and maximum values can be found in Table 7.4 for the different estimations. One can see that with the control variates approach,

Table 7.4: Minimum and maximum Kendal's tau coefficient and Spearman's correlation coefficient for both estimation methods, with 5 or 10 decisions in the learning set.

		$\hat{\mu}_{yp}$		$\hat{\mu}^{y}_{yp}$	
		Min	Max	Min	Max
5 decisions	τ	0.5158	0.8421	0.8211	0.9263
	ρ	0.7038	0.9534	0.9338	0.9835
10 decisions	τ	0.6211	0.8632	0.8211	0.9159
	ρ	0.7624	0.9654	0.9353	0.9820

these metrics are closer to 1, meaning that there is a stronger relationship between the true ranking and the estimated one. One can also notice that the ranking is better when there are 10 decisions in the learning set, but at the cost of more SCOPF calls.

Figure 7.7 and Table 7.5 present the different estimations of the real-time operating costs associated to a decision (with their standard errors for Figure 7.7) as well as the corresponding ranking for the best case experiment when there is only 5 decisions in the learning set.



(a) Ground truth $\hat{\mu}^{i}_{y}$ (b) Proxy $\hat{\mu}^{i}_{yp}$ (c) CV approach $\hat{\mu}^{y,i}_{yp}$

Figure 7.7: Figure (a) is the ground truth. The decisions are sorted in crescent order of expected preventive control cost and the color of the decisions are of crescent intensity while following the order of the decisions. Figures (b) and (c) correspond respectively to the proxy method and the control variates method. The decisions are again sorted in crescent order but the colors are maintained to help visualizing the differences in the obtained rankings. The error bars show the $\pm\hat{\sigma}^{i}/\sqrt{n}$ interval, where $\hat{\sigma}^{i}$ is the sample estimate of the standard deviation of respectively y^{i}, y^{i}_{p} and $(y^{i} - y^{i}_{p})$ and n is respectively equal to 600, 2000 and 100.

Table 7.5: True and estimated expected total preventive control cost per decision and the associated ranking $r(\cdot)$. The decisions used to learn the proxies are colored in red.

δ^i	$\hat{\mu}_y^i$ ($\times 10^6$)	$\hat{\mu}_{y_p}^i$ ($\times 10^6$)	$\hat{\mu}_{y_p}^{y,i}$ ($\times 10^6$)	$r(\hat{\mu}_y^i)$	$r(\hat{\mu}_{y_p}^i)$	$r(\hat{\mu}_{y_p}^{y,i})$
2	1.641	1.639 −	1.635 −	2	19 $\wedge_1$	19 $\wedge_1$
19	1.656	1.625 −	1.626 −	19	2 $\vee_1$	2 $\vee_1$
1	1.661	1.681 +	1.646 −	1	11 $\wedge_2$	1 =
3	1.676	1.664 −	1.664 −	3	8 $\wedge_2$	3 =
11	1.689	1.643 −	1.672 −	11	3 $\vee_1$	11 =
8	1.691	1.662 −	1.706 +	8	1 $\vee_3$	6 $\wedge_1$
6	1.692	1.692 ∘	1.692 ∘	6	6 =	8 $\vee_1$
7	1.719	1.700 −	1.716 −	7	7 =	7 =
10	1.728	1.725 −	1.742 +	10	4 $\wedge_2$	9 $\wedge_1$
9	1.731	1.731 ∘	1.731 ∘	9	14 $\wedge_4$	10 $\vee_1$
4	1.732	1.711 −	1.761 +	4	10 $\vee_2$	4 =
20	1.740	1.728 −	1.788 +	20	20 =	20 =
16	1.746	1.728 −	1.796 +	16	16 =	16 =
14	1.748	1.716 −	1.807 +	14	9 $\vee_4$	12 $\wedge_2$
18	1.788	1.750 −	1.810 +	18	18 =	5 $\wedge_2$
12	1.797	1.797 ∘	1.797 ∘	12	12 =	14 $\vee_2$
5	1.798	1.798 ∘	1.798 ∘	5	5 =	18 $\vee_2$
13	1.855	1.855 ∘	1.855 ∘	13	15 $\wedge_1$	13 =
15	1.872	1.816 −	1.871 −	15	13 $\vee_1$	15 =
17	1.917	1.857 −	1.905 −	17	17 =	17 =

One can directly notice that the control variates approach allows to improve the estimation of the preventive total cost and thus the ranking, but it has a larger computational burden (50% more SCOPF calls) than method 1. Note that, even if the first decision in both estimated rankings is not the correct decision, one can see that the good decisions (small expected total preventive control cost) are identified with both methods.

7.3.5.4 *Updated results after publication*

When writing [Duchesne et al., 2020a], from which the results presented in this chapter are taken, we did not have more than 600 scenarios for which real-time operation was simulated with a SCOPF, and thus we used them to define a ground truth. This is, however, not ideal, given that the 600 scenarios exploited to compute the 'true' expected total preventive control cost for each decision are also used to learn the proxies (for 500 scenarios) and to correct a possible bias with the control variates method (for the 100 remaining scenarios). Doing so could result in assessing rankings in an optimistic way. In order to evaluate the ranking methods, it is therefore a better practice to compute the ground truth with independent scenarios, that are neither used to train the proxy or to correct a bias.

In this section, we improve upon our evaluation methodology and compare the rankings computed in the previous section with the help of proxies with a ground truth obtained from independent trajectories. To obtain this ground truth, we simulate real-time operation for 2000 independent scenarios[4] and each of the 20 candidate day-ahead decisions and average the results to compute an estimate $\hat{\mu}^i_{y,2000}$ of the expected total preventive control cost for each decision i. Note that one can expect this updated ground truth to be closer to the true ground truth, given that more scenarios are used to compute it.

The detailed comparisons between the rankings obtained with the proxies and this updated ground truth can be found in appendix C, but we state the main conclusions here.

With this improved validation protocol, the conclusions drawn in the previous section are validated. Both rankings obtained from the proxies are close to the updated true ranking. In particular, the less costly candidate decisions and the more costly ones are correctly identified with both methods and the control variates approach clearly outperforms method 1. Furthermore we see that the first decision in both estimated rankings is this time the correct decision. The rankings obtained with the control variates method are even closer to the independent ground truth ranking $r(\hat{\mu}^i_{y,2000})$ than to the previous ground truth ranking $r(\hat{\mu}^i_y)$.

7.3.6 Computing times

The average computing times of the experiments presented in this paper, with a MacBookPro (2.2GHz Intel Core i7, 16GB RAM), are presented in Table 7.6.

In the upper part of the table, we highlight the CPU times needed to assess a single scenario and day-ahead decision, composed of first sampling the scenario, and then computing the costs induced by the decision either via the detailed SCOPF-wise simulation or via applying the learnt proxy. We observe a gain of a factor 10,000 with the ML proxy.

Table 7.6: Average computing times

	Average time (s)
Sampling of one scenario	0.002
Real-time SCOPF simulation for one scenario	303.600
Using the ML proxy for one scenario	0.027
Learning/validation/test dataset generation	3,643,200.000
Learning one proxy	5,660.000
Optimizing the meta-parameters of the proxy	169,813.000

The CPU times in the lower part of the table correspond to the off-line learning stage based on the leave-one-decision-out experiment. We observe that

[4]These 2000 scenarios are in this case the ones used to estimate $\hat{\mu}^i_{y_p}$ with the proxies.

the bulk of the computations are about the dataset generation corresponding in our simulations to 12,000 trajectories (i.e. 600 scenarios combined with 20 day-ahead decisions) times 24 hours, while optimizing the meta-parameters of the machine learning method corresponded in our case to tuning 30 different proxy models. It is important to realize that the computing times of both the dataset generation and the meta-parameters' optimization parts could be reduced by using massive parallel computations, respectively to about 303 seconds and 5,660 seconds.

7.4 Related works

Our approach shares strong similarities with analytical two-stage optimization such as stochastic unit commitment [Håberg, 2019], where commitment decisions are taken in the first stage and dispatch decisions are taken in the second stage considering uncertainties in the form of several possible future operating conditions (scenarios). However, we see several differences. First, our approach can be used to rank under uncertainties various sets of candidate first-stage decisions. In contrast, solving a two-stage stochastic program returns a (locally or globally) optimal candidate decision but no quantitative information on the merit of this decision with respect to the other available alternatives. Furthermore, our methodology relies essentially only on massively parallel simulations of the operator's real-time behavior and induced costs along different scenarios. On the other hand, stochastic unit commitment approaches require that the real-time operation strategy of the operator is explicitly modeled in the form of a second-stage optimization problem, and typically impose strong restrictions on such second-stage models in order to yield computationally tractable two-stage stochastic formulations.

7.5 Conclusions and future work

With respect to the previous chapter, we took here the additional steps of (i) studying how to generalize the method over 'unseen' day-ahead decisions, and (ii) testing the usefulness of the learnt proxies for ranking candidate day-ahead decisions. To do so, we proposed a methodology to build automatically a database of candidate day-ahead decisions and next-day scenarios, to each combination of which we can apply a real-time operation simulator in order to compute the resulting next-day operating costs. We also proposed a methodology to build and validate proxies, exploiting such a database.

We showed with a case study on the three-area IEEE-RTS96 benchmark that our proxies of real-time operation, predicting the next-day total preventive control cost, are able to generalize well to unseen decisions. We also showed that they can be exploited to identify candidate decisions with smallest expected induced costs in real-time operation.

The problem addressed in this chapter is the difficult problem of finding 'optimal' day-ahead decisions considering uncertainties. It is important to note that, with the approach we have proposed in this study, we did not seek to find the optimal decision among the space of all possible day-ahead decisions. Instead, we proposed to rank a list of given candidate decisions in order to identify the most suitable ones, provided that they are produced with the method we have used, based on forecast scenarios on which the same UC program has been applied.

There are many possible future directions of research following this work. We highlight the following ones:

- how to improve the ranking methodology;
- how to adapt the proposed methodology to evaluate, instead of the expected real-time operation costs for a given decision, the probability to meet the reliability target in real-time and in particular be able to evaluate the probability of rare events;
- how to exploit the presented methodology to reverse the problem and find hints for the operation planner about what forecast scenarios or set of forecast scenarios could lead him to a 'good' day-ahead decision.

Regarding our last suggestion for future works, doing so converts the problem of finding the optimal day-ahead decision in the space of all candidate day-ahead decisions that could be generated with any method one can imagine to a problem of finding the optimal forecast scenario (or set of representative forecast scenarios) in the space of all possible forecast scenarios. The advantage is that the space of possible forecast scenarios is much easier to define and optimize over than the space of candidate day-ahead decisions. Once this optimal scenario (or set of representative scenarios) is found, the UC program can be applied to compute a 'good' day-ahead decision. This would reduce the initial problem of finding an optimal decision in the space of all decisions to finding an 'optimal' day-ahead decision in the space of decisions generated with the defined UC program based on the such selected forecast scenario (or set of representative scenarios). This reduced problem is much smaller than the initial space of decisions. The decisions obtained with this method are guaranteed to be feasible, although being most probably suboptimal regarding the space of all candidate decisions.

Part III

Machine learning for convex approximations of optimization problems

8

Input Convex Neural Networks for convex piecewise linear approximations of optimization problems

Overview

In this chapter, we propose to use Input Convex Neural Networks (ICNNs) to build convex approximations of non-convex feasible sets of optimization problems, in the form of a set of linear inequalities in a lifted space. Our approach may be tailored to yield both inner- and outer- approximations, or to maximize its accuracy in regions closer to the minimum of a given objective function. We illustrate the method on two-dimensional toy problems and motivate it by various instances of reliability management problems of large-scale electric power systems.

References: This chapter expands on the following document submitted for publication:

Duchesne, L., Louveaux, Q., and Wehenkel, L. (2021). Supervised learning of convex piece-wise linear approximations of optimization problems. Submitted for publication.

Compared to this document, the introduction and conclusion sections have been adapted for the sake of consistency with the rest of the manuscript, and illustrations in sections 8.3.3, 8.3.4 and 8.3.5 have been added.

8.1 Introduction

During power systems operation planning, the operator regularly solves large optimization problems such as OPF problems to determine control actions and ensure the reliability of the power system over the time horizon considered. However, these problems (when not approximated) are non-linear and non-convex.

Non-convex optimization problems are often intractable or computationally intensive, especially when there is a large number of variables and constraints such as in power systems applications. On the other hand, efficient and tractable optimization algorithms exist to solve convex optimization problems, and especially linear programs [Boyd and Vandenberghe, 2004]. It is therefore of interest for the operator to have access to convex approximations of non-convex optimization problems, especially in contexts where there is a lot of uncertainty, and the problems must be solved for a large range of possible future operating conditions in a short period of time.

With this motivation in mind, we propose in this chapter to develop machine learning approaches that would allow us to automatically build convex approximations of non-linear and/or non-convex feasible domains in the form of a set of linear (and thus convex) constraints, in order to exploit the extremely efficient methods and solvers already available for linear programs. If the obtained linear approximation of the feasible set is an inner approximation, it would allow us to generate feasible solutions and upper bounds of the minimal value of the original problem, and if it is an outer approximation it would provide lower bounds.

The ICNN [Amos et al., 2017] is a neural network with constraints on its parameters and activation functions implying that the learnt input-output function $h(x, \theta)$ is a convex function of the inputs x. While this method was originally proposed in a regression context (e.g. to build convex approximations of objective functions of optimization problems), we propose to use it in a classification setting in order to build convex approximations of feasible sets of optimization problems. Note that this approach can also be of interest in the context of convex feasible sets. Indeed, there is no guarantee that a region learnt with classical learning algorithms to approximate a convex feasible set would be convex as well. Using an ICNN would allow us to enforce this property.

This chapter is organized as follows. Section 8.2 presents the main idea, i.e. the feed-forward ICNN model and our proposed adaptation to represent and learn convex approximations of a feasible domain, and how this learnt convex approximation can be used effectively in an optimization problem if the ICNN architecture is using piecewise linear activation functions such as ReLU or leaky-ReLU[1]. Section 8.3 presents some illustrative experiments, section 8.4 discusses related works, and section 8.5 possible real-world applications and directions for future work. Finally, we provide an appendix 8.A, which gives further details about mathematical proofs.

8.2 ICNNs for convex classification and optimization

We consider feed-forward networks as shown in Fig. 8.1, where $x \in \mathbb{R}^n$ denotes the vector of inputs, $\theta = \{W_i^z, W_i^x, b_i\}_{i=0,\ldots,k-1}$ the set of parameters of the network, and g_i the activation function used in the ith layer. The relationship

[1]Leaky-ReLU is defined as

$$leaky - ReLU(z) = \max(0.01z, z).$$

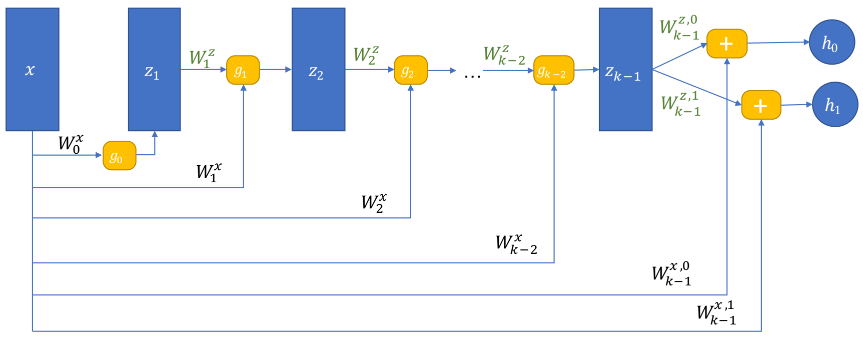


Figure 8.1: Representation of the layers of an ICNN with two outputs h_0 and h_1. The weights W^z, colored in green, are constrained to be non-negative.

between the inputs and the outputs of layer i of such a model is thus recursively given by:

$$z_{i+1} = g_i(W_i^z \times z_i + W_i^x \times x + b_i) \text{ for } i = 0, ..., k-1,$$

with $z_0 = 0$ and (the outputs) $h(x, \theta) = z_k$.

Notice that compared to a classical feed-forward neural network, *pass-through layers* connecting directly the input vector x to each layer have been added to increase the representation power of these networks. In the ICNN model [Amos et al., 2017], the weights W_i^z for $i = 1, ..., k-1$ are constrained to be non-negative and the activation functions g_i are constrained to be convex and non-decreasing. These two conditions are sufficient to guarantee that the activations z_i of each layer and hence the outputs $h(x, \theta)$ are convex functions of the input vector x. In the rest of this paper we will use (convex and non-decreasing) *piecewise linear* activation functions g_i (such as ReLU or leaky-ReLU).

8.2.1 Convex set representation by an ICNN with two outputs

Among several possibilities, we decided to use an ICNN with two (scalar) outputs h_0 and h_1 to create a binary classifier, where an input is associated to the target class 0 if $g(x, \theta) = h_1(x, \theta) - h_0(x, \theta) \leq 0$.

With this choice, it is clear that the set $\tilde{\mathcal{D}}$ of elements classified in class 0 by the ICNN is a convex subset of $\mathbb{R}^n$, as soon as $g(x, \theta)$ is a convex function of x [Boyd and Vandenberghe, 2004, p. 75].

However, it is not is not necessarily the case. Indeed, $g(x, \theta)$ is a difference of two convex functions $h_1(x, \theta)$ and $h_0(x, \theta)$ and as such is not necessarily convex. In order to ensure this, we use identity activation functions for the output layer ($g_{k-1}(z) = z$) and impose an additional constraint on the parameter vectors $W_{k-1}^{z,0}$ and $W_{k-1}^{z,1}$ feeding the output layer of the ICNN: they should satisfy component-wise the inequality[2]

$$W_{k-1}^{z,1} \geq W_{k-1}^{z,0}. \tag{8.1}$$

[2]Note that if this condition is satisfied, $W_{k-1}^{z,0}$ and $W_{k-1}^{z,1}$ do not need to be non-negative for $\tilde{\mathcal{D}}$ to be convex. Nevertheless, we kept this non-negativity condition in the illustrations section 8.3.

With these conditions,

$$\begin{aligned} g(x,\theta) &= h_1(x,\theta) - h_0(x,\theta) \\ &= (W^{z,1}_{k-1} - W^{z,0}_{k-1}) \times z_{k-1} + (W^{z,1}_{k-1} - W^{x,0}_{k-1}) \times x + b^1_{k-1} - b^0_{k-1} \end{aligned}$$

is convex in x since it is the sum of a nonnegative weighted sum of convex functions of x and an affine function of x.

Notice that if we feed such a network with an extended vector $x_e = (x, -x)$ (or, more generally $x_e = Ax$), the input-output relationship remains convex in x.

8.2.2 Building a family of nested convex sets

To build a convex approximation of a set $\mathcal{D} \subset \mathbb{R}^n$, we assume that we have (or that we can build) a dataset of input-output pairs $\hat{\mathcal{D}} = \{(x^i, y^i)\}_{i=1}^n$, where each input x^i describes the coordinates of a point in $\mathbb{R}^n$ and where the corresponding output $y^i = 0$ if the point x^i belongs to $\mathcal{D}$ and $y^i = 1$ otherwise.

We propose to learn from the dataset $\hat{\mathcal{D}}$ the parameters θ of an ICNN classifier which has as inputs $x_e = (x, -x)$, and by using the cross-entropy loss

$$\text{loss}(\theta, x, y) = -\log\left(\frac{\exp(h_y(\theta, x))}{\exp(h_0(\theta, x)) + \exp(h_1(\theta, x))}\right).$$

After training, we consider the whole family of (convex) sets

$$\tilde{\mathcal{D}}_\lambda = \{x \in \mathbb{R}^n | g(x,\theta) = h_1(x,\theta) - h_0(x,\theta) \le \lambda\},$$

with $\lambda \in \mathbb{R}$, as candidate convex approximations of $\mathcal{D}$.

8.2.3 Exploitation in the context of optimization

A convex ICNN classifier can be used to approximate the feasible set $\mathcal{D}$ of an optimization problem. If the objective $f(x)$ is convex, the approximated problem

$$\min_{x \in \tilde{\mathcal{D}}_\lambda} f(x), \tag{8.2}$$

is then also convex. Furthermore, if the objective function $f(x)$ is (piecewise) linear (and convex), and if all the activation functions g_i used in the ICNN are piecewise linear, convex, and non-decreasing functions (such as $ReLU(x) = \max(0, x)$, or $leaky - ReLU(x) = \max(0.01x, x)$), we can show that the resulting optimization problem reduces to a linear program (see appendix 8.A for the proof). In general, we can use convex ICNN classifiers to approximate in a linear fashion any non-convex part of the constraints and/or objective function of any optimization problem. Solving (8.2) for an increasing sequence of λ values, would yield a decreasing sequence of optimal values.

8.3 Illustrations

In this section, we consider some toy problems where $\mathcal{D} \subset \mathbb{R}^2$ to illustrate our approach. Considering the application of this method for optimization problems, we modify the threshold λ and the training loss to obtain inner or outer approximations of the initial set $\mathcal{D}$ and we incorporate the objective function in the training loss in order to guide the model, so that it better approximates the regions close to the minimum of the objective function.

8.3.1 Approximating convex and non-convex regions in $\mathbb{R}^2$

We consider some (convex and non-convex) toy problems where $\mathcal{D} \subset \mathbb{R}^2$. For each one, we used a dataset of 20,000 labelled points. These points were sampled uniformly in a square of length 10 centered at $(0, 0)$; 16,000 were used to train and 4,000 to test.

We show results with ICNNs of 6 hidden layers and 50 neurons per layer, and ReLU activations. The ADAM optimizer [Kingma and Ba, 2014] with a learning rate of 10^{-3} was used to update the network parameters at each epoch. To enforce the non-negativity condition for the weights W^z, each negative element of the update computed with the optimizer is set to 0 before the next iteration. Similarly, to enforce the convexity constraint on the last layer, we set to 0 each pair of weights for which the convexity condition (8.1) is not met. Before training, the inputs are standardized based on their minimum and maximum values in the training set to be in the range $[0, 1]$.

Fig. 8.2 shows four regions $\mathcal{D}$ and their approximation $\tilde{\mathcal{D}}_0$ with an ICNN. One can see that the convex region is well approximated by the ICNN. For the non-convex domains, the ICNN provides a convex approximation $\tilde{\mathcal{D}}_0$ of $\mathcal{D}$.

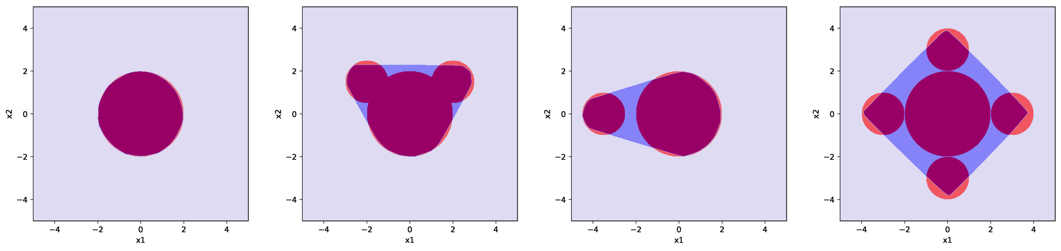
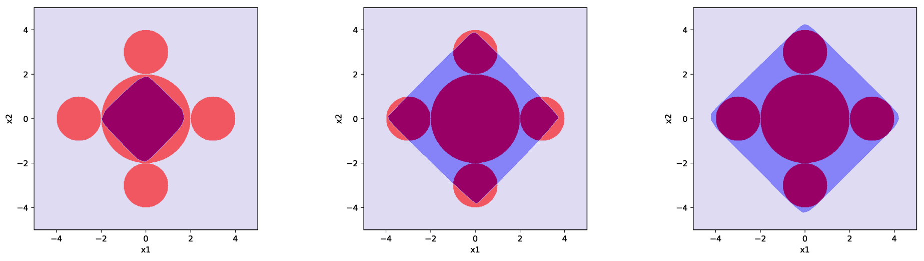


Figure 8.2: The set $\mathcal{D}$ is represented in red and the approximated set $\tilde{\mathcal{D}}_0$ in blue.

8.3.2 Modifying λ

It is possible to play with the size of the approximated region by modifying the threshold λ in the definition $\tilde{\mathcal{D}}_\lambda = \{x \in \mathbb{R}^n | h_1(x) - h_0(x) \leq \lambda\}$.

Fig. 8.3 shows the approximated region $\tilde{\mathcal{D}}_\lambda$ for various λ. We see that increasing the threshold allows us to find outer approximations of $\mathcal{D}$ and decreasing the threshold allows us to find inner approximations. With this method, the model needs only to be learnt once and then the threshold can be manually adjusted to obtain inner or outer approximations.



(a) $\lambda = -1.75$ (b) $\lambda = 0$ (c) $\lambda = 2.7$

Figure 8.3: Effect of λ on the $\tilde{\mathcal{D}}_\lambda$ set.

8.3.3 Considering the class weights in the loss function

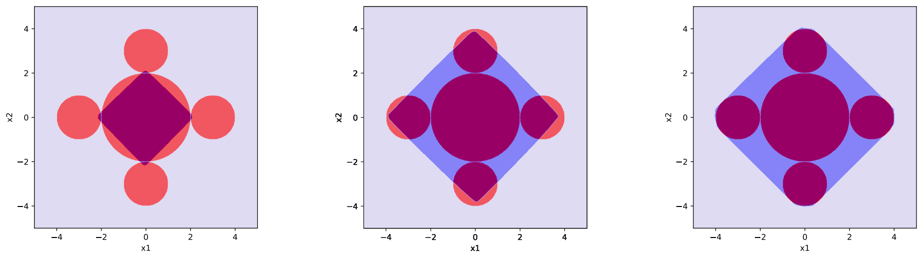
In order to improve the prediction of one class over the other, it is possible to multiply the loss of all observations when training the classifier by their class coefficient w_0 or w_1, in such a way that the loss of an observation x of class y becomes

$$loss(\theta, x, y) = w_y \left[-\log \left(\frac{\exp(h_y(\theta, x))}{\exp(h_0(\theta, x)) + \exp(h_1(\theta, x))} \right) \right].$$

If w_0 is much larger than w_1, the prediction error for observations of class 1 ($x \notin \mathcal{D}$) will be less penalized during training, leading to larger regions $\tilde{\mathcal{D}}_\lambda$. On the contrary, if w_1 is much larger than w_0, the regions $\tilde{\mathcal{D}}_\lambda$ will be smaller.

We carried out some experiments where we vary the weights w_0 and w_1 in order to modify the size of the approximated region. The results can be seen in Figure 8.4. We see that with $w_0 \gg w_1$, almost all points in $\mathcal{D}$ are correctly classified and therefore belong to $\tilde{\mathcal{D}}_0$, such that $\tilde{\mathcal{D}}_0$ almost corresponds to an outer convex approximation. On the contrary, when $w_1 \gg w_0$, almost all points not in $\mathcal{D}$ are correctly classified and $\tilde{\mathcal{D}}_0$ almost corresponds to an inner convex approximation of $\mathcal{D}$.

Note that both this method and the previous one (modifying λ) can be combined to yield inner or outer approximations of $\mathcal{D}$.



(a) $w_0 = 0.1$, $w_1 = 0.9$ (b) $w_0 = 1$, $w_1 = 1$ (c) $w_0 = 0.9$, $w_1 = 0.1$

Figure 8.4: Effect of w_0 and w_1 on the $\tilde{\mathcal{D}}_0$ set.

8.3.4 Considering an objective function when learning the ICNN

In an optimization context, it is of interest to guide the learning of the ICNN in order to improve the approximation close to the minimum values of the objective function. For that, one possibility is to exploit the objective function $f(x)$ in the definition of the loss function used when training the classifier, by giving less weight to elements of the training set farther from the optimum, and thus induce the learnt approximation $\tilde{\mathcal{D}}_0$ to be tighter near to the sought optimum.

In order to implement this, we assume that for each element i of the training set, we also know the value $f_i = f(x_i)$ of the objective function (in addition to the input x_i and the output y_i indicating constraint satisfaction of x_i w.r.t. $\mathcal{D}$).

In order to force the learnt approximation $\tilde{\mathcal{D}}_0$ to be better in regions where the constrained optimum might be located, we give a higher weight in the loss function to input-output samples with smaller associated values of f. Thus, the loss function for an observation x of class y and corresponding to an objective value f would be given by:

$$\text{loss}(\theta, x, y, f) = w_{y,f}\left[-\log\left(\frac{\exp(h_y(\theta, x))}{\exp(h_0(\theta, x)) + \exp(h_1(\theta, x))}\right)\right],$$

where $w_{y,f}$ would depend both on the true class of the sample and on the value of the objective function for this sample.

We tested two methods to compute the weights $w_{y,f}$.

The first method is such that the weights vary linearly between two bounds with the objective function:

$$w_{y,f} = w_y^{\min} + \frac{f_y^{\max} - f}{f_y^{\max} - f_y^{\min}}(w_y^{\max} - w_y^{\min}),$$

where $f_y^{\max}$ and $f_y^{\min}$ are respectively the maximum and minimum values of the objective function among the training samples with label y, and $w_y^{\min}$ and $w_y^{\max}$ for $y \in \{0, 1\}$ are four hyper-parameters that can be tuned.

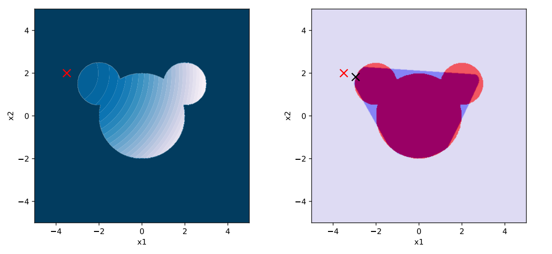
The other method that we tested is such that the weights $w_{y,f}$ take only two values per class y, a high value when the (x, y, f) tuple corresponds to a “small enough" value of the objective function $f(x)$, and a lower value otherwise:

$$w_{y,f} = \mathbb{1}_{A_y}(f) w_y^{\max} + (1 - \mathbb{1}_{A_y}(f)) w_y^{\min},$$

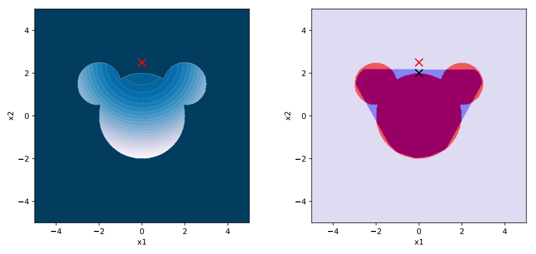
where $w_y^{\max}$ and $w_y^{\min}$ are the two values the weights of the training samples with label y can take and $\mathbb{1}_{A_y}(\cdot)$ is the indicator function indicating if the sample corresponds to a “small enough" value of f for class y. On can imagine various possibilities to define each one of the two sets A_y so that it focuses on the “small enough" f-values of the corresponding class y.

We realized some experiments with the two methods. In these computations, $w_0^{\max} = 1$ and $w_0^{\min} = 0.2$. On the other hand, for class 1, we used $w_1^{\max} = w_1^{\min} = 1$ so that, for the second method, the choice of the set A_1 has no impact in this case. The set A_0 (of elements of class 0 with “small enough" f) was defined so as to contain the 10% training elements of class 0 ($x \in \mathcal{D}$) showing the smallest values of f.

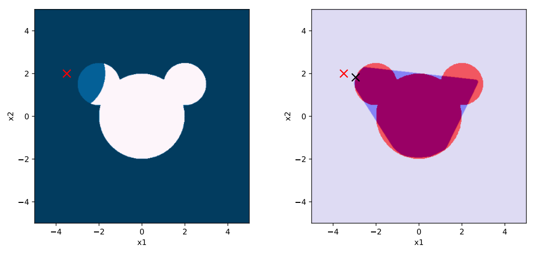
Figure 8.5 shows decision boundaries thus obtained for different objective functions and the two methods proposed to compute $w_{y,f}$. We see that the approximation is indeed tighter close to the unconstrained optimum and actually approaches the constrained optimum very well. The binary weights method seems to be slightly better than the linear weights method. Indeed, we see in Figure 8.5b that, contrarily to the other examples, we would not find the constrained minimum of $f(x)$ if we minimize $f(x)$ over the approximated set $\tilde{\mathcal{D}}_0$.



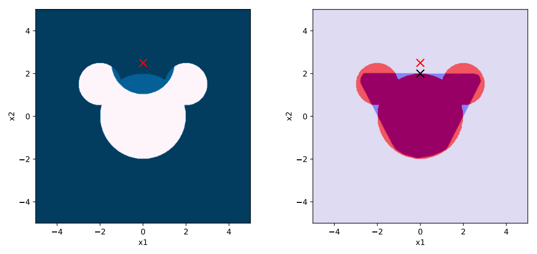
(a) Linear $w_{y,f}$, $f(x) = (x_1 + 3.5)^2 + (x_2 - 2)^2$



(b) Linear $w_{y,f}$, $f(x) = (x_1)^2 + (x_2 - 2.5)^2$



(c) Binary $w_{y,f}$, $f(x) = (x_1 + 3.5)^2 + (x_2 - 2)^2$



(d) Binary $w_{y,f}$, $f(x) = (x_1)^2 + (x_2 - 2.5)^2$

Figure 8.5: Left parts: (a) and (b): training weights $w_{0,f}$ vary linearly between $w_{0,f} = 1$ (dark blue) and $w_{0,f} = 0.2$ (white), $w_{1,f} = 1$; (c) and (d): training weights $w_{0,f}$ take only two values: $w_{0,f} = 1$ (dark blue) and $w_{0,f} = 0.2$ (white), $w_{1,f} = 1$. Right parts: $\mathcal{D}$ and approximated set $\tilde{\mathcal{D}}_0$ learnt while considering $f(x)$. Red and black crosses indicate respectively the unconstrained and the constrained minimum of $f(x)$.

8.3.5 Impact of the size of the learning set and the network

For the previous illustrations, we considered large datasets and networks given the size of the toy problems. In this section, we reduce both the learning set size and the network size for one of our 2-D example problems to verify that the method is still valid in less favorable conditions. Similarly to Figure 8.2, there is no weighting of the loss function. Figure 8.6 shows the resulting approximation $\tilde{\mathcal{D}}_0$ for various combinations (n, n_{hl}, n_n), where n is the size of the learning set, n_{hl} the number of hidden layers and n_n the number of neurons. The first row of the figure with only one hidden layer is quite disappointing. However, we see that from two hidden layers with a sufficient number of neurons per layer ($n_n \geq 30$) and of training observations ($n \geq 4000$), we already have nice results. Therefore, the large datasets and networks used in the previous sections were indeed not necessary.

8.4 Related works

This work is not the first one to exploit the particular properties of ICNNs in the context of optimization problems. ICNNs have been used for complex physical systems control [Chen et al., 2018a, 2020b; Yang and Bequette, 2021], to learn the objective function of an optimization problem and/or its constraints. In these papers, the considered case studies are convex. Using ICNNs is therefore a way to exploit the prior knowledge of convexity while offering tractable control methods.

Similar to our method, ICNNs are used [Sankaranarayanan and Rengaswamy, 2021] in the context of non-convex optimization. The authors developed an algorithm called the Convex Difference Neural Network that expresses the learnt function as a difference of convex functions, so that they can use Difference of Convex programming techniques in problems where their algorithm has been used to learn objective functions and constraints of optimization problems.

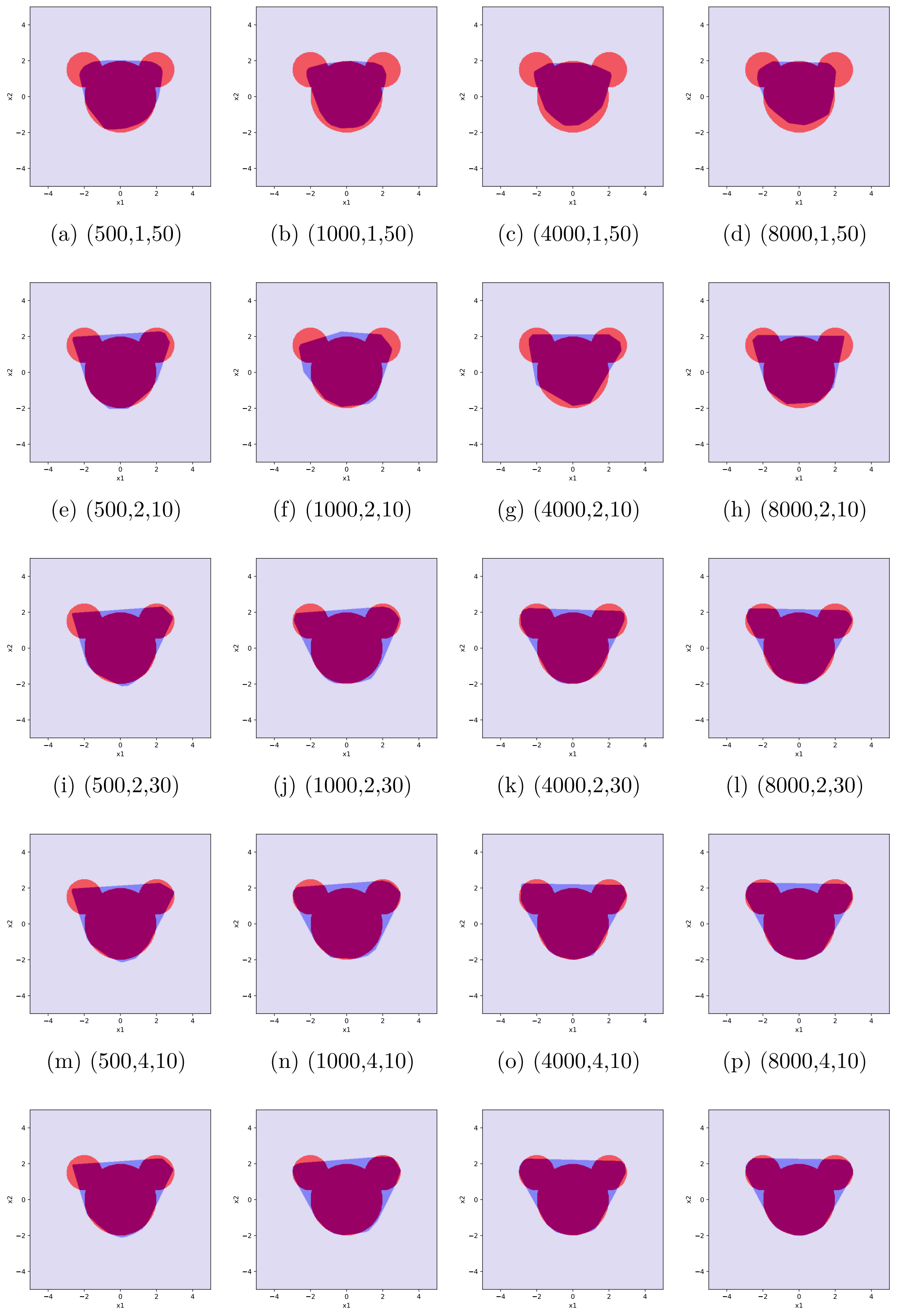
Compared to this research, our method learns convex (actually linear) approximations of non-convex optimization problems, at the cost of possibly larger approximation errors but with the advantage that much more efficient and scalable optimization solvers can be used.

8.5 Discussions and future works

We propose to use ICNNs to learn convex approximations of feasible sets of general optimization problems. This approximation reduces to a series of linear inequalities when ReLU activation functions are used, and we showed how the model may yield outer or inner approximations and/or tight approximations in regions near the optimum of a given objective function.

Compared to a supervised learning method where the optimal solution of an optimization problem is directly learnt, this approach may offer more flexibility. Once the approximation $\tilde{\mathcal{D}}_\lambda$ is learnt, it is possible to modify the objective function or add new (linear) constraints to the problem without requiring to re-train the learnt model. This method could also be used to learn a subset of constraints that could be easily plugged in another optimization problem as linear constraints.

The next step is to test this method on practical non-convex optimization problems. Depending on the context in which this method could be applied, the found solution could be used directly or as a warm-start point for solving the non-convex problem. It can also be used to compute lower or upper bounds



(a) (500,1,50) (b) (1000,1,50) (c) (4000,1,50) (d) (8000,1,50)

(e) (500,2,10) (f) (1000,2,10) (g) (4000,2,10) (h) (8000,2,10)

(i) (500,2,30) (j) (1000,2,30) (k) (4000,2,30) (l) (8000,2,30)

(m) (500,4,10) (n) (1000,4,10) (o) (4000,4,10) (p) (8000,4,10)

(q) (500,4,30) (r) (1000,4,30) (s) (4000,4,30) (t) (8000,4,30)

Figure 8.6: Impact of the size of the learning set and size of the network. The legends (n, n_{hl}, n_n) indicate respectively the size of the learning set n, the number of hidden layers n_{hl} and the number of neurons per layer n_n. Each column corresponds to a different size of the learning set and each row to a different size of network.

of the optimal solution when the learnt approximation is built so as to obtain an inner or an outer approximation[3] of the feasible region.

Another direction of research is to consider the case where the feasible set $\mathcal{D}$ to be approximated depends on some external parameters ξ. In electric power systems, for instance, the secure region of operation depends on load and renewable generation levels and so ξ could represent the realizations of these exogenous uncertainties. One could then learn a convex approximation of the parameterized domain $\mathcal{D}(\xi)$ with an ICNN that would have both x and ξ as inputs while being constrained to be convex only in x. Indeed, in [Amos et al., 2017], such models are proposed as well and they are called Partially Input Convex Neural Network (PICNN). If the PICNN is able to capture the relationship between the shape of the feasible region and the parameters ξ, then it could largely speed up the solving of this type of problems with varying ξ values, once it is learnt.

This PICNN model could be useful in a power systems reliability management context, to learn piecewise linear approximations of the security region of power systems with such a shape that they could easily be plugged in optimization problems, for instance for reliability control.

A further direction of research would consider distributed optimization problems, where the proposed approach could be used to enable various agents to learn and exchange convex approximations of their subsets of constraints and their sub-objectives. This would be of extremely high relevance to the field of multi-area electric power systems planning and operation. For instance, in a multi-TSO context, each operator could learn convex linear approximations of their system security constraints and share these constraints with the other operators in order for the final optimization problem to consider the individual constraints of each operator while still being tractable.

[3]One can argue that these outer and inner approximations of the feasible region being learnt with a supervised learning method, there is no absolute guarantee that they are indeed inner or outer approximations if the dataset $\hat{\mathcal{D}}$ is finite. However, by exploiting an independent sample to verify this assertion, we can have probabilistic guarantees.

Appendix

8.A Proof: $\min_{x\in\tilde{\mathcal{D}}_\lambda} f(x)$ can be reduced to a linear program

We consider a convex ICNN classifier used to approximate the feasible set $\mathcal{D}$ of an optimization problem. Let us show that if the objective function $f(x)$ is piecewise linear and convex, and if all the activation functions g_i used in the ICNN are piecewise linear, convex, and non-decreasing functions (such as $\mathit{ReLU}(x) = \max(0, x)$, or $\mathit{leaky} - \mathit{ReLU}(x) = \max(0.01x, x)$), the resulting optimization problem

$$\min f(x) \text{ s.t. } x \in \tilde{\mathcal{D}}_\lambda,$$

reduces to a linear program. We first notice that if $f(x)$ is piecewise linear and convex, then

$$\min f(x) \text{ s.t. } x \in \tilde{\mathcal{D}}_\lambda$$

may also be rewritten as

$$\min z \text{ s.t. } z \geq a_j^T x + b_j, \forall j = 1, \ldots, l; x \in \tilde{\mathcal{D}}_\lambda,$$

where the set of inequalities $z \geq a_j^T x + b_j, \forall j = 1, \ldots, l$ represents the epigraph of $f(x)$. We thus need only to prove that $\tilde{\mathcal{D}}_\lambda$ may itself also be represented by a set of linear (in)equalities.

For simplicity, we prove this in the case of ReLU activation functions but the result can be extended to any other kind of piecewise linear, convex and non-decreasing activation functions.

First, let us consider the domain P which is the set of points $(x, z_0, \ldots, z_{k-1})$ such that

$$z_0 = 0, \tag{8.3}$$

and

$$\forall i = 0, \ldots k-2 : z_{i+1} = \max(W_i^z \times z_i + W_i^x \times x + b_i, 0) \tag{8.4}$$

and

$$(W_{k-1}^{z,1} - W_{k-1}^{z,0}) \times z_{k-1} + (W_{k-1}^{z,1} - W_{k-1}^{x,0}) \times x + b_{k-1}^1 - b_{k-1}^0 \leq \lambda. \tag{8.5}$$

Note that the projection on x of the domain P, $\mathit{proj}_x(P)$, corresponds to $\tilde{\mathcal{D}}_\lambda$. When the max function is replaced with a set of equations, the equivalent formulation of the constraints in P becomes

$$(W_{k-1}^{z,1} - W_{k-1}^{z,0}) \times z_{k-1} + (W_{k-1}^{z,1} - W_{k-1}^{x,0}) \times x + b_{k-1}^1 - b_{k-1}^0 \leq \lambda, \tag{8.6}$$

$$z_{i+1} = 0 + s_i^0 \qquad \text{for } i = 0, \ldots, k-2 \tag{8.7}$$

$$z_{i+1} = W_i^z \times z_i + W_i^x \times x + b_i + s_i^z \qquad \text{for } i = 0, \ldots, k-2 \tag{8.8}$$

$$s_i^0 s_i^z = 0 \qquad \text{for } i = 0, \ldots, k-2 \tag{8.9}$$
$$s_i^0, s_i^z \geq 0 \qquad \text{for } i = 0, \ldots, k-2 \tag{8.10}$$
$$z_0 = 0, \tag{8.11}$$

where we introduce slack variables $s_i^\cdot$ to express that z_{i+1} is either equal to 0 or to $W_i^z \times z_i + W_i^x \times x + b_i$ for $i = 0, \ldots, k-2$.

All the constraints in P are linear, except equation (8.9). However, we can show that this non-linear equation is not necessary regarding our purpose because the projection on x of a relaxed version of the domain P, that we call Q and for which all the constraints defining Q correspond to a set of linear equations, is also equal to $\tilde{\mathcal{D}}_\lambda$.

Lemma 8.1. *We are given the parameters of an ICNN using ReLU as activation functions and learnt to build a convex approximation $\tilde{\mathcal{D}}_\lambda$ of the feasible set $\mathcal{D}$. Consider the set P defined as*

$$\begin{aligned} P = \Big\{ &(x, z_1, \ldots, z_{k-1}, s_0^0, \ldots, s_{k-2}^0, s_0^z, \ldots, s_{k-2}^z) | \\ &(W_{k-1}^{z,1} - W_{k-1}^{z,0}) \times z_{k-1} + (W_{k-1}^{z,1} - W_{k-1}^{x,0}) \times x + b_{k-1}^1 - b_{k-1}^0 \leq \lambda, && (8.12)\\ &z_{i+1} = 0 + s_i^0 \qquad \forall i = 0, \ldots, k-2, && (8.13)\\ &z_{i+1} = W_i^z \times z_i + W_i^x \times x + b_i + s_i^z \qquad \forall i = 0, \ldots, k-2, && (8.14)\\ &s_i^0 s_i^z = 0 \qquad \forall i = 0, \ldots, k-2, && (8.15)\\ &s_i^0, s_i^z \geq 0 \qquad \forall i = 0, \ldots, k-2, && (8.16)\\ &z_0 = 0 \Big\}. && (8.17) \end{aligned}$$

The set Q, which is defined as P but where constraint (8.15) is removed, i.e. defined as

$$\begin{aligned} Q = \{ &(x, z_1, \ldots, z_{k-1}, s_0^0, \ldots, s_{k-2}^0, s_0^z, \ldots, s_{k-2}^z) | \\ &(W_{k-1}^{z,1} - W_{k-1}^{z,0}) \times z_{k-1} + (W_{k-1}^{z,1} - W_{k-1}^{x,0}) \times x + b_{k-1}^1 - b_{k-1}^0 \leq \lambda, && (8.18)\\ &z_{i+1} = 0 + s_i^0 \qquad \forall i = 0, \ldots, k-2, && (8.19)\\ &z_{i+1} = W_i^z \times z_i + W_i^x \times x + b_i + s_i^z \qquad \forall i = 0, \ldots, k-2, && (8.20)\\ &s_i^0, s_i^z \geq 0 \qquad \forall i = 0, \ldots, k-2, && (8.21)\\ &z_0 = 0 \}, && (8.22) \end{aligned}$$

is therefore a relaxation of P. We now prove that the projection on x of the set P is equal to the projection on x of the set Q:

$$proj_x(P) = proj_x(Q). \tag{8.23}$$

Proof. (a) $proj_x(P) \subseteq proj_x(Q)$ is obvious since Q is a relaxation of P.

(b) We now prove $proj_x(P) \supseteq proj_x(Q)$. Consider $(x, z, s^0, s^z) \in Q$. If $(x, z, s^0, s^z) \in P$, the result follows. Assume now that $(x, z, s^0, s^z) \notin P$. It is always possible, by following Algorithm 8.1, to build from a set of points (x, z, s^0, s^z) in Q but not in P, a set of points $(x, \bar{z}, \bar{s}^0, \bar{s}^z)$ in P with the same x, which proves the result.

Algorithm 8.1: Update $(x, z, s^0, s^z) \in Q$ such that $(x, \bar{z}, \bar{s}^0, \bar{s}^z) \in P$.

```
Result: $(x, \bar{z}, \bar{s}^0, \bar{s}^z)$ in $P$
// Initialization
$(x, \bar{z}, \bar{s}^0, \bar{s}^z) = (x, z_1, \dots, z_{k-1}, s^0_0, \dots, s^0_{k-2}, s^z_0, \dots, s^z_{k-2})$ ;
for $j = 0 : k-2$ do
    if $\bar{s}^0_j > 0$ and $\bar{s}^z_j > 0$ then
        $\Delta := \min(\bar{s}^0_j, \bar{s}^z_j)$;
        $\hat{s}^0_j = \bar{s}^0_j - \Delta$;
        $\hat{s}^z_j = \bar{s}^z_j - \Delta$;
        $\hat{z}_{j+1} = 0 + \hat{s}^0_j$ ;
        // or $\hat{z}_{j+1} = W^z_j \times \bar{z}_j + W^x_j \times x + b_j + \hat{s}^z_j$
        if $j < k-2$ then
            $\hat{s}^z_{j+1} = \bar{s}^z_{j+1} + W^z_{j+1}(\bar{z}_{j+1} - \hat{z}_{j+1})$;
            $(\bar{z}_{j+1}, \bar{s}^0_j, \bar{s}^z_j, \bar{s}^z_{j+1}) = (\hat{z}_{j+1}, \hat{s}^0_j, \hat{s}^z_j, \hat{s}^z_{j+1})$;
        else
            $(\bar{z}_{j+1}, \bar{s}^0_j, \bar{s}^z_j) = (\hat{z}_{j+1}, \hat{s}^0_j, \hat{s}^z_j)$;
        end
    end
end
```

Let us show that $(x, \bar{z}, \bar{s}^0, \bar{s}^z)$, built from $(x, z, s^0, s^z) \in Q$ with Algorithm 8.1, belongs to P. For that we proceed iteratively and we show that at each iteration, the updated vector still belongs to Q. At the end of the iterations, since by construction $(x, \bar{z}, \bar{s}^0, \bar{s}^z)$ satisfies constraint (8.15), $(x, \bar{z}, \bar{s}^0, \bar{s}^z) \in P$.

Let j be the smallest index such that $s^0_j > 0$ and $s^z_j > 0$.

Consider the point $(x, \bar{z}, \bar{s}^0, \bar{s}^z)$, obtained after j iterations with Algorithm 8.1. We can readily see that this point belongs to Q. Indeed, compared to the point $(x, z, s^0, s^z) \in Q$, the only constraint with a different realization of the left-hand-side or right-hand-side is constraint (8.20) for $i = j$ and $i = j+1$. The constraint is nevertheless still valid in both cases:

- $i = j$: By definition of $\bar{z}_{j+1}$, the constraint holds with equality.

- $i = j+1$: We have that $\bar{z}_{j+1} < z_{j+1}$ since $\bar{s}^{\cdot}_j < s^{\cdot}_j$ by construction. Furthermore, given that $W^z_{j+1} > 0$, $W^z_{j+1} \times z_{j+1} > W^z_{j+1} \times \bar{z}_{j+1}$. Therefore,

$$z_{j+2} \geq W^z_{j+1} \times z_{j+1} + W^x_{j+1} \times x + b_{j+1} > W^z_{j+1} \times \bar{z}_{j+1} + W^x_{j+1} \times x + b_{j+1}$$

and the constraint holds.

Note that in case $j = k-2$, the right-hand-side of constraint (8.18) is also impacted. However, since $(W^{z,1}_{k-1} - W^{z,0}_{k-1}) \geq 0$,

$$(W^{z,1}_{k-1} - W^{z,0}_{k-1})\bar{z}_{k-1} < (W^{z,1}_{k-1} - W^{z,0}_{k-1})z_{k-1} \leq \lambda,$$

where the last inequality holds because $(x, z, s^0, s^z) \in Q$. Therefore, $(x, \bar{z}, \bar{s}^0, \bar{s}^z) \in Q$.

Observe that, if $(x, \bar{z}, \bar{s}^0, \bar{s}^z) \notin P$, there exists $\bar{j} > j$ such that $s^0_{\bar{j}} > 0$ and $s^z_{\bar{j}} > 0$. We can again decrease the value of $z_{\bar{j}+1}$ in order to make one of the slacks tight. We obtain the result by applying the procedure repeatedly. Since there is a finite number of indices, the procedure can be applied at most $k-1$ times until we obtain $(x, \bar{z}, \bar{s}^0, \bar{s}^z) \in P$.

We can thus state that $proj_x(P) = proj_x(Q)$. □

Note that this lemma can be extended to other activation functions, as long as they are convex, piecewise linear and non-decreasing. It can thus be applied to leaky-ReLU activation functions. It is also still valid at the limit, for an infinite number of pieces and so it is valid for any smooth convex and non-decreasing activation function.

Consequently to Lemma 8.1, given an objective function that only depends on x, we have the following result.

Corollary 8.2. *Given an ICNN using ReLU as activation functions and learnt to build a convex approximation $\tilde{\mathcal{D}}_\lambda$ of the domain $\mathcal{D}$,*

$$\begin{aligned} &\min f(x) \\ s.t.\ &(W^{z,1}_{k-1} - W^{z,0}_{k-1}) \times z_{k-1} + (W^{z,1}_{k-1} - W^{x,0}_{k-1}) \times x + b^1_{k-1} - b^0_{k-1} \leq \lambda \\ &z_{i+1} = \max(W^z_i \times z_i + W^x_i \times x + b_i, 0) \qquad \textit{for } i = 0, \dots, k-2 \\ &z_0 = 0 \end{aligned} \tag{8.24}$$

is equivalent to

$$\begin{aligned} &\min f(x) \\ s.t.\ &(W^{z,1}_{k-1} - W^{z,0}_{k-1}) \times z_{k-1} + (W^{z,1}_{k-1} - W^{x,0}_{k-1}) \times x + b^1_{k-1} - b^0_{k-1} \leq \lambda \\ &z_{i+1} \geq 0 \qquad \textit{for } i = 0, \dots, k-2 \\ &z_{i+1} \geq W^z_i \times z_i + W^x_i \times x + b_i \qquad \textit{for } i = 0, \dots, k-2 \\ &z_0 = 0. \end{aligned} \tag{8.25}$$

Indeed, the domain P is equivalent to the feasible set of problem (8.24) while Q is equivalent to the feasible set of problem (8.25). Since the projection of these two sets on the x space is equivalent, *i.e.* $proj_x(P) = proj_x(Q)$, then the feasible set of solutions regarding x and thus the optimal solution x^* of the two optimization problems are the same.

Therefore, $\min_{x \in \tilde{\mathcal{D}}_\lambda} f(x)$ reduces to a linear program if the objective function $f(x)$ is (piecewise) linear (and convex), and if all the activation functions g_i used in the ICNN are piecewise linear, convex, and non-decreasing functions.

Part IV

Conclusions and further works

9

Conclusions and perspectives

Overview

This chapter concludes the thesis and identifies some possible directions of future research in the topic of machine learning for reliability management.

9.1 Main findings

In this thesis, we proposed to exploit machine learning to develop decision-making tools for reliability management in short-term operation planning, in order to take into account the increasing uncertainties in power systems.

To consider these uncertainties, we chose the approach of modeling the reliability management strategy over many future possible operating conditions, instead of one (or a few) most likely scenario. The main drawback of this approach is the computational burden linked to modeling the response of the operator to realizations of uncertainties over the time horizon considered. Optimization tools such as OPF or SCOPF are often used but they are large-scale non-linear non-convex optimization problems, with large computation cost to be solved.

To overcome this drawback, we proposed in this dissertation to exploit machine learning to build fast and accurate simplified models, called *proxies*, of the response of the operator to realizations of uncertainties along the time horizon of consideration. The concept of proxies and how they are integrated in power systems reliability management is well illustrated by the Russian dolls presented in the European project GARPUR [GARPUR Consortium, 2016] and schematized in Figure 9.1 for day-ahead operation planning. It works as follows. The first Russian doll corresponds to the studied context (here operation planning) and then each following doll represents a proxy of the shorter-term reliability management decision-making contexts and is entangled in the previous context. The level of simplification of the proxies should be tailored to the need of the studied context, considering both the time available and the necessary precision of the modeling. There can be as many dolls as is necessary to model the reliability management strategy over the future time horizon and, generally, the further we are from the studied context, the faster the proxy should be, often at the cost of a loss in precision.

We showcased our approach on the day-ahead operation planning context, while modeling the real-time reliability management response of the operator to uncertainties, without loss of generality.

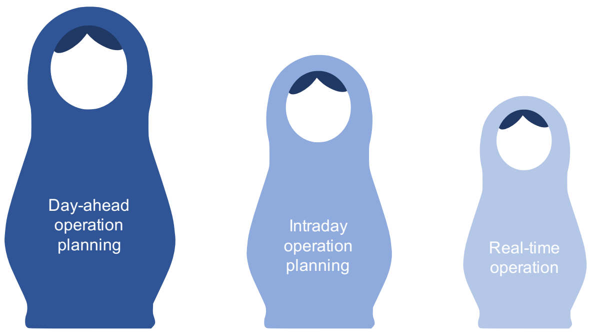


Figure 9.1: Russian dolls illustrating the entangling of models of intraday and real-time reliability management in day-ahead operation planning to ensure coherence between contexts.

In particular we developed a methodology to automatically build these proxies of real-time operation with supervised learning and we showed that the proxies are several orders of magnitude faster than the traditional model of real-time reliability management operation.

We then proposed a method to exploit these proxies for probabilistic reliability assessment in operation planning. More accurately, we proposed a methodology based on a variance reduction technique called the control variates approach combined with our proxies to speed up the estimation of the expected cost of real-time operation for a given candidate operation planning decision. We showed on a case study that this method allows to significantly reduce the number of scenarios necessary to obtain an estimation of the induced expected costs of real-time operation of the given candidate operation planning decision, without loss of accuracy compared to the traditional crude Monte-Carlo approach.

We generalized this methodology over new, unseen candidate operation planning decisions and tested the usefulness of this methodology to rank a list of candidate decisions according to their expected costs of real-time operation. We showed on a case study that this methodology can indeed identify the decisions with the lowest expected costs and can therefore be used to select 'good' operation planning decisions.

Slightly different from our other contributions, we proposed to exploit a supervised learning algorithm, the ICNN, to learn convex approximations of feasible sets of optimization problems. We showed that, in the context of optimization problems, this method can be used to approximate a non-convex feasible domain with a series of linear constraints.

Finally, we also performed a survey of the recent works applying machine learning in the context of reliability management. We highlighted that more and more papers are published each year in this topic. Although at the beginning these papers were mostly dealing with reliability assessment, new applications are being studied, such as using deep learning to predict power flows, predicting the outcomes of OPF with simplified models, learning features of OPF or UC problems, and embedding machine learnt security rules in their formulations.

To conclude, we believe that the use of machine learning techniques, and proxies in particular, can help the operator to manage the grid closely to its limits, and thus can be part of the solution to ensure the reliability of the system despite the increasing uncertainties.

9.2 Future works

Although the research about machine learning, and proxies in particular, in power systems reliability management is still at the beginning, the results are promising and therefore we see many directions for further research.

9.2.1 Machine learning in power systems applications

Considering cyber-security With the rise of smart grids, information and communication technologies have been integrated to the grid to help monitor and control the power system in real-time. This opens the system up to new vulnerabilities, such as cyber attacks that could modify the data received by the operators and cause them to take wrong decisions. As a consequence, the information system is becoming as critical as the physical system. The vulnerability of the system is further increased by the development of machine learning models that could be directly impacted by a failure of the communication system or erroneous data inputs. The robustness of these machine learning models against missing or erroneous data should be studied. Furthermore, the vulnerability of machine learning models against man-crafted adversarial data[1] should also be considered [Chen et al., 2018b] and techniques to detect these adversarial examples should be developed.

9.2.2 Proxies in power systems reliability management

Using proxies to evaluate the reliability of a decision In this thesis, we focused on the estimation of real-time operation costs to assess and select suitable operation planning decisions. A next step would be to build proxies evaluating if a given operation planning decision would lead to acceptable trajectories, i.e trajectories for which the reliability criterion is met in real-time. The objective is to be able to evaluate for a given operation planning decision the probability of meeting the reliability criterion in real-time and to allow the operator to do a trade-off between minimizing operation costs and satisfying the reliability criterion. As we expect many trajectories to be acceptable given the very high reliability level of real-life power systems, this problem can be characterized as a rare event problem. Estimating probabilities of occurrence of rare events is a very difficult task as these probabilities are often underes-

[1]In the field of computer vision, it was noticed that small modifications in the input features of a model could lead to very different predicted outputs. These small perturbations can be intentionally added to an instance to force the model to make a false prediction. This is called an adversarial example [Szegedy et al., 2013].

timated with classical machine learning methods and thus specific techniques for rare events will be needed to address this problem.

Building proxies from more realistic models of power systems operation For the case studies, we considered rather simplistic models of day-ahead uncertainties and real-time operation to build our databases used to learn the proxies, for instance by using the DC approximation. It serves as a proof-of-concept, but it is not a limitation for the approach. The proxies could indeed be learnt from any simulator of real-time operation or even from historical observations. It would therefore be interesting to learn proxies with more accurate models of day-ahead uncertainties and of real-time operation, for instance by using an AC-SCOPF, modeling probabilistic reliability criteria instead of the N-1 criterion, modeling other preventive and corrective actions such as topology changes, etc.

Improving the performances of the proxies with more advanced ML techniques With the rapid advances in machine learning, and deep learning in particular, new algorithms and methods are available, that could be used to improve the performances of our proxies.

Practical applications of the ICNN method The method based on ICNNs presented in chapter 8 can be seen as a method to build proxies of feasible regions of optimization problems. It would be interesting to test this method on practical problems, especially for reliability control applications.

Proxies of intraday operation planning and real-time operation for day-ahead operation planning In this work, we considered a two-step process for decision-making: first day-ahead and then real-time. However, in practice this process is more a multi-stage process with decisions taken as well in intraday operation planning [Panciatici et al., 2012]. It would therefore be of interest to also consider the intraday operation planning context and thus combine the real-time and intraday proxies (as suggested in Figure 9.1) for more accurate day-ahead operation planning.

Proxies for other reliability management contexts The methods developed in this manuscript could be used as such or adapted to be used in the other reliability management contexts such as intraday operation planning, asset management and grid development.

Proxies of interconnected systems and sub-systems In this thesis, we did not consider the TSO-DSO and TSO-TSO interactions. The concept of proxies could however be applied, not only to model sequentially other reliability management contexts but also to model smaller sub-systems or other large-scale systems interacting with the managed one, such as distribution grids and other interconnected transmission systems. In these contexts, the method proposed in chapter 8 could for instance be exploited by the operators to compute piecewise linear approximations of their security constraints, that could then be embedded in multi-agent optimization problems to ensure that the variables shared by several operators (agents) satisfy the security constraints of each one of them. Finally, proxies should not be limited to modeling power systems but could also be used in the context of multi-energy systems, to model with the proper level of abstraction other interconnected systems such as gas transportation systems, electric vehicle charging infrastructures, district heating systems, etc.

Bibliography

Ahmed, R., Sreeram, V., Mishra, Y., and Arif, M. (2020). A review and evaluation of the state-of-the-art in PV solar power forecasting: Techniques and optimization. *Renewable and Sustainable Energy Reviews*, 124:109792.

Amodei, D., Olah, C., Steinhardt, J., Christiano, P., Schulman, J., and Mané, D. (2016). Concrete problems in AI safety. *arXiv preprint arXiv:1606.06565*.

Amos, B., Xu, L., and Kolter, J. Z. (2017). Input convex neural networks. In *International Conference on Machine Learning*, pages 146–155. PMLR.

Ardakani, A. J. and Bouffard, F. (2018). Prediction of umbrella constraints. In *2018 Power Systems Computation Conference (PSCC)*, pages 1–7. IEEE.

Baker, K. (2019). Learning warm-start points for AC optimal power low. *Arxiv*.

Baker, K. (2020a). A learning-boosted Quasi-Newton method for AC optimal power flow. *arXiv preprint arXiv:2007.06074*.

Baker, K. (2020b). Emulating ac opf solvers for obtaining sub-second feasible, near-optimal solutions. *arXiv preprint arXiv:2012.10031*.

Battaglini, A., Komendantova, N., Brtnik, P., and Patt, A. (2012). Perception of barriers for expansion of electricity grids in the European Union. *Energy Policy*, 47:254–259.

Bhatt, I., Dhandhia, A., and Pandya, V. (2017). Static security assessment of power system using radial basis function neural network module. In *2017 IEEE International WIE Conference on Electrical and Computer Engineering (WIECON-ECE)*, pages 274–278. IEEE.

Biagioni, D., Graf, P., Zhang, X., Zamzam, A. S., Baker, K., and King, J. (2020). Learning-Accelerated ADMM for Distributed DC Optimal Power Flow. *IEEE Control Systems Letters*.

Biggar, D. R. and Hesamzadeh, M. R. (2014). *The economics of electricity markets*. John Wiley & Sons.

Billinton, R. (1970). *Power System Reliability Evaluation*. Gordon and Breach.

Billinton, R. and Allan, R. N. (1996). *Reliability evaluation of power systems*. Springer, 2nd edition.

Billinton, R. and Wenyuan, L. (1991). A novel method for incorporating weather effects in composite system adequacy evaluation. *Power Systems, IEEE Transactions on*, 6(3):1154–1160.

Bloomberg (2021). The day europe's power grid came close to a massive blackout. Available at https://www.bloomberg.com/news/articles/2021-01-27/green-shift-brings-blackout-risk-to-world-s-biggest-power-grid, accessed on 2021-05-12.

Bludszuweit, H., Domínguez-Navarro, J. A., and Llombart, A. (2008). Statistical analysis of wind power forecast error. *IEEE Transactions on Power Systems*, 23(3):983–991.

Boßmann, T. and Staffell, I. (2015). The shape of future electricity demand: Exploring load curves in 2050s Germany and Britain. *Energy*, 90:1317–1333.

Boyd, S. and Vandenberghe, L. (2004). *Convex optimization*. Cambridge University Press.

Breiman, L. (1996). Bagging predictors. *Machine Learning*, 24(2):123–140.

Breiman, L. (2001). Random forests. *Machine Learning*, 45(1):5–32.

Canyasse, R., Dalal, G., and Mannor, S. (2017). Supervised learning for optimal power flow as a real-time proxy. *2017 IEEE Power and Energy Society Innovative Smart Grid Technologies Conference, ISGT 2017*.

Capitanescu, F. (2016). Critical review of recent advances and further developments needed in AC optimal power flow. *Electric Power Systems Research*, 136:57–68.

Capitanescu, F., Ramos, J. M., Panciatici, P., Kirschen, D., Marcolini, A. M., Platbrood, L., and Wehenkel, L. (2011). State-of-the-art, challenges, and future trends in security constrained optimal power flow. *Electric Power Systems Research*, 81(8):1731–1741.

Carpentier, J. (1962). Contribution to the economic dispatch problem. *Bulletin de la Societe Francoise des Electriciens*, 3(8):431–447.

Caruana, R. (1997). Multitask learning. *Machine learning*, 28(1):41–75.

Chakraborty, K. and Saha, G. (2016). Off-line voltage security assessment of power transmission systems using uvsi through artificial neural network. In *2016 International Conference on Intelligent Control Power and Instrumentation (ICICPI)*, pages 158–162. IEEE.

Chapelle, O., Scholkopf, B., and Zien, A. (2010). *Semi-Supervised Learning*. MIT Press, 1st edition.

Chatzos, M., Fioretto, F., Mak, T. W., and Van Hentenryck, P. (2020). High-fidelity machine learning approximations of large-scale optimal power flow. *arXiv preprint arXiv:2006.16356*.

Chatzos, M., Mak, T. W., and Van Hentenryck, P. (2021). Spatial network decomposition for fast and scalable ac-opf learning. *arXiv preprint arXiv:2101.06768.*

Chen, L. and Tate, J. E. (2020). Hot-starting the ac power flow with convolutional neural networks. *arXiv preprint arXiv:2004.09342.*

Chen, Y., Lakshminarayana, S., Maple, C., and Poor, H. V. (2020a). A meta-learning approach to the optimal power flow problem under topology reconfigurations. *arXiv preprint arXiv:2012.11524.*

Chen, Y., Shi, Y., and Zhang, B. (2018a). Optimal control via neural networks: A convex approach. *arXiv preprint arXiv:1805.11835.*

Chen, Y., Shi, Y., and Zhang, B. (2020b). Data-driven optimal voltage regulation using input convex neural networks. *Electric Power Systems Research*, 189:106741.

Chen, Y., Tan, Y., and Deka, D. (2018b). Is machine learning in power systems vulnerable? In *2018 IEEE International Conference on Communications, Control, and Computing Technologies for Smart Grids (SmartGridComm)*, pages 1–6. IEEE.

Chen, Y. and Zhang, B. (2020). Learning to solve network flow problems via neural decoding. *arXiv preprint arXiv:2002.04091.*

Coffrin, C., Hijazi, H. L., and Van Hentenryck, P. (2015). The QC relaxation: A theoretical and computational study on optimal power flow. *IEEE Transactions on Power Systems*, 31(4):3008–3018.

Colorado, J. D. A., Londoño, S. P., and Flórez, J. J. M. (2016). Identifying voltage stability critical areas using k-means clustering technique. *2016 IEEE PES Transmission and Distribution Conference and Exposition-Latin America, PES T and D-LA 2016*, (1).

Conejo, A. J. and Baringo, L. (2018). *Power system operations.* Springer.

Cremer, J. L., Konstantelos, I., Strbac, G., and Tindemans, S. H. (2018). Sample-derived disjunctive rules for secure power system operation. In *2018 IEEE International Conference on Probabilistic Methods Applied to Power Systems (PMAPS)*, pages 1–6. IEEE.

Cremer, J. L., Konstantelos, I., Tindemans, S. H., and Strbac, G. (2019). Data-driven power system operation: Exploring the balance between cost and risk. *IEEE Transactions on Power Systems*, 34(1):791–801.

Dalal, G., Gilboa, E., Mannor, S., and Wehenkel, L. (2018). Unit commitment using nearest neighbor as a short-term proxy. *20th Power Systems Computation Conference, PSCC 2018.*

Dalal, G., Gilboa, E., Mannor, S., and Wehenkel, L. (2019). Chance-constrained outage scheduling using a machine learning proxy. *IEEE Transactions on Power Systems*, 34(4):2528–2540.

Dalal, G., Gilboa, G., and Mannor, S. (2016). Hierarchical decision making in electricity grid management. *33rd International Conference on Machine Learning, ICML 2016*, 5:3249–3258.

Dalal, G. and Mannor, S. (2015). Reinforcement learning for the unit commitment problem. *2015 IEEE Eindhoven PowerTech, PowerTech 2015*.

Deka, D. and Misra, S. (2019). Learning for DC-OPF: Classifying active sets using neural nets. *Arxiv*, pages 1–6.

Dhandhia, A., Pandya, V., and Bhatt, P. (2019). Multi-class support vector machines for static security assessment of power system. *Ain Shams Engineering Journal*, (xxxx).

Diehl, F. (2019). Warm-starting AC optimal power flow with graph neural networks. In *33rd Conference on Neural Information Processing Systems (NeurIPS 2019)*.

Dietterich, T. G. (2000). An experimental comparison of three methods for constructing ensembles of decision trees: Bagging, boosting, and randomization. *Machine Learning*, 40(2):139–157.

Dobbe, R., Hidalgo-Gonzalez, P., Karagiannopoulos, S., Henriquez-Auba, R., Hug, G., Callaway, D. S., and Tomlin, C. J. (2020). Learning to control in power systems: Design and analysis guidelines for concrete safety problems. *Electric Power Systems Research*, 189:106615.

Dong, W., Xie, Z., Kestor, G., and Li, D. (2020). Smart-PGSim: using neural network to accelerate AC-OPF power grid simulation. *arXiv preprint arXiv:2008.11827*.

Donnot, B., Guyon, I., Marot, A., Schoenauer, M., and Panciatici, P. (2018a). Optimization of computational budget for power system risk assessment. *Proceedings - 2018 IEEE PES Innovative Smart Grid Technologies Conference Europe, ISGT-Europe 2018*, (1):1–6.

Donnot, B., Guyon, I., Schoenauer, M., Marot, A., and Panciatici, P. (2018b). Anticipating contingengies in power grids using fast neural net screening. *Proceedings of the International Joint Conference on Neural Networks*, 2018-July.

Donnot, B., Guyon, I., Schoenauer, M., Marot, A., and Panciatici, P. (April 2018c). Fast power system security analysis with guided dropout. *26th European Symposium on Artificial Neural Networks (ESAAN), Bruges*.

Donnot, B., Guyon, I., Schoenauer, M., Panciatici, P., and Marot, A. (2017). Introducing machine learning for power system operation support. *Arxiv*.

Donon, B., Clément, R., Donnot, B., Marot, A., Guyon, I., and Schoenauer, M. (2020a). Neural networks for power flow: Graph neural solver. *Electric Power Systems Research*, 189:106547.

Donon, B., Donnot, B., Guyon, I., Liu, Z., Marot, A., Panciatici, P., and Schoenauer, M. (2020b). Leap nets for system identification and application to power systems. *Neurocomputing*, 416:316–327.

Du, Y., Li, F., and Huang, C. (2019a). Applying deep convolutional neural network for fast security assessment with n-1 contingency. In *2019 IEEE Power & Energy Society General Meeting (PESGM)*, pages 1–5. IEEE.

Du, Y., Li, F., Li, J., and Zheng, T. (2019b). Achieving 100x acceleration for n-1 contingency screening with uncertain scenarios using deep convolutional neural network. *IEEE Transactions on Power Systems*, 34(4):3303–3305.

Du, Y., Li, F., Zheng, T., and Li, J. (2020). Fast cascading outage screening based on deep convolutional neural network and depth-first search. *IEEE Transactions on Power Systems*, 35(4):2704–2715.

Duchesne, L. (2016). Machine learning of proxies for power systems reliability management. Master's thesis, Université de Liège, Liège, Belgique.

Duchesne, L., Cornélusse, B., and Savelli, I. (2019). Sensitivity analysis of a local market model for community microgrids. In *2019 IEEE Milan PowerTech*, pages 1–6. IEEE.

Duchesne, L., Karangelos, E., Sutera, A., and Wehenkel, L. (2020a). Machine learning for ranking day-ahead decisions in the context of short-term operation planning. *Electric Power Systems Research*, 189:106548.

Duchesne, L., Karangelos, E., and Wehenkel, L. (2017). Machine learning of real-time power systems reliability management response. In *2017 IEEE Manchester PowerTech*, pages 1–6. IEEE.

Duchesne, L., Karangelos, E., and Wehenkel, L. (2018). Using machine learning to enable probabilistic reliability assessment in operation planning. In *2018 Power Systems Computation Conference (PSCC)*, pages 1–8. IEEE.

Duchesne, L., Karangelos, E., and Wehenkel, L. (2020b). Recent developments in machine learning for energy systems reliability management. *Proceedings of the IEEE*, 108(9):1656–1676.

Duchesne, L., Louveaux, Q., and Wehenkel, L. (2021). Supervised learning of convex piece-wise linear approximations of optimization problems. Submitted for publication.

Dvorkin, Y., Pandžić, H., Ortega-Vazquez, M. A., and Kirschen, D. S. (2014). A hybrid stochastic/interval approach to transmission-constrained unit commitment. *IEEE Transactions on Power Systems*, 30(2):621–631.

Dy Liacco, T. E. (1967). The adaptive reliability control system. *IEEE Transactions on Power Apparatus and Systems*, (5):517–531.

Dyner, I. and Larsen, E. R. (2001). From planning to strategy in the electricity industry. *Energy policy*, 29(13):1145–1154.

Elia (2021). Our infrastructure. [Online], Available at https://www.elia.be/en/infrastructure-and-projects/our-infrastructure, accessed on 2021-02-22.

ENTSO-E (2009). P3 – policy 3: Operational security. [Online], Available at https://eepublicdownloads.entsoe.eu/clean-documents/pre2015/publications/entsoe/Operation_Handbook/Policy_3_final.pdf, accessed on 2021-08-17.

European Commission (2017). Commission regulation (eu) 2017/1485 of 2 august 2017 establishing a guideline on electricity transmission system operation.

European commission (2019). Communication from the commission to the european parliament, the european council, the council, the european economic and social committee and the committee of the regions: The European green deal. Available at https://eur-lex.europa.eu/legal-content/EN/TXT/?uri=CELEX%3A52019DC0640&qid=1620809845606.

Eurostat (2021). Share of energy from renewable sources. [Online], Available at https://ec.europa.eu/eurostat/databrowser/product/view/NRG_IND_REN, accessed on 2021-04-14.

Falconer, T. and Mones, L. (2020). Deep learning architectures for inference of ac-opf solutions. *arXiv preprint arXiv:2011.03352*.

Fan, Y., Li, X., and Zhang, P. (2017). Real-time static voltage stability assessment in large-scale power systems based on maximum-relevance minimum-redundancy ensemble approach. *IEEE Access*, 5:27281–27291.

Fink, L. H. and Carlsen, K. (1978). Operating under stress and strain [electrical power systems control under emergency conditions]. *IEEE Spectrum*, 15(3):48–53.

Florita, A., Hodge, B.-M., and Milligan, M. (2012). Wind power forecasting error frequency analyses for operational power system studies. Technical report, National Renewable Energy Lab.(NREL), Golden, CO (United States).

Fotuhi-Firuzabad, M. and Billinton, R. (2000). Impact of load management on composite system reliability evaluation short-term operating benefits. *IEEE Transactions on power systems*, 15(2):858–864.

Fu, J., Abbott, S., Fox, B., Morrow, D., and Abdelkader, S. (2010). Wind cooling effect on dynamic overhead line ratings. In *45th International Universities Power Engineering Conference UPEC2010*, pages 1–6. IEEE.

GAMS (2012). GAMS Development Corporation. General Algebraic Modeling System (GAMS) Release 24.0.2.

GARPUR Consortium (2014). D1. 2, Current practices, drivers and barriers for new reliability standards. 7th framework programme, EU Commission grant agreement 608540. Available at http://www.garpur-project.eu/deliverables.

GARPUR Consortium (2015). D6.1 Functional analysis of System Operation processes. 7th framework programme, EU Commission grant agreement 608540. Available at http://www.garpur-project.eu/deliverables.

GARPUR Consortium (2016). D2. 2, Guidelines for implementing the new reliability assessment and optimization methodology. 7th framework programme, EU Commission grant agreement 608540. Available at http://www.garpur-project.eu/deliverables.

Geurts, P. (2002). *Contributions to decision tree induction: bias/variance tradeoff and time series classification*. PhD thesis, University of Liège, Liège, Belgique.

Geurts, P., Ernst, D., and Wehenkel, L. (2006). Extremely randomized trees. *Machine learning*, 63(1):3–42.

Gholami, M., Gharehpetian, G., and Mohammadi, M. (2016). Online decision tree based strategy for power system static security margin improvement using wind farms. *International Journal of Electrical Power & Energy Systems*, 83:15–20.

Gholami, M., Gharehpetian, G. B., and Mohammadi, M. (2015). Intelligent hierarchical structure of classifiers to assess static security of power system. *Journal of Intelligent and Fuzzy Systems*, 28(6):2875–2880.

Glorot, X., Bordes, A., and Bengio, Y. (2011). Deep sparse rectifier neural networks. In *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, pages 315–323. JMLR Workshop and Conference Proceedings.

Goodfellow, I., Bengio, Y., and Courville, A. (2016). *Deep Learning*. MIT Press. http://www.deeplearningbook.org.

Goodfellow, I. J., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y. (2014). Generative adversarial networks. *arXiv preprint arXiv:1406.2661*.

Grigg, C., Wong, P., Albrecht, P., Allan, R., Bhavaraju, M., Billinton, R., Chen, Q., Fong, C., Haddad, S., Kuruganty, S., et al. (1999). The IEEE reliability test system-1996. a report prepared by the reliability test system task force of the application of probability methods subcommittee. *IEEE Transactions on power systems*, 14(3):1010–1020.

Guha, N., Wang, Z., Wytock, M., and Majumdar, A. (2019). Machine learning for ac optimal power flow. *arXiv preprint arXiv:1910.08842*.

Guyon, I. and Elisseeff, A. (2003). An introduction to variable and feature selection. *Journal of machine learning research*, 3(Mar):1157–1182.

Håberg, M. (2019). Fundamentals and recent developments in stochastic unit commitment. *International Journal of Electrical Power & Energy Systems*, 109:38–48.

Haes Alhelou, H., Hamedani-Golshan, M. E., Njenda, T. C., and Siano, P. (2019). A survey on power system blackout and cascading events: Research motivations and challenges. *Energies*, 12(4):682.

Halilbašić, L., Thams, F., Venzke, A., Chatzivasileiadis, S., and Pinson, P. (2018). Data-driven security-constrained AC-OPF for operations and markets. In *2018 Power Systems Computation Conference (PSCC)*, pages 1–7. IEEE.

Hastie, T., Tibshirani, R., and Friedman, J. (2009). *The elements of statistical learning: data mining, inference, and prediction*. Springer Science & Business Media.

Heylen, E., Ovaere, M., Proost, S., Deconinck, G., and Van Hertem, D. (2019). A multi-dimensional analysis of reliability criteria: From deterministic N-1 to a probabilistic approach. *Electric Power Systems Research*, 167:290–300.

Hodge, B.-M., Florita, A., Orwig, K., Lew, D., and Milligan, M. (2012a). Comparison of wind power and load forecasting error distributions. Technical report, National Renewable Energy Lab.(NREL), Golden, CO (United States).

Hodge, B.-M., Lew, D., Milligan, M., Holttinen, H., Sillanpaa, S., Gómez-Lázaro, E., Scharff, R., Soder, L., Larsén, X. G., Giebel, G., et al. (2012b). Wind power forecasting error distributions: An international comparison. Technical report, National Renewable Energy Lab.(NREL), Golden, CO (United States).

Hong, T. and Fan, S. (2016). Probabilistic electric load forecasting: A tutorial review. *International Journal of Forecasting*, 32(3):914–938.

Hor, C.-L., Watson, S. J., and Majithia, S. (2005). Analyzing the impact of weather variables on monthly electricity demand. *IEEE transactions on power systems*, 20(4):2078–2085.

Hou, Q., Zhang, N., Kirschen, D. S., Du, E., Cheng, Y., and Kang, C. (2020). Sparse oblique decision tree for power system security rules extraction and embedding. *IEEE Transactions on Power Systems*.

Hu, R., Li, Q., and Qiu, F. (2021). Ensemble learning based convex approximation of three-phase power flow. *IEEE Transactions on Power Systems*.

Hu, X., Hu, H., Verma, S., and Zhang, Z.-L. (2020). Physics-guided deep neural networks for power flow analysis. *IEEE Transactions on Power Systems*.

Huang, W., Pan, X., Chen, M., and Low, S. H. (2021). DeepOPF-V: Solving AC-OPF problems efficiently. *arXiv preprint arXiv:2103.11793*.

Jain, P., Batra, V., and Darak, S. J. (2018). Improved hierarchical decision making policy for reliable and green electricity grid. In *2018 10th International Conference on Communication Systems & Networks (COMSNETS)*, pages 450–453. IEEE.

Karangelos, E., Panciatici, P., and Wehenkel, L. (2013). Whither probabilistic security management for real-time operation of power systems? In *2013 IREP Symposium Bulk Power System Dynamics and Control-IX Optimization, Security and Control of the Emerging Power Grid*, pages 1–17. IEEE.

Karangelos, E. and Wehenkel, L. (2016). Probabilistic reliability management approach and criteria for power system real-time operation. In *2016 Power Systems Computation Conference (PSCC)*, pages 1–9. IEEE.

Khuntia, S. R., Tuinema, B. W., Rueda, J. L., and van der Meijden, M. A. (2016). Time-horizons in the planning and operation of transmission networks: an overview. *IET Generation, Transmission & Distribution*, 10(4):841–848.

Kingma, D. P. and Ba, J. (2014). Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Kirschen, D. and Jayaweera, D. (2007). Comparison of risk-based and deterministic security assessments. *IET Generation, Transmission & Distribution*, 1(4):527–533.

Kobyzev, I., Prince, S., and Brubaker, M. (2020). Normalizing flows: An introduction and review of current methods. *IEEE Transactions on Pattern Analysis and Machine Intelligence*.

Konstantelos, I., Sun, M., Tindemans, S. H., Issad, S., Panciatici, P., and Strbac, G. (2019). Using vine copulas to generate representative system states for machine learning. *IEEE Transactions on Power Systems*, 34(1):225–235.

Kundur, P., Balu, N., and Lauby, M. (1994). *Power System Stability and Control*. EPRI power system engineering series. McGraw-Hill Education.

LeCun, Y., Bengio, Y., and Hinton, G. (2015). Deep learning. *nature*, 521(7553):436–444.

Lei, M., Shiyan, L., Chuanwen, J., Hongling, L., and Yan, Z. (2009). A review on the forecasting of wind speed and generated power. *Renewable and Sustainable Energy Reviews*, 13(4):915–920.

Lei, X., Yang, Z., Yu, J., Zhao, J., Gao, Q., and Yu, H. (2020). Data-driven optimal power flow: A physics-informed machine learning approach. *IEEE Transactions on Power Systems*, 36(1):346–354.

Lekshmi, M. and Nagaraj, M. S. (2018). Online static security assessment module using radial basis neural network trained with particle swarm optimization. *Lecture Notes in Electrical Engineering*, 446:215–224.

Li, W., Zhang, P., Su, S., Meng, X., Ding, C., and Wang, Y. (2018a). Comparison of decision tree attribute selection methods for static voltage stability margin assessment. *2018 IEEE 2nd International Electrical and Energy Conference*, pages 201–206.

Li, Y., Li, Y., and Sun, Y. (2018b). Online static security assessment of power systems based on Lasso algorithm. *Applied Sciences (Switzerland)*, 8(9).

Liang, G., Weller, S. R., Zhao, J., Luo, F., and Dong, Z. Y. (2016). The 2015 Ukraine blackout: Implications for false data injection attacks. *IEEE Transactions on Power Systems*, 32(4):3317–3318.

Louppe, G., Wehenkel, L., Sutera, A., and Geurts, P. (2013). Understanding variable importances in forests of randomized trees. *Advances in neural information processing systems 26*.

Low, S. H. (2014). Convex relaxation of optimal power flow?Part I: Formulations and equivalence. *IEEE Transactions on Control of Network Systems*, 1(1):15–27.

Luo, X., Zhang, S., and Litvinov, E. (2018). Practical design and implementation of cloud computing for power system planning studies. *IEEE Transactions on Smart Grid*, 10(2):2301–2311.

Mak, T. W., Fioretto, F., and VanHentenryck, P. (2021). Load embeddings for scalable ac-opf learning. *arXiv preprint arXiv:2101.03973*.

Marceau, R., Endrenyi, J., Allan, R., Alvarado, F., Bloemhof, G., Carlsen, T., Couto, G., Dialynas, E., Hatziargyriou, N., Holmberg, D., et al. (1997). Power system security assessment: A position paper. *Electra*, 175:49–77.

Marot, A., Tazi, S., Donnot, B., and Panciatici, P. (2018). Guided machine learning for power grid segmentation. In *2018 IEEE PES Innovative Smart Grid Technologies Conference Europe (ISGT-Europe)*, pages 1–6. IEEE.

MATLAB (n.d.). MATLAB and Statistics Toolbox Release 2015b.

Mezghani, I., Misra, S., and Deka, D. (2020). Stochastic AC optimal power flow: A data-driven approach. *Electric Power Systems Research*, 189:106567.

Misra, S., Roald, L., and Ng, Y. (2018). Learning for constrained optimization: Identifying optimal active constraint sets. *Arxiv*, pages 1–16.

Mitchell, T. M. (1997). *Machine learning.* McGraw Hill.

Mnih, V., Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., Wierstra, D., and Riedmiller, M. (2013). Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602.*

Molzahn, D. K. and Hiskens, I. A. (2019). A survey of relaxations and approximations of the power flow equations. *Electric Energy Systems*, 4(1-2):1–221.

Murphy, K. P. (2021). *Probabilistic Machine Learning: An introduction.* MIT Press.

Ng, Y., Misra, S., Roald, L. A., and Backhaus, S. (2018). Statistical learning for DC optimal power flow. *20th Power Systems Computation Conference, PSCC 2018*, pages 1–7.

Nguyen-Duc, H., Tran-Hoai, L., and Vo Ngoc, D. (2017). A novel approach to solve transient stability constrained optimal power flow problems. *Turkish Journal of Electrical Engineering and Computer Sciences*, 25(6):4696–4705.

Oates, C. J., Girolami, M., and Chopin, N. (2017). Control functionals for Monte Carlo integration. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 79(3):695–718.

Oliveira, W. D., Vieira, J. P., Bezerra, U. H., Martins, D. A., and Rodrigues, B. d. G. (2017). Power system security assessment for multiple contingencies using multiway decision tree. *Electric Power Systems Research*, 148:264–272.

Owerko, D., Gama, F., and Ribeiro, A. (2020). Optimal power flow using graph neural networks. In *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 5930–5934. IEEE.

Pan, X., Chen, M., Zhao, T., and Low, S. H. (2020a). DeepOPF: A feasibility-optimized deep neural network approach for AC optimal power flow problems. *arXiv preprint arXiv:2007.01002.*

Pan, X., Zhao, T., and Chen, M. (2019). DeepOPF: deep neural network for DC optimal power flow. *Arxiv.*

Pan, X., Zhao, T., Chen, M., and Zhang, S. (2020b). Deepopf: A deep neural network approach for security-constrained dc optimal power flow. *IEEE Transactions on Power Systems.*

Panciatici, P., Bareux, G., and Wehenkel, L. (2012). Operating in the fog: Security management under uncertainty. *IEEE Power and Energy Magazine*, 10(5):40–49.

Pandzic, H., Dvorkin, Y., Qiu, T., Wang, Y., and Kirschen, D. (n.d.). Unit Commitment under Uncertainty - GAMS Models. [Online]. Available at: http://www.ee.washington.edu/research/real/gams_code.html.

Panteli, M. and Mancarella, P. (2015). The grid: Stronger, bigger, smarter?: Presenting a conceptual framework of power system resilience. *IEEE Power and Energy Magazine*, 13(3):58–66.

Paramathma, M. K., Devaraj, D., and Reddy, B. S. (2016). Artificial neural network based static security assessment module using PMU measurements for smart grid application. *1st International Conference on Emerging Trends in Engineering, Technology and Science, ICETETS 2016 - Proceedings*, pages 1–5.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., and Duchesnay, E. (2011). Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830.

Pérez-Londoño, S. M., Olivar-tost, G., and Mora-florez, J. J. (2017). Online determination of voltage stability weak areas for situational awareness improvement. *Electric Power Systems Research*, 145:112–121.

Perkin, S., Svendsen, A. B., Tollefsen, T., Honve, I., Baldursdottir, I., Stefansson, H., Kristjansson, R., and Jensson, P. (2017). Modelling weather dependence in online reliability assessment of power systems. *Proceedings of the Institution of Mechanical Engineers, Part O: Journal of Risk and Reliability*, 231(4):364–372.

Pineda, S., Morales, J. M., and Jiménez-Cordero, A. (2020). Data-driven screening of network constraints for unit commitment. *IEEE Transactions on Power Systems*, 35(5):3695–3705.

Pozo, D. and Contreras, J. (2012). A chance-constrained unit commitment with an $n-k$ security criterion and significant wind generation. *IEEE Transactions on Power systems*, 28(3):2842–2851.

Prat, E. and Chatzivasileiadis, S. (2020). Learning active constraints to efficiently solve bilevel problems. *arXiv preprint arXiv:2010.06344*.

PyTorch developers (n.d.). Cross-entropy loss. [Online], Available at `https://pytorch.org/docs/stable/generated/torch.nn.CrossEntropyLoss.html`, accessed on 2021-02-16.

Rahman, J., Feng, C., and Zhang, J. (2021). A learning-augmented approach for AC optimal power flow. *International Journal of Electrical Power & Energy Systems*, 130:106908.

Roald, L. and Andersson, G. (2017). Chance-constrained AC optimal power flow: Reformulations and efficient algorithms. *IEEE Transactions on Power Systems*, 33(3):2906–2918.

Robson, A., Jamei, M., Ududec, C., and Mones, L. (2019). Learning an optimally reduced formulation of opf through meta-optimization. *arXiv preprint arXiv:1911.06784*.

Rosenblatt, F. (1958). The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological review*, 65(6):386.

Rubinstein, R. Y. and Kroese, D. P. (2016). *Simulation and the Monte Carlo method*, volume 10. John Wiley & Sons.

Saeh, I. S., Mustafa, M. W., Mohammed, Y. S., and Almaktar, M. (2016). Static security classification and evaluation classifier design in electric power grid with presence of PV power plants using C-4.5. *Renewable and Sustainable Energy Reviews*, 56:283–290.

Sankaranarayanan, P. and Rengaswamy, R. (2021). Cdinn-convex difference neural networks. *arXiv preprint arXiv:2103.17231*.

Schäfer, F., Menke, J.-H., and Braun, M. (2020). Evaluating machine learning models for the fast identification of contingency cases. *Applied AI Letters*.

Scikit-learn developers (n.d.). Model evaluation: quantifying the quality of predictions. [Online], Available at http://scikit-learn.org/stable/modules/model_evaluation.html, accessed on 2021-02-16.

Sekhar, P. and Mohanty, S. (2016). Classification and assessment of power system static security using decision tree and random forest classifiers. *International Journal of Numerical Modelling: Electronic Networks, Devices and Fields*, 29(3):465–474.

Shao, S., Pipattanasomporn, M., and Rahman, S. (2012). Grid integration of electric vehicles and demand response with customer choice. *IEEE transactions on smart grid*, 3(1):543–550.

Singh, P., Titare, L. S., Arya, L. D., and Choube, S. C. (2016). On-line probabilistic voltage security assessment using Radial Basis Function neural network. *Proceedings of 2016 International Conference on ICT in Business, Industry, and Government, ICTBIG 2016*.

Singhal, N. G., Li, N., and Hedman, K. W. (2018). A data-driven reserve response set policy for power systems with stochastic resources. *IEEE Transactions on Sustainable Energy*, 10(2):693–705.

Smola, A. J. and Schölkopf, B. (2004). A tutorial on support vector regression. *Statistics and computing*, 14(3):199–222.

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. (2014). Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, 15(1):1929–1958.

Strbac, G., Kirschen, D., and Moreno, R. (2016). Reliability standards for the operation and planning of future electricity networks.

Su, H. Y. and Liu, T. Y. (2018). Enhanced-online-random-forest model for static voltage stability assessment using wide area measurements. *IEEE Transactions on Power Systems*, 33(6):6696–6704.

Sun, M., Konstantelos, I., and Strbac, G. (2018). A deep learning-based feature extraction framework for system security assessment. *IEEE Transactions on Smart Grid*, PP(c):1.

Sutera, A. (2019). *Importance measures derived from random forests: characterisation and extension.* PhD thesis, Université de Liège, Liège, Belgique.

Sutton, R. S. and Barto, A. G. (2018). *Reinforcement learning: An introduction.* MIT press.

Szegedy, C., Zaremba, W., Sutskever, I., Bruna, J., Erhan, D., Goodfellow, I., and Fergus, R. (2013). Intriguing properties of neural networks. *arXiv preprint arXiv:1312.6199.*

Thams, F., Halilbasic, L., Pinson, P., Chatzivasileiadis, S., and Eriksson, R. (2017). Data-driven security-constrained OPF. *Proceedings of the 10th Bulk Power Systems Dynamics and Control Symposium.*

Thirugnanasambandam, V. and Jain, T. (2018). AdaBoost classifiers for phasor measurements-based security assessment of power systems. *IET Generation, Transmission and Distribution*, 12(8):1747–1755.

Tracey, B. D. and Wolpert, D. H. (2016). Reducing the error of Monte Carlo algorithms by learning control variates. *arXiv preprint arXiv:1606.02261.*

Urgun, D. and Singh, C. (2018). Multi label RBF classification method for composite system reliability evaluation. *2018 International Conference on Probabilistic Methods Applied to Power Systems, PMAPS 2018 - Proceedings.*

Urgun, D. and Singh, C. (2019). A hybrid monte carlo simulation and multi label classification method for composite system reliability evaluation. *IEEE Transactions on Power Systems*, 34(2):908–917.

U.S.-Canada Power System Outage Task Force (2004). Final report on the august 14, 2003 blackout in the united states and canada: Causes and recommendations. Technical report.

van Ackooij, W., Lopez, I. D., Frangioni, A., Lacalandra, F., and Tahanan, M. (2018). Large-scale unit commitment under uncertainty: an updated literature survey. *Annals of Operations Research*, 271(1):11–85.

Vasconcelos, M., Carvalho, L. M., Meirinhos, J., Omont, N., Gambier-Morel, P., Jamgotchian, G., Cirio, D., Ciapessoni, E., Pitto, A., Konstantelos, I., et al. (2016). Online security assessment with load and renewable generation uncertainty: The itesla project approach. In *2016 International Conference on Probabilistic Methods Applied to Power Systems (PMAPS)*, pages 1–8. IEEE.

Velloso, A. and Van Hentenryck, P. (2020). Combining deep learning and optimization for security-constrained optimal power flow. *arXiv preprint arXiv:2007.07002*.

Venzke, A., Molzahn, D. K., and Chatzivasileiadis, S. (2019). Efficient creation of datasets for data-driven power system applications. *Arxiv*, pages 1–8.

Venzke, A., Qu, G., Low, S., and Chatzivasileiadis, S. (2020a). Learning optimal power flow: Worst-case guarantees for neural networks. In *2020 IEEE International Conference on Communications, Control, and Computing Technologies for Smart Grids (SmartGridComm)*, pages 1–7. IEEE.

Venzke, A., Viola, D. T., Mermet-Guyennet, J., Misyris, G. S., and Chatzivasileiadis, S. (2020b). Neural networks for encoding dynamic security-constrained optimal power flow to mixed-integer linear programs. *arXiv preprint arXiv:2003.07939*.

Wang, B., Zhou, M., Xin, B., Zhao, X., and Watada, J. (2019). Analysis of operation cost and wind curtailment using multi-objective unit commitment with battery energy storage. *Energy*, 178:101–114.

Weedy, B. M., Cory, B. J., Jenkins, N., Ekanayake, J. B., and Strbac, G. (2012). *Electric power systems*. John Wiley & Sons.

Wehenkel, L. (1997). Machine learning approaches to power-system security assessment. *IEEE Expert*, 12(5):60–72.

Wehenkel, L., Van Cutsem, T., and Ribbens-Pavella, M. (1989). An artificial intelligence framework for online transient stability assessment of power systems. *IEEE Transactions on Power Systems*, 4(2):789–800.

Wehenkel, L. A. (2012). *Automatic learning techniques in power systems*. Springer Science & Business Media.

Weiss, K., Khoshgoftaar, T. M., and Wang, D. (2016). A survey of transfer learning. *Journal of Big data*, 3(1):1–40.

Wikipedia (2021). 2021 texas power crisis. Available at https://en.wikipedia.org/wiki/2021_Texas_power_crisis, accessed on 2021-05-12.

Wikipedia (n.d.). Northeast blackout of 2003. Available at https://en.wikipedia.org/wiki/Northeast_blackout_of_2003, accessed on 2021-08-17.

Wood, A. J. and Wollenberg, B. F. (2012). *Power generation, operation, and control*. John Wiley & Sons.

Wu, L., Shahidehpour, M., and Li, T. (2007). Stochastic security-constrained unit commitment. *IEEE Transactions on power systems*, 22(2):800–811.

Wu, Z., Pan, S., Chen, F., Long, G., Zhang, C., and Philip, S. Y. (2020). A comprehensive survey on graph neural networks. *IEEE transactions on neural networks and learning systems*.

Xavier, Á. S., Qiu, F., and Ahmed, S. (2020). Learning to solve large-scale security-constrained unit commitment problems. *INFORMS Journal on Computing*.

Yan, J., Liu, Y., Han, S., Wang, Y., and Feng, S. (2015a). Reviews on uncertainty analysis of wind power forecasting. *Renewable and Sustainable Energy Reviews*, 52:1322–1330.

Yan, J., Tang, Y., He, H., and Sun, Y. (2015b). Cascading failure analysis with dc power flow model and transient stability analysis. *IEEE Transactions on Power Systems*, 30(1):285–297.

Yan, Z. and Xu, Y. (2020). Real-time optimal power flow: A lagrangian based deep reinforcement learning approach. *IEEE Transactions on Power Systems*, 35(4):3270–3273.

Yang, S. and Bequette, B. W. (2021). Optimization-based control using input convex neural networks. *Computers & Chemical Engineering*, 144:107143.

Yang, Y., Yang, Z., Yu, J., Zhang, B., Zhang, Y., and Yu, H. (2019). Fast calculation of probabilistic power flow: A model-based deep learning approach. *IEEE Transactions on Smart Grid*, 11(3):2235–2244.

Yun, Z., Zhou, Q., Feng, Y., Sun, D., Sun, J., and Yang, D. (2017). On-line static voltage security risk assessment based on Markov chain model and SVM for wind integrated power system. In *2017 13th International Conference on Natural Computation, Fuzzy Systems and Knowledge Discovery (ICNC-FSKD)*, pages 2469–2473. IEEE.

Zamzam, A. and Baker, K. (2019). Learning optimal solutions for extremely fast AC optimal power flow. *Arxiv*, pages 1–7.

Zhang, L., Chen, Y., and Zhang, B. (2020). A convex neural network solver for dcopf with generalization guarantees. *arXiv preprint arXiv:2009.09109*.

Zhang, Y., Cui, H., Liu, J., Qiu, F., Hong, T., Yao, R., and Li, F. (2021). Encoding frequency constraints in preventive unit commitment using deep learning with region-of-interest active sampling. *arXiv preprint arXiv:2102.09583*.

Zhao, T., Pan, X., Chen, M., Venzke, A., and Low, S. H. (2020). Deepopf+: A deep neural network approach for dc optimal power flow for ensuring feasibility. In *2020 IEEE International Conference on Communications, Control, and Computing Technologies for Smart Grids (SmartGridComm)*, pages 1–6. IEEE.

Zhou, M., Wang, B., Li, T., and Watada, J. (2018a). A data-driven approach for multi-objective unit commitment under hybrid uncertainties. *Energy*, 164:722–733.

Zhou, S. and Brown, M. A. (2017). Smart meter deployment in Europe: A comparative case study on the impacts of national policy schemes. *Journal of Cleaner Production*, 144:22–32.

Zhou, Y., Wu, J., Ji, L., Yu, Z., Lin, K., and Hao, L. (2018b). Transient stability preventive control of power systems using chaotic particle swarm optimization combined with two-stage support vector machine. *Electric Power Systems Research*, 155:111–120.

Zhou, Y., Zhang, B., Xu, C., Lan, T., Diao, R., Shi, D., Wang, Z., and Lee, W.-J. (2020a). A data-driven method for fast ac optimal power flow solutions via deep reinforcement learning. *Journal of Modern Power Systems and Clean Energy*, 8(6):1128–1139.

Zhou, Y., Zhang, B., Xu, C., Lan, T., Diao, R., Shi, D., Wang, Z., and Lee, W.-J. (2020b). Deriving fast ac opf solutions via proximal policy optimization for secure and economic grid operation. *arXiv preprint arXiv:2003.12584*.

Zhukov, A., Tomin, N., Kurbatsky, V., Sidorov, D., Panasetsky, D., and Foley, A. (2017). Ensemble methods of classification for power systems security assessment. *Applied Computing and Informatics*, 15(1):45–53.

Part V

Appendices

A

Mathematical models of day-ahead and real-time reliability management

Overview

This appendix details the implementation of the day-ahead and real-time operation simulators used during the thesis. It begins with a description of the notations used in the mathematical models, then it presents the day-ahead decision-making program and finally it describes the real-time SCOPF.

References: This appendix is an adapted version of the supplementary material of the following publication:

Duchesne, L., Karangelos, E., and Wehenkel, L. (2018). Using machine learning to enable probabilistic reliability assessment in operation planning. In *2018 Power Systems Computation Conference (PSCC)*, pages 1–8. IEEE.

A.1 Notations

Indices

c	Index of contingencies
d	Index of demands
g	Index of generating units
k	Index of piece-wise linear dispatchable generation cost curve segments
l	Index of transmission elements (lines, cables and transformers)
n	Index of nodes
t	Index of hours in a day
w	Index of wind power generators

Sets

$\mathcal{C}$	Set of contingencies
$\mathcal{D}$	Set of demands
$\mathcal{D}_n$	Subset of demands connected at node n
$\mathcal{G}$	Set of dispatchable units
$\mathcal{K}$	Set of piece-wise linear dispatchable generation cost curve segments
$\mathcal{L}$	Set of transmission elements (lines, cables and transformers)

$\mathcal{N}$	Set of nodes
$\mathcal{W}$	Set of wind power generators
$\mathcal{W}_n$	Subset of wind power generators connected at node n

Parameters

$P_{d,t}^{forecast}$	Forecast of load active power of demand d at time t
$P_{w,t}^{forecast}$	Forecast of generation of wind power generator w at time t
$P_{d,t}^{RT}$	Realisation of load active power of demand d at time t
$P_{w,t}^{RT}$	Realisation of generation of wind power generator w at time t
on_g^{init}	Initial status of generating unit g at the beginning of the day-ahead decision-making (1 if started up, 0 otherwise)
$t_g^{up,init}$	Minimum number of time periods generating unit g must stay up at the beginning of the considered day
$t_g^{dn,init}$	Minimum number of time periods generating unit g must stay down at the beginning of the considered day
$t_g^{up,min}$	Minimum number of time periods generating unit g must stay up once started up
$t_g^{dn,min}$	Minimum number of time periods generating unit g must stay down once shut down
c_g	Redispatch marginal cost of generating unit g
c_g^0	Start-up cost of generating unit g
$c_{g,k}^{inc}$	Marginal running cost of generating unit g at the segment k of its piece-wise linear curve
P_g^{max}	Capacity of generating unit g
P_g^{min}	Minimum stable output of generating unit g
ΔP_g^-	Ramp-down limit of generating unit g (for 60min)
ΔP_g^+	Ramp-up limit of generating unit g (for 60min)
$\Delta P_g^{-,c}$	Ramp-down limit of generating unit g in case of corrective actions (for 20min)
$\Delta P_g^{+,c}$	Ramp-up limit of generating unit g in case of corrective actions (for 20min)
$P_{g,k}^{inc,max}$	Maximum power output of generating unit g at the segment k of its piece-wise linear curve
v_d	Voll of demand d in €/MWh
p_w	Wind penalty for curtailment of wind power generator w in €/MWh
R^+	Minimum up spinning reserve required per hour for one area
R^-	Minimum down spinning reserve required per hour for one area
f_l^{max}	Long-term thermal rating of transmission element l
r_l	Ratio of the short-term thermal rating to the long-term thermal rating of transmission element l ($r_l \geq 1$)
X_l	Reactance of transmission element l

$\beta_{n,l}$	Element of the flow incidence matrix, taking a value of one if node n is the sending node of element l, a value of minus one if node n is the receiving node of element l, and a zero value otherwise.
$a_{l,c}$	Binary parameter taking a zero value if element l is unavailable under contingency c.

Variables

$P_{g,t}^{DA}$	Dispatch of generating unit g at time t as per the day-ahead decision-making
$on_{g,t}^{DA}$	Binary variable representing the status of generating unit g as per the day-ahead decision-making (1 if started up, 0 otherwise)
$st_{g,t}^{up}$	Binary variable indicating when generating unit g is started-up (value 1 when started up, 0 otherwise)
$st_{g,t}^{dn}$	Binary variable indicating when generating unit g is shut down (value 1 when shut down, 0 otherwise)
$WC_{w,t}^{DA}$	Provisional curtailment of wind power generator w at hour t in day-ahead
$R_{g,t}^{+}$	Upward redispatch flexibility provided by generating unit g at time t in day-ahead
$R_{g,t}^{-}$	Downward redispatch flexibility provided by generating unit g at time t in day-ahead
$f_{l,t}^{DA}$	Power flowing through transmission element l at time t under the pre-contingency state in day-ahead
$f_{l,t,c}^{DAST}$	Power flowing through transmission element l at time t following contingency c in day-ahead
$\theta_{n,t}^{DA}$	Voltage angle at node n under the pre-contingency state in day-ahead
$\theta_{l,t,c}^{DAST}$	Voltage angle at node n following contingency c in day-ahead.
${}^{+}P_{g,t}^{RTp}$	Preventive ramp-up of generator g in real-time at hour t
${}^{-}P_{g,t}^{RTp}$	Preventive ramp-down of generator g in real-time at hour t
$LS_{d,t}^{RTp}$	Preventive load shedding of demand d in real-time at hour t
$WC_{w,t}^{RT^{p}}$	Preventive wind curtailment of wind power generator w in real-time at hour t
${}^{+}P_{g,t,c}^{RTc}$	Corrective ramp-up of generator g in real-time at hour t following contingency c
${}^{-}P_{g,t,c}^{RTc}$	Corrective ramp-down of generator g in real-time at hour t following contingency c
$LS_{d,t,c}^{RTc}$	Corrective load shedding of demand d in real-time at hour t following contingency c
$WC_{w,t,c}^{RTc}$	Corrective wind curtailment of wind power generator w in real-time at hour t following contingency c
$f_{l,t}^{p}$	Power flowing through transmission element l under the pre-contingency state

$f^{ST}_{l,t,c}$	Power flowing through transmission element l following contingency c and prior to the application of corrective control.
$f^{c}_{l,t,c}$	Power flowing through transmission element l following contingency c and the successful application of corrective control.
$\theta^{p}_{n,t}$	Voltage angle at node n under the pre-contingency state
$\theta^{ST}_{n,t,c}$	Voltage angle at node n following contingency c and prior to the application of corrective control.
$\theta^{c}_{n,t,c}$	Voltage angle at node n following contingency c and the successful application of corrective control.

All the variables are continuous, except for $on_{g,t}$, $st^{dn}_{g,t}$ and $st^{up}_{g,t}$ which are binary variables. Powers flowing through transmission elements and voltage angles are continuous in $\mathbb{R}$ and the remaining variables are positive.

A.2 Day-ahead decision-making

We simulate day-ahead decision-making with a multi-period security-constrained UC in order to commit and dispatch the generating units of the system and also determine the provisional wind curtailment. We use the DC approximation [Wood and Wollenberg, 2012] and consider as reliability criterion the N-1 criterion for transmission elements only. This formulation does not intend to fairly reflect the actual process of day-ahead decision-making but is an approximation that remains reasonable without leading to a too complex implementation.

The objective function minimizes generation cost as well as provisional wind curtailment:

$$\text{minimize} \quad \sum_{t=1}^{24}\left(\sum_{g\in\mathcal{G}}\left(c^0_g * st^{up}_{g,t} + \sum_{k\in\mathcal{K}} c^{inc}_{g,k} * P^{inc}_{g,k,t}\right) + \sum_{w\in\mathcal{W}} p_w * WC^{DA}_{w,t}\right) \tag{A.1}$$

The first set of constraints (A.2-A.9) of the day-ahead program concerns the minimum time a generating unit must stay up or down, either at the beginning of the day or during the day.

For $t = 1$, $\forall g \in \mathcal{G}$:

$$st^{up}_{g,t} - st^{dn}_{g,t} = on^{DA}_{g,t} - on^{init}_{g} \tag{A.2}$$

$$st^{up}_{g,t} + st^{dn}_{g,t} \leq 1 \tag{A.3}$$

$\forall t = 2, ..., 24$, $\forall g \in \mathcal{G}$:

$$st^{up}_{g,t} - st^{dn}_{g,t} = on^{DA}_{g,t} - on^{DA}_{g,t-1} \tag{A.4}$$

$$st^{up}_{g,t} + st^{dn}_{g,t} \leq 1 \tag{A.5}$$

$\forall g \in \mathcal{G}$:

$$\sum_{t'=1}^{t^{up,init}_g} \left(1 - on^{DA}_{g,t'}\right) = 0 \tag{A.6}$$

$$\sum_{t'=1}^{t_g^{dn,init}} on_{g,t'}^{DA} = 0 \tag{A.7}$$

$\forall g \in \mathcal{G}$, $\forall t = 1, ..., \left(24 - t_g^{up,min}\right)$:

$$\sum_{t'=t}^{t+t_g^{up,min}} on_{g,t'}^{DA} \geq st_{g,t}^{up} \cdot t_g^{up,min} \tag{A.8}$$

$\forall g \in \mathcal{G}$, $\forall t = 1, ..., \left(24 - t_g^{dn,min}\right)$:

$$\sum_{t'=t}^{t+t_g^{dn,min}} \left(1 - on_{g,t'}^{DA}\right) \geq st_{g,t}^{dn} \cdot t_g^{dn,min} \tag{A.9}$$

The following set of constraints limits the power output of each generating unit between its minimum stable output and its maximum capacity, computes the upward and downward redispatch flexibility of each generator and also imposes ramping constraints to go from one committed dispatch to the one of the next period in one hour:

$\forall g \in \mathcal{G}$, $\forall t = 1, ..., 24$:

$$-P_{g,t}^{DA} + R_{g,t}^{-} \leq -P_g^{min} \cdot on_{g,t}^{DA} \tag{A.10}$$

$$P_{g,t}^{DA} + R_{g,t}^{+} \leq P_g^{max} \cdot on_{g,t}^{DA} \tag{A.11}$$

$$P_{g,t+1}^{DA} - P_{g,t}^{DA} \leq \Delta P_g^{+} \cdot on_{g,t}^{DA} + P_g^{max}(1 - on_{g,t}^{DA}) \tag{A.12}$$

$$-\left(P_{g,t+1}^{DA} - P_{g,t}^{DA}\right) \leq \Delta P_g^{-} \cdot on_{g,t}^{DA} + P_g^{max}(1 - on_{g,t+1}^{DA}) \tag{A.13}$$

We assume a piece-wise linear cost function of $|\mathcal{K}|$ segments for the marginal running cost of a generating unit g, which gives eq. (A.14) and (A.15).

$\forall g \in \mathcal{G}$, $\forall k \in \mathcal{K}$, $\forall t = 1, ..., 24$:

$$P_{g,k,t}^{inc} \leq on_{g,t}^{DA} \cdot \; P_{g,k}^{inc,max} \tag{A.14}$$

$\forall g \in \mathcal{G}$, $\forall t = 1, ..., 24$:

$$P_{g,t}^{DA} = \sum_{k=1}^{K} P_{g,k,t}^{inc} \tag{A.15}$$

Equation (A.16) represents the balancing of the system and equations (A.18)-(A.20) the transmission constraints in case of the DC approximation.

$\forall t = 1, ..., 24$, $\forall n \in \mathcal{N}$:

$$\sum_{w \in \mathcal{W}_n} (P_{w,t}^{forecast} - WC_{w,t}^{DA}) + \sum_{g \in \mathcal{G}_n} P_{g,t}^{DA} - \sum_{l \in \mathcal{L}} \beta_{n,l} \cdot f_{l,t}^{DA} = \sum_{d \in \mathcal{D}_n} P_{d,t}^{forecast} \tag{A.16}$$

$\forall t = 1, ..., 24$, $\forall w \in \mathcal{W}$:

$$0 \leq WC_{w,t}^{DA} \leq P_{w,t}^{forecast} \tag{A.17}$$

$\forall t = 1, ..., 24$, $\forall l \in \mathcal{L}$:

$$f_{l,t}^{DA} = \frac{1}{X_l} \sum_{n \in \mathcal{N}} \beta_{n,l} \cdot \theta_{l,t}^{DA} \tag{A.18}$$

$$f_{l,t}^{DA} \leq f_l^{max} \tag{A.19}$$

$$-f_{l,t}^{DA} \leq f_l^{max} \tag{A.20}$$

Equations (A.21)-(A.24) force the system to still be secure in the case of the loss of one transmission element.

$\forall t = 1, ..., 24$, $\forall c \in \mathcal{C}, \forall n \in \mathcal{N}$:

$$\sum_{w \in \mathcal{W}_n} (P_{w,t}^{forecast} - WC_{w,t}^{DA}) + \sum_{g \in \mathcal{G}_n} P_{g,t}^{DA} - \sum_{l \in \mathcal{L}} \beta_{n,l} \cdot f_{l,t,c}^{DAST} = \sum_{d \in \mathcal{D}_n} P_{d,t}^{forecast} \tag{A.21}$$

$\forall t = 1, ..., 24$, $\forall c \in \mathcal{C}, \forall l \in \mathcal{L}$:

$$f_{l,t,c}^{DAST} = a_{l,c} \cdot \frac{1}{X_l} \sum_{n \in \mathcal{N}} \beta_{n,l} \cdot \theta_{l,t,c}^{DAST} \tag{A.22}$$

$$f_{l,t,c}^{DAST} \leq a_{l,c} \cdot f_l^{max} \tag{A.23}$$

$$-f_{l,t,c}^{DAST} \leq a_{l,c} \cdot f_l^{max} \tag{A.24}$$

Equations (A.25)-(A.30) determine the minimum size of the up and down spinning reserves per area in the system (in this work, we have three areas and the same spinning reserve requirements per area).

$\forall t = 1, ..., 24$, $\forall g = \in \mathcal{G}$

$$\sum_{g \in \mathcal{G}_{area1}} R_{g,t}^{+} \geq R^{+} \tag{A.25}$$

$$\sum_{g \in \mathcal{G}_{area1}} R_{g,t}^{-} \geq R^{-} \tag{A.26}$$

$$\sum_{g \in \mathcal{G}_{area2}} R_{g,t}^{+} \geq R^{+} \tag{A.27}$$

$$\sum_{g \in \mathcal{G}_{area2}} R_{g,t}^{-} \geq R^{-} \tag{A.28}$$

$$\sum_{g \in \mathcal{G}_{area3}} R_{g,t}^{+} \geq R^{+} \tag{A.29}$$

$$\sum_{g \in \mathcal{G}_{area3}} R_{g,t}^{-} \geq R^{-} \tag{A.30}$$

A.3 Real-time operation

In order to simulate real-time operation along a system trajectory, we solve sequentially the 24 hourly steps of the trajectory. That is we solve 24 single period problems corresponding to the 24 hours of one day.

We model real-time operation with a SCOPF problem with the N-1 reliability criterion, again considering only transmission elements. We consider preventive (pre-contingency) as well as corrective (post-contingency) actions and we do not forget the intermediate state after the occurrence of a contingency but before any corrective action can be applied, that we call short-term post-contingency state.

Note that continuous variables from the day-ahead decision-making program are parameters for this problem.

A.3.1 Objective function

The objective function (A.31) minimizes the redispatch cost (upward and downward) as well as load shedding and wind curtailment, both in preventive and corrective modes. The value of lost load and wind penalty should be such that load shedding and wind curtailment are used only where no other solution exists. In order to favour corrective actions over preventive ones, we multiply the total preventive cost by a large factor M.

minimize

$$M * \left(\sum_{g\in\mathcal{G}} c_g \left({}^{+}P^{RTp}_{g,t} + {}^{-}P^{RTp}_{g,t} \right) + \sum_{d\in\mathcal{D}} v_d * LS^{RTp}_{d,t} + \sum_{w\in\mathcal{W}} p_w * WC^{RTp}_{w,t} \right) + \sum_{c\in\mathcal{C}} \left(\sum_{d\in\mathcal{D}} v_d * LS^{RTc}_{d,t,c} + \sum_{w\in\mathcal{W}} p_w * WC^{RTc}_{w,t,c} + \sum_{g\in\mathcal{G}} c_g \left({}^{+}P^{RTc}_{g,t,c} + {}^{-}P^{RTc}_{g,t,c} \right) \right) \tag{A.31}$$

A.3.2 Pre-contingency state

The following equations determine the preventive actions.The possible redispatch of generating units is limited by maximum and minimum output power of generating units as well as by ramping constraints of one hour. Equations (A.36) and (A.37) also impose that with the re-dispatch of a unit g, it is still possible to go in one hour to the dispatch of the generating unit g at time $t+1$ as per the day-ahead decision-making.

$\forall g \in \mathcal{G}$:

$$P^{DA}_{g,t} + ({}^{+}P^{RTp}_{g,t} - {}^{-}P^{RTp}_{g,t}) \geq P^{min}_g \cdot on^{DA}_{g,t} \tag{A.32}$$

$$P^{DA}_{g,t} + ({}^{+}P^{RTp}_{g,t} - {}^{-}P^{RTp}_{g,t}) \leq P^{max}_g \cdot on^{DA}_{g,t} \tag{A.33}$$

$${}^{+}P^{RTp}_{g,t} - {}^{-}P^{RTp}_{g,t} \leq \Delta P^{+}_g \tag{A.34}$$

$$-({}^{+}P^{RTp}_{g,t} - {}^{-}P^{RTp}_{g,t}) \leq \Delta P^{-}_g \tag{A.35}$$

$$P^{DA}_{g,t+1} - \left(P^{DA}_{g,t} + ({}^{+}P^{RTp}_{g,t} - {}^{-}P^{RTp}_{g,t}) \right) \leq \Delta P^{+}_g \tag{A.36}$$

$$-\left(P^{DA}_{g,t+1} - \left(P^{DA}_{g,t} + {}^{+}P^{RTp}_{g,t} - {}^{-}P^{RTp}_{g,t} \right) \right) \leq \Delta P^{-}_g \tag{A.37}$$

The next constraints correspond to the classical DC approximation.

$\forall n \in \mathcal{N}$:

$$\begin{aligned}&\sum_{w\in\mathcal{W}_n}\left(P^{RT}_{w,t} - WC^{DA}_{w,t} - WC^{RT^p}_{w,t}\right)\\&+\sum_{g\in\mathcal{G}_n}\left(P^{DA}_{g,t} + ({}^+P^{RTp}_{g,t} - {}^-P^{RTp}_{g,t})\right)\\&-\sum_{l\in\mathcal{L}}\beta_{n,l}\cdot f^p_{l,t}\\&=\sum_{d\in\mathcal{D}_n}\left(P^{RT}_{d,t} - LS^{RT^p}_{d,t}\right)\end{aligned} \tag{A.38}$$

$\forall l \in \mathcal{L}$:

$$f^p_{l,t} = \frac{1}{X_l}\sum_{n\in\mathcal{N}_n}\beta_{n,l}\cdot\theta^p_{l,t} \tag{A.39}$$

$$f^p_{l,t} \leq f^{max}_l \tag{A.40}$$

$$-f^p_{l,t} \leq f^{max}_l \tag{A.41}$$

Finally, we ensure that we do not shed more load and wind generation than what is possible:
$\forall d \in \mathcal{D}$:

$$0 \leq LS^{RT^p}_{d,t} \leq P^{RT}_{d,t} \tag{A.42}$$

$\forall w \in \mathcal{W}$:

$$0 \leq WC^{DA}_{w,t} + WC^{RT^p}_{w,t} \leq P^{RT}_{w,t} \tag{A.43}$$

A.3.3 Short-term post-contingency state

In this stage, a contingency occurred but the operator has not reacted yet. Since we are in emergency state, the line thermal ratings correspond to the short-term ones.
$\forall c \in \mathcal{C}, \forall n \in \mathcal{N}$:

$$\begin{aligned}&\sum_{w\in\mathcal{W}_n}\left(P^{RT}_{w,t} - WC^{DA}_{w,t} - WC^{RT^p}_{w,t}\right)\\&+\sum_{g\in\mathcal{G}_n}\left(P^{DA}_{g,t} + ({}^+P^{RTp}_{g,t} - {}^-P^{RTp}_{g,t})\right)\\&-\sum_{l\in\mathcal{L}}\beta_{n,l}\cdot f^{ST}_{l,t,c}\\&=\sum_{d\in\mathcal{D}_n}\left(P^{RT}_{d,t} - LS^{RT^p}_{d,t}\right)\end{aligned} \tag{A.44}$$

$\forall c \in \mathcal{C}, \forall l \in \mathcal{L}$:

$$f^{ST}_{l,t,c} = a_{l,c}\cdot\frac{1}{X_l}\sum_{n\in\mathcal{N}_n}\beta_{n,l}\cdot\theta^{ST}_{l,t,c} \tag{A.45}$$

$$f^{ST}_{l,t,c} \leq a_{l,c}\cdot r_l\cdot f^{max}_l \tag{A.46}$$

$$-f^{ST}_{l,t,c} \leq a_{l,c}\cdot r_l\cdot f^{max}_l \tag{A.47}$$

A.3.4 Corrective control

Finally, corrective actions can be applied to keep the system secure.
$\forall c \in \mathcal{C}, \forall g \in \mathcal{G}$:

$$P_{g,t}^{DA} + ({}^{+}P_{g,t}^{RTp} -{}^{-} P_{g,t}^{RTp}) + ({}^{+}P_{g,t,c}^{RTc} -{}^{-} P_{g,t,c}^{RTc}) \geq P_{g}^{min} \cdot on_{g,t}^{DA} \tag{A.48}$$

$$P_{g,t}^{DA} + ({}^{+}P_{g,t}^{RTp} -{}^{-} P_{g,t}^{RTp}) + ({}^{+}P_{g,t,c}^{RTc} -{}^{-} P_{g,t,c}^{RTc}) \leq P_{g}^{max} \cdot on_{g,t}^{DA} \tag{A.49}$$

$${}^{+}P_{g,t,c}^{RTc} -{}^{-} P_{g,t,c}^{RTc} \leq \Delta P_{g}^{+,c} \tag{A.50}$$

$$-({}^{+}P_{g,t,c}^{RTc} -{}^{-} P_{g,t,c}^{RTc}) \leq \Delta P_{g}^{-,c} \tag{A.51}$$

$\forall c \in \mathcal{C}, \forall n \in \mathcal{N}$:

$$\begin{aligned} &\sum_{w \in \mathcal{W}_n} (P_{w,t}^{RT} - WC_{w,t}^{DA} - WC_{w,t}^{RT^p} - WC_{w,t,c}^{RT^c}) \\ &+ \sum_{g \in \mathcal{G}_n} \Big(P_{g,t}^{DA} + ({}^{+}P_{g,t}^{RTp} -{}^{-} P_{g,t}^{RTp}) + ({}^{+}P_{g,t,c}^{RTc} -{}^{-} P_{g,t,c}^{RTc}) \Big) \\ &- \sum_{l \in \mathcal{L}} \beta_{n,l} \cdot f_{l,t,c}^{c} \\ &= \sum_{d \in \mathcal{D}_n} (P_{d,t}^{RT} - LS_{d,t}^{RT^p} - LS_{d,t,c}^{RT^c}) \end{aligned} \tag{A.52}$$

$\forall c \in \mathcal{C}, \forall d \in \mathcal{D}$:

$$0 \leq LS_{d,t}^{RT^p} + LS_{d,t,c}^{RT^c} \leq P_{d,t}^{RT} \tag{A.53}$$

$\forall c \in \mathcal{C}, \forall w \in \mathcal{W}$:

$$0 \leq WC_{w,t}^{DA} + WC_{w,t}^{RT^p} + WC_{w,t,c}^{RT^c} \leq P_{w,t}^{RT} \tag{A.54}$$

$\forall c \in \mathcal{C}, \forall l \in \mathcal{L}$:

$$f_{l,t,c}^{c} = a_{l,c} \cdot \frac{1}{X_l} \sum_{n \in \mathcal{N}_n} \beta_{n,l} \cdot \theta_{l,t}^{ST} \tag{A.55}$$

$$f_{l,t,c}^{c} \leq a_{l,c} \cdot f_{l}^{max} \tag{A.56}$$

$$-f_{l,t,c}^{c} \leq a_{l,c} \cdot f_{l}^{max} \tag{A.57}$$

B

Day-ahead uncertainty models

Overview

This appendix describes the uncertainty models used in chapter 5 in a day-ahead context (more specifically, at noon the day before) to generate the real-time scenarios for Monte-Carlo simulations. The considered day-ahead uncertainties are the weather, the components unplanned outages, the probabilities of contingencies of lines and transformers and the load and wind generation forecast errors. Note that the load and wind generation forecast error models used in chapters 6 and 7 to generate real-time scenarios are based on the models presented in this appendix.
Unless otherwise specified, all the data come from [Grigg et al., 1999].
References: This appendix is an adapted version of chapter 4 of the following work:

Duchesne, L. (2016). Machine learning of proxies for power systems reliability management. Master's thesis, Université de Liège, Liège, Belgique.

Compared to this document, the text has been processed for coherence with the rest of the manuscript.

B.1 Weather

The weather has an influence on the real-time reliability management of the system. For the sake of simplicity, only two states are considered: normal and adverse. The weather is adverse when there is lightning, thunderstorm, ice or snow [Billinton and Allan, 1996]. It is assumed that the whole system has the same weather.

In chapter 5, the weather has an impact on the transmission lines. Indeed, when the weather is adverse, their probability of outage increases. This affects the availability of transmission lines as well as the probabilities of contingencies.

In order to be able to analyze the weather influence, each state is generated twice: once with a normal weather and the other with an adverse weather.

B.2 Generating units

To assess which generators will be available the next day, a Poisson distribution is used to determine the probability of unavailability [Billinton, 1970]. The failure rate per hour (λ) is the inverse of the Mean Time To Failure (MTTF), given in [Grigg et al., 1999] for each type of generating unit. The probability of having x occurrences of failure in t hours is:

$$P_t(x) = \frac{(\lambda t)^x e^{-\lambda t}}{x!}. \tag{B.1}$$

We consider that the generating units will not be repaired in one day and thus the probability of unavailability of a generator is given by 1 minus the probability that there is no failure ($x = 0$) in t hours:

$$P_{outage}(t) = 1 - P(x = 0) = 1 - e^{-\lambda t}. \tag{B.2}$$

The probabilities of outage for each unit group can be found in Table B.1. For simplicity, the generating units are assumed to be in only two possible states, up or down.

In our study, the parameter t ranges from 12 to 35h.

Table B.1: Number of outages per hour λ and probability of unavailability for $t = 12$, 24 and 36h for each type of generating unit.

Unit group	λ [outage/h]	P_{outage} $(t = 12h)$	P_{outage} $(t = 24h)$	P_{outage} $(t = 36h)$
U12	$3.4\ 10^{-4}$	0.0041	0.0081	0.012
U20	$2.2\ 10^{-3}$	0.026	0.052	0.077
U50	$5.1\ 10^{-4}$	0.006	0.012	0.018
U76	$5.1\ 10^{-4}$	0.0061	0.012	0.018
U100	$8.3\ 10^{-4}$	0.01	0.0198	0.0296
U155	$1\ \ 10^{-3}$	0.012	0.025	0.037
U197	$1.1\ \ 10^{-3}$	0.013	0.025	0.037
U350	$8.7\ 10^{-4}$	0.010	0.021	0.031
U400	$9.1\ 10^{-4}$	0.011	0.027	0.032

B.3 Transmission lines and transformers

The same reasoning as for the generating units is used to determine the probability that there is a forced outage on a line or a transformer. However, the weather has a strong impact on the probability of failure of transmission lines.

The failure rate per hour is significantly increased for adverse weather. Some examples given in [Billinton and Wenyuan, 1991] show that the number of outages per year can be multiplied by a factor 30. In this work, the outage rates per year will be multiplied by 30 in case of adverse weather for all the lines. The outage rates per year for the cables (branches 1 and 9) and the transformers are not impacted by the weather.

Two examples of the probabilities of unavailability can be found in Table B.2. The first example (line 1) is a cable and therefore the probability of unavailability is independent of the weather. The second example (line 22) is a typical line and both probabilities of unavailability are shown. The transmission lines outage probabilities for all the lines and both weather conditions can be found in [Duchesne, 2016, Appendix C].

Table B.2: Number of outages per hour λ and probability of unavailability for $t = 12$, 24 and 36h for the cable 1 and the transmission line 22. λ for cable 1 is identical for both weather status.

Line	λ [outage/h]	P_{outage} $(t = 12h)$	P_{outage} $(t = 24h)$	P_{outage} $(t = 36h)$
1 (both)	$2.74\ 10^{-5}$	$3.29\ 10^{-4}$	$6.57\ 10^{-4}$	$9.86\ 10^{-4}$
22 (normal)	$4.68\ 10^{-5}$	$5.62\ 10^{-4}$	$1.12\ 10^{-3}$	$1.68\ 10^{-3}$
22 (adverse)	0.00140	0.0167	0.0331	0.0493

The outage probabilities of the transformers for different values of t can be observed in Table B.3. The number of outages per hour is identical for the 5 phase-shifting transformers.

Table B.3: Number of outages per hour λ and probability of unavailability for $t = 12$, 24 and 36h for a phase-shifting transformer.

λ [outage/h]	$P_{outage}(t = 12h)$	$P_{outage}(t = 24h)$	$P_{outage}(t = 36h)$
$2.28\ 10^{-6}$	$2.74\ 10^{-5}$	$5.48\ 10^{-5}$	$8.22\ 10^{-5}$

B.4 Probabilities of contingencies

The considered set of contingencies is composed of 7 (common mode) double outages and 38 single outages. The contingency $c = 1$ is the no outage case, therefore there are 46 contingencies. The 7 common outages correspond to the lines circled in Figure 5.2.

Given the mean time to failure (MTTF) per line and per common outage, one can compute the probability of occurrence of only one contingency. A MTTF is also known for common outage contingencies.

The probability of having at least one occurrence of contingency c in 1 hour given the MTTF is:

$$P(\text{failure}_c) = 1 - e^{-\lambda_c \Delta t},$$

with $\Delta t = 1\text{h}$ and $\lambda_c = \frac{1}{MTTF_c}$ for $c = 2, ..., 46$ for the 38 single outages and the 7 common outages. The Poisson law is again chosen, as for the determination of generating units and transmission lines availability.

The probability of no failure is thus the product of the probabilities for each line to be in service and for each common outages to not occur:

$$P(\text{no failure}) = \prod_{c=2}^{46} (1 - P(\text{failure}_c)). \tag{B.3}$$

We need to compute the probability that only contingency c occurs. It is given by the probability that the failure described by contingency c occurs multiplied by the probability that all the other possible failures do not occur. This gives:

$$P(\text{cont}_c) = P(\text{failure}_c) \cdot \prod_{j=2, j\neq c}^{46} (1 - P(\text{failure}_j)). \tag{B.4}$$

It is worth mentioning that by considering a set of 46 contingencies, a lot of contingencies are neglected, such as the combination of 3 lines out-of-service, 4 lines out-of-service and so on. Two possibilities are considered. Either their criticality is neglected and they are incorporated in the no outage case. Or they are all considered as a unique contingency where their associated criticality could be the worse criticality computed among the 46 studied contingencies and their probability is 1 minus the sum of the probabilities of the 46 studied contingencies. In this work, the first possibility is chosen for simplicity and thus the risk is underestimated. Therefore, the probability of having no outage is not equal to $P(\text{no failure})$. It is slightly increased to have $\sum_{c\in\mathcal{C}} \pi_c = 1$, where $\mathcal{C}$ contains the 46 contingencies previously introduced:

$$P(\text{cont}_1) = 1 - \sum_{c\in\mathcal{C}, c\neq 1} P(\text{cont}_c).$$

The probabilities of contingencies are modified for two reasons: the weather and the availability of the lines.

Concerning the weather, the number of failures per hour λ is multiplied by 30 in case of adverse weather, except for the cables (lines 1 and 9) and the 5 transformers.

In case of unavailability of particular lines, the probabilities of outage of these lines are equal to 1. If the unavailable line was part of a double common outage contingency, this has no influence on the probability of the double common outage. This probability is equal to 1 only when both lines are unavailable.

If at least one line is unavailable, the definition of the no outage case should change. In fact, the probability of no outage is 0 since some factors in equation (B.3) are equal to 0. Thus a redefinition of the no failure probability is needed: $P(\text{no failure})$ is the probability to have no failure of the available lines or available common outages. If $\mathcal{C}_{available}$ is the set of contingencies that represent the available common outages and the failure of the available lines and transformers , $P(\text{no failure})$ is computed as:

$$P(\text{no failure}) = \prod_{c\in\mathcal{C}_{available}} (1 - P(\text{failure}_c)).$$

Note that the contingency 1, corresponding to the no outage case, is not included in $\mathcal{C}_{available}$. For each $c \in \mathcal{C}_{available}$, the probability for only contingency c to occur $P(\text{cont}_c)$ is also slightly modified compared to equation (B.4):

$$P(\text{cont}_c) = P(\text{failure}_c) \cdot \prod_{j\in\mathcal{C}_{available}, j\neq c} (1 - P(\text{failure}_j)).$$

The contingencies that are not included in the set $\mathcal{C}_{available}$ describe the failure of the same lines, thus the same system topology. In that case, to avoid taking into account the same expected criticality several times, the probabilities of the redundant contingencies are set to 0. Thus,

$$P(\mathrm{cont}_c) = 0, \forall c \notin \mathcal{C}_{available} \ \& \ c \neq 1.$$

For instance, if the line 1 is unavailable, contingency 1 (no outage case) and 2 (fault on line 1) describe the same system topology, that is a system without line 1. The probability of contingency 2 is thus set to 0.

Finally, the probability of the no outage case, defined here as no failure of available lines and no available common outages is

$$P(\mathrm{cont}_1) = 1 - \sum_{c \in \mathcal{C}_{available}} P(\mathrm{cont}_c).$$

B.5 Model of load uncertainty

The error on the load for day-ahead forecast is assumed to follow a normal distribution [Billinton and Allan, 1996] with a mean equal to 0. In order to take into account both the correlation in the fluctuation of the different bus loads and their independence, two error terms are added to the prediction value. The first one is common to all the loads and the second is independent. In other words, for the bus load i, the realization load $\tilde{p}_i$ is given by:

$$\tilde{p}_i = (1 + \epsilon_{\alpha_d} + \epsilon_{\beta_{d,i}}) \cdot \hat{p}_i, \tag{B.5}$$

where $\hat{p}_i$ is the predicted load of bus i, ϵ_{α_d} is constant for all buses and drawn randomly according to a normal law $\mathcal{N}(0, \alpha_d)$ and $\epsilon_{\beta_{d,i}}$ is different for each bus and drawn according to a law $\mathcal{N}(0, \beta_d)$.

B.5.1 Determination of parameters α_d and β_d

In order for the realizations to be close to what could be encountered in practice, the parameters α_d and β_d are determined according to values found in the literature. In a study realized by the NREL (National Renewable Energy Laboratory) in 2012, a normalized standard deviation $\sigma_{d,global}$ of about 0.03 was found for the total day-ahead load forecast error [Hodge et al., 2012a]. The normalized standard deviation is defined as the standard deviation divided by the yearly average load. The error on a single bus load is assumed to have a greater standard deviation. It is arbitrarily defined as $\sigma_{d,local} = 0.1$.

Considering one area of the IEEE-RTS96 network with 17 loads [Grigg et al., 1999], the variance of the total load realization can be computed:

$$\begin{aligned}
& \textstyle\sum_{i=1}^{17} \tilde{p}_i = \sum_{i=1}^{17} \hat{p}_i (1 + \epsilon_{\alpha_d} + \epsilon_{\beta_{d,i}}) \\
\Leftrightarrow \quad & \operatorname{var}\left(\textstyle\sum_{i=1}^{17} \tilde{p}_i\right) = \operatorname{var}\left(\textstyle\sum_{i=1}^{17} \hat{p}_i (1 + \epsilon_{\alpha_d} + \epsilon_{\beta_{d,i}})\right) \\
\Leftrightarrow \quad & \operatorname{var}\left(\textstyle\sum_{i=1}^{17} \tilde{p}_i\right) = \operatorname{var}\left(\epsilon_{\alpha_d} \textstyle\sum_{i=1}^{17} \hat{p}_i\right) + \operatorname{var}\left(\textstyle\sum_{i=1}^{17} \hat{p}_i \epsilon_{\beta_{d,i}}\right) \\
\Leftrightarrow \quad & \operatorname{var}\left(\textstyle\sum_{i=1}^{17} \tilde{p}_i\right) = \left(\textstyle\sum_{i=1}^{17} \hat{p}_i\right)^2 \alpha_d^2 + \operatorname{var}\left(\textstyle\sum_{i=1}^{17} \hat{p}_i \epsilon_{\beta_{d,i}}\right).
\end{aligned}$$

Considering that $\hat{p}_i = l_i \hat{p}_{tot}$ for the bus load i with $l_i \geq 0$ and $\sum_{i=1}^{17} l_i = 1$ and $\hat{p}_{tot}$ being the total predicted load, it is possible to simplify this expression. Indeed, the variables l_i are constant for each hour (as defined in [Grigg et al., 1999]). One can write:

$$\begin{aligned}
&\Leftrightarrow \quad \operatorname{var}\left(\sum_{i=1}^{17} \tilde{p}_i\right) = \left(\sum_{i=1}^{17} \hat{p}_i\right)^2 \alpha_d^2 + \operatorname{var}\left(\sum_{i=1}^{17} l_i \hat{p}_{tot} \epsilon_{\beta_{d,i}}\right)\\
&\Leftrightarrow \quad \operatorname{var}\left(\sum_{i=1}^{17} \tilde{p}_i\right) = \left(\hat{p}_{tot}\right)^2 \alpha_d^2 + (\hat{p}_{tot})^2 \sum_{i=1}^{17} l_i^2 \beta_d^2\\
&\Leftrightarrow \quad \left(p_{avg}\right)^2 \sigma_{d,global}^2 = \left(\hat{p}_{tot}\right)^2 \alpha_d^2 + (\hat{p}_{tot})^2 \sum_{i=1}^{17} l_i^2 \beta_d^2\\
&\Leftrightarrow \quad \left(\hat{p}_{tot}\right)^2 \sigma_{d,global}^2 \simeq \left(\hat{p}_{tot}\right)^2 \alpha_d^2 + (\hat{p}_{tot})^2 \sum_{i=1}^{17} l_i^2 \beta_d^2\\
&\Leftrightarrow \quad \sigma_{d,global}^2 \simeq \alpha_d^2 + \sum_{i=1}^{17} l_i^2 \beta_d^2.
\end{aligned}$$

p_{avg} is defined as the average load over a year. The approximation that $\hat{p}_{tot} \simeq p_{avg}$ is used to simplify the calculations.

The variance of the load on bus load i can be computed as:

$$\begin{aligned}
& \tilde{p}_i = \hat{p}_i(1 + \epsilon_{\alpha_d} + \epsilon_{\beta_{d,i}})\\
\Leftrightarrow \quad & \operatorname{var}(\tilde{p}_i) = \operatorname{var}\left(\hat{p}_i(1 + \epsilon_{\alpha_d} + \epsilon_{\beta_{d,i}})\right)\\
\Leftrightarrow \quad & \sigma_{d,local}^2 (\hat{p}_i)^2 = (\hat{p}_i)^2 \alpha_d^2 + (\hat{p}_i)^2 \beta_d^2\\
\Leftrightarrow \quad & \sigma_{d,local}^2 = \alpha_d^2 + \beta_d^2
\end{aligned}$$

Finally, one obtains a system of two equations with two unknowns:

$$\begin{cases} \sigma_{d,global}^2 & = \quad \alpha_d^2 + \sum_{i=1}^{17} l_i^2 \beta_d^2 \\ \sigma_{d,local}^2 & = \quad \alpha_d^2 + \beta_d^2 \end{cases}$$

Injecting the values of $\sigma_{d,global}$, $\sigma_{d,local}$ and l_i, we have $\alpha_d = 0.0144$ and $\beta_d = 0.09896$. This is approximated by $\alpha_d = 0.015$ and $\beta_d = 0.1$.

There is no change in the demand in function of the state of the weather. The weather influence is included in the normal distribution of forecast error.

The day chosen to determine the load forecast values in chapter 5 is a Tuesday in Winter.

B.6 Model of wind generation uncertainty

The forecasting of wind farms production is one of the main issues of operation planning. The usual approximation is to consider that the forecast error follows a normal distribution as for the load. However several studies showed that this is not the case. In [Hodge et al., 2012b], wind forecast errors from several countries were analyzed and it was concluded that a hyperbolic distribution could better fit the error than a normal distribution. Another proposal was to use a Beta pdf [Bludszuweit et al., 2008].

Given the difficulty to obtain some real parameters for these two laws, a normal distribution is chosen and is defined in the same way as for the load forecast error. The realized generation $\tilde{g}_i$ for the wind farm i is thus given by:

$$\tilde{g}_{w,i} = (1 + \epsilon_{\alpha_w} + \epsilon_{\beta_{w,i}}) \cdot \hat{g}_{w,i},$$

where $\hat{g}_{w,i}$ is the predicted production of wind farm i, ϵ_{α_w} is constant for all wind farms and drawn randomly according to a normal law $\mathcal{N}(0, \alpha_w)$ and $\epsilon_{\beta_{w,i}}$ is different for each wind farm and drawn according to a law $\mathcal{N}(0, \beta_w)$.

B.6.1 Determination of parameters α_w and β_w

According to a study of the NREL, a normalized standard deviation $\sigma_{w,global} = 0.13$ was obtained for the total day-ahead wind power forecasting errors of the California Independent System Operator (CAISO) region [Hodge et al., 2012a]. The normalized standard deviation was obtained by calculating the standard deviation and dividing it by the system maximum capacity. Another study indicates a variance of 60.4MW^2 for a typical 30 MW-capacity wind farm from the western wind and solar integration study data set [Florita et al., 2012]. This leads to a normalized standard deviation $\sigma_{w,local} = \frac{\sqrt{60.4}}{30} = 0.259$.

The same approach as for the load is used to estimate the parameters α_w and β_w defining the distribution of wind power forecast errors. We consider one area of the IEEE-RTS96 network, with the addition of 9 wind farms, as suggested in [Pandzic et al., nd]. The day chosen to determine the wind generation forecast values is the first day of the year with 'favorable' wind. Nevertheless, the repartition of production among the wind farms is different from one hour to the next. Therefore all the wind farms are considered to produce the same proportion of power for the sake of simplicity[1].

The variance of the total forecast generation error can be computed as:

$$\begin{aligned}
& \textstyle\sum_{i=1}^{9} \tilde{g}_{w,i} = \sum_{i=1}^{9} \hat{g}_{w,i}(1 + \epsilon_{\alpha_w} + \epsilon_{\beta_{w,i}}) \\
\Leftrightarrow\ & \text{var}\left(\textstyle\sum_{i=1}^{9} \tilde{g}_{w,i}\right) = \text{var}\left(\textstyle\sum_{i=1}^{9} \hat{g}_{w,i}(\epsilon_{\alpha_w} + \epsilon_{\beta_{w,i}})\right) \\
\Leftrightarrow\ & \text{var}\left(\textstyle\sum_{i=1}^{9} \tilde{g}_{w,i}\right) = \left(\textstyle\sum_{i=1}^{9} \hat{g}_{w,i}\right)^2 \alpha_w^2 + \text{var}\left(\textstyle\sum_{i=1}^{9} \hat{g}_{w,i}\epsilon_{\beta_{w,i}}\right) \\
\Leftrightarrow\ & (C_{w,tot})^2\, \sigma_{w,global}^2 = (\hat{g}_{w,tot})^2 \alpha_w^2 + (\hat{g}_{w,tot})^2 \left(\tfrac{1}{9}\right)^2 \beta_w^2 \\
\Leftrightarrow\ & (\hat{g}_{w,tot})^2\, \sigma_{w,global}^2 \simeq (\hat{g}_{w,tot})^2 \alpha_w^2 + (\hat{g}_{w,tot})^2 \left(\tfrac{1}{9}\right)^2 \beta_w^2 \\
\Leftrightarrow\ & \sigma_{w,global}^2 \simeq \alpha_w^2 + \tfrac{1}{81}\beta_w^2
\end{aligned}$$

Note that $\sigma_{w,global}$ is defined as the standard deviation divided by the capacity of the system and not the forecast production. In this computation, it is considered that $\hat{g}_{w,tot} \simeq C_{w,tot}$ where $C_{w,tot}$ is the maximum wind power generation. The reason is that $\hat{g}_{w,tot}$ is different for each hour and it is a way to have a solution independent of the predicted value.

The variance of the forecast generation error for one wind farm is:

$$\sigma_{w,local}^2 = \alpha_w^2 + \beta_w^2.$$

It is the same reasoning as for the load and the approximation $\hat{g}_{w,i} = C_{w,i}$ where $C_{w,i}$ is the capacity of wind farm i is used.

The resolution of the system :

$$\begin{cases} \sigma_{w,global}^2 &= \alpha_w^2 + \frac{1}{81}\beta_w^2 \\ \sigma_{w,local}^2 &= \alpha_w^2 + \beta_w^2 \end{cases}$$

[1]Note that this assumption is relaxed to generate the scenarios for the case studies in chapters 6 and 7. As a consequence, the coefficient 1/9 that appears in the presented uncertainty model is adapted for each wind farm i (and each hour of the day) to correspond to the true repartition of production and therefore $\beta_{w,i}$ is different not only for each hour but also for each wind farm i.

gives $\alpha_w = 0.127$ and $\beta_w = 0.225$. This is approximated by $\alpha_w = 0.13$ and $\beta_w = 0.23$.

However, it appeared that the approximations performed were not valid. Given $\hat{g}_{w,tot}$ for each hour and still considering that $\hat{g}_{w,i} = \hat{g}_{w,tot}/9$, a less approximated system of equations is:

$$\begin{cases} \sigma^2_{w,global} & = \frac{\hat{g}^2_{w,tot}}{C^2_{w,tot}}\left(\alpha^2_w + \frac{1}{81}\beta^2_w\right) \\ \sigma^2_{w,local} & = \frac{\hat{g}^2_{w,tot}/81}{C^2_{w,tot}/81}\left(\alpha^2_w + \beta^2_w\right) = \frac{\hat{g}^2_{w,tot}}{C^2_{w,tot}}\left(\alpha^2_w + \beta^2_w\right) \end{cases} \tag{B.6}$$

Solving this system leads to different values of α_w and β_w per hour. It gives better results in term of variance but the solution is not general and depends on the predicted wind power for each hour.

Using this method to generate the samples sometimes leads to negative values or values greater than the maximum capacity. In order to create realistic samples, the wind output power is set to 0 if the value is negative and is equal to the maximum capacity ($C_{w,i}$) if the value is greater than $C_{w,i}$. However this has the effect of decreasing the variance and modifying the mean.

It is worth noting that the large difference in relative standard deviation for different hours is due not only to the truncation of values but also to the approximation that each wind farm produces the same amount of power. Depending on the hour, this assumption is more or less correct.

C

Additional results

Overview

In this appendix, we show some additional results considering the case study of chapter 7. In particular, we compare the results of the leave-one-out experiment with and without the active power level features and we present results relative to an improved version of the validation protocol of the ranking methods.

C.1 Impact of generator active power levels as features on proxies performances

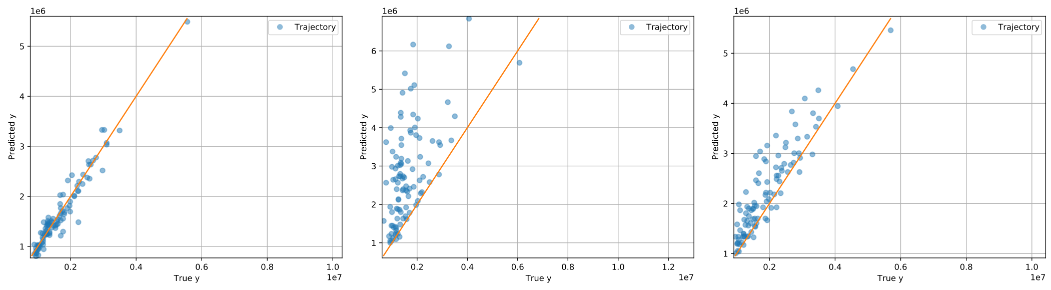
In chapter 7, we did not to use the generator active power levels as features of our proxies to describe candidate day-ahead decisions. In this appendix, we justify this choice. In particular, we present an experiment showing that using these features with our setting can worsen the generalization performances of the learnt proxies and we analyze why it can be the case. We also present several solutions to avoid this problem and explain why we chose to remove these variables.

To show the impact of active power levels as features of our proxies, we use the leave-one-decision-out experiment presented in section 7.3.4.1. We realize this experiment with the active power levels as features and we compare the resulting R^2-score on the test sets. We chose the combination of meta-parameters that maximizes the validation score. The results for the 20 decisions can be seen in the first 'Test score' column of Table C.1.

One can notice that the next-day total preventive control cost is not well predicted for decisions 2 and 17 when they are not in the training set. This is due to the fact that for these two decisions some of the generator active power levels are very different compared to other decisions, leading to a significant difference in the distribution of these variables between the training set and the test set.

To illustrate this, let us consider decision 17. The total preventive control cost of this decision is well predicted at each hour except at hour 4, where the prediction systematically overestimates the true value, resulting most of the time in an overestimation of the trajectory total preventive control cost, as can be seen in Figure C.1c. When analyzing the feature values, it can be identified that it is due to generators 10 and 11. The active power level of these generators is either equal to 0 or 25MW depending on the hour and the decision, except at hour 4 for decision 17, where it is equal to 100MW. This value is much larger than the maximum value seen by the network during training for these two features, and is not handled well by the network during the test phase, leading in this case to an overestimation of the target output.

Similar phenomena can be observed for decision 2, resulting also in an overestimation of the total preventive control cost (see Figure C.1b).



(a) Decision 1 (b) Decision 2 (c) Decision 17

Figure C.1: True vs predicted total preventive control cost of respectively unseen decision 1, 2 and 17 with unseen scenarios, when the active power levels are used as features.

This problem could be tackled by increasing the number and variety of decisions in the training set, in order to correctly span the range of values of each input variable in the training set and avoid this large difference of distribution between the training and the test sets. This would help the neural network to better learn the impact of these variables.

However, given the computational burden linked to generating trajectories for new decisions, we chose to remove these variables when learning and exploiting the proxies. The information about candidate decisions is therefore given by the difference between the realized and the forecast scenarios (since to one candidate decision corresponds one forecast scenario), as well as the minimum and maximum total generation capacities. The advantage of using the differences between the real-time and forecast scenarios is that the distribution of these input variables does not vary as much as for the active power outputs of the generating units from one decision to another. However, removing these input variables can remove relevant information to describe the day-ahead decisions. Hopefully, in our setting, one can see in the second 'Test score' column of Table C.1 that the decrease of performance of the proxies is limited. For several unseen decisions, the generalization score is even improved.

For the sake of completeness, we mention another solution to avoid this problem, which would have been to use a tree-based algorithm such as ET. These algorithms are indeed less sensible to a change in the range of feature values between the training and test sets. However, the proxies built with the ET algorithm ($k = p$, $M = 100$ and $n_{min} = 4$) have smaller scores than the proxies built with an NN algorithm without the active power levels of generating units, as can be seen in Table C.1. It is why we did not use this solution in chapter 7.

Table C.1: Trajectory R^2-scores obtained over the 20 left-out folds (with unseen scenarios) of the leave-one-decision-out experiment with the NN or ET algorithms and with and without the generator active power levels as features of the proxies.

Unseen decision	Test score		
	NN with power levels	NN without power levels	ET with power levels
1	0.936	0.879	0.755
2	-2.938	0.906	0.747
3	0.903	0.886	0.702
4	0.902	0.882	0.747
5	0.891	0.916	0.767
6	0.903	0.877	0.700
7	0.928	0.940	0.806
8	0.892	0.888	0.783
9	0.906	0.870	0.806
10	0.923	0.930	0.786
11	0.901	0.884	0.761
12	0.915	0.922	0.815
13	0.888	0.907	0.661
14	0.916	0.899	0.737
15	0.925	0.886	0.783
16	0.891	0.880	0.757
17	0.655	0.897	0.794
18	0.946	0.922	0.803
19	0.911	0.894	0.733
20	0.922	0.912	0.794

Note that this problem and the chosen solution are linked to our particular setting, where to one forecast scenario corresponds one candidate decision. If the candidate decisions are generated differently, it may be possible that using the generator active power levels does not create any generalization problems.

C.2 Using the proxies for ranking day-ahead decisions - updated results

In this section, we compare the two validation protocols of the ranking methods mentioned in chapter 7. The first one, for which the detailed results were already presented in chapter 7, is based on a ground truth ranking computed from 600 scenarios, while the second one is an improved protocol for which the ground truth ranking is computed with 2000 independent scenarios. Note that for the second validation protocol, we chose to also exploit the proxy to estimate the expected next-day operation cost for each decision $\delta^i_{da} \in \Delta_s$, to show both methods estimates for all decisions.

The Kendal's tau coefficients and Spearman's correlation coefficients comparing the rankings obtained with the proxies and the two ground truths can be found in Table C.2, while Table C.3 and Figure C.2 show for the best case experiment with 5 decisions in the training set the estimated expected total preventive control costs and the corresponding rankings compared to the two ground truths.

Table C.2: Minimum and maximum Kendal's tau coefficient and Spearman's correlation coefficient for both estimation methods, with 5 or 10 decisions in the learning set, for both the previous ground truth with 600 scenarios and the updated one with 2000 independent scenarios.

		Previous ground truth				Updated ground truth			
		$\hat{\mu}_{y_p}$		$\hat{\mu}^y_{y_p}$		$\hat{\mu}_{y_p}$		$\hat{\mu}^y_{y_p}$	
		Min	Max	Min	Max	Min	Max	Min	Max
5	τ	0.5158	0.8421	0.8211	0.9263	0.4842	0.7263	0.8947	0.9473
dec.	ρ	0.7038	0.9534	0.9338	0.9835	0.8616	0.9383	0.9699	0.9880
10	τ	0.6211	0.8632	0.8211	0.9159	0.5158	0.8316	0.9158	0.9579
dec.	ρ	0.7624	0.9654	0.9353	0.9820	0.8677	0.9353	0.9759	0.9925

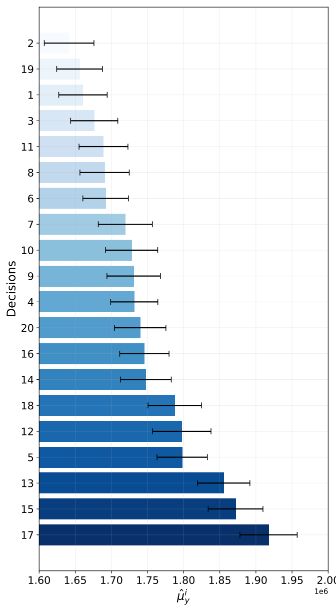
The first observation to be made is that the conclusions drawn with the first validation protocol are indeed validated by the improved validation protocol. Both rankings obtained from the proxies are close to the updated true ranking. In particular, the less costly candidate decisions and the more costly ones are correctly identified with both methods and the control variates approach clearly outperforms method 1. Furthermore we see that the first decision in both estimated rankings is this time the correct decision.

When comparing the rankings with the two ground truths in Tables C.2, one can see that the rankings obtained with method 1 are closer to the ground truth obtained from the 600 scenarios than to the ground truth obtained from the 2000 independent scenarios. When we compare the estimates $\hat{\mu}^i_y$ and $\hat{\mu}^i_{y,2000}$ of the expected total preventive control cost, we see that $\hat{\mu}^i_y$ is smaller than $\hat{\mu}^i_{y,2000}$ for all $i = 1, .., 20$. Since the estimates $\hat{\mu}^i_{y_p}$ were already mostly underestimating $\hat{\mu}^i_y$, their negative biases increase when considering $\hat{\mu}^i_{y,2000}$ as the ground truth. These systematic biases are partly corrected with the control variates approach. The rankings obtained with this method are even closer to the independent ground truth ranking $r(\hat{\mu}^i_{y,2000})$ than to the previous ground truth ranking $r(\hat{\mu}^i_y)$.

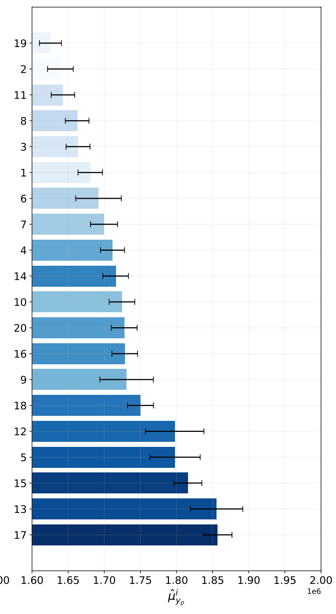
Table C.3: True and estimated expected total preventive control cost per decision and the associated ranking $r(\cdot)$. The decisions used to learn the proxies are colored in red. Upper table: previous ground truth. Lower table: updated one.

δ^i	$\hat{\mu}_y^i$ $(\times 10^6)$	$\hat{\mu}_{yp}^i$ $(\times 10^6)$	$\hat{\mu}_{yp}^{y,i}$ $(\times 10^6)$	$r(\hat{\mu}_y^i)$	$r(\hat{\mu}_{yp}^i)$	$r(\hat{\mu}_{yp}^{y,i})$
2	1.641	1.639 $-$	1.635 $-$	2	19 $\wedge_1$	19 $\wedge_1$
19	1.656	1.625 $-$	1.626 $-$	19	2 $\vee_1$	2 $\vee_1$
1	1.661	1.681 $+$	1.646 $-$	1	11 $\wedge_2$	1 $=$
3	1.676	1.664 $-$	1.664 $-$	3	8 $\wedge_2$	3 $=$
11	1.689	1.643 $-$	1.672 $-$	11	3 $\vee_1$	11 $=$
8	1.691	1.662 $-$	1.706 $+$	8	1 $\vee_3$	6 $\wedge_1$
6	1.692	1.692 $\circ$	1.692 $\circ$	6	6 $=$	8 $\vee_1$
7	1.719	1.700 $-$	1.716 $-$	7	7 $=$	7 $=$
10	1.728	1.725 $-$	1.742 $+$	10	4 $\wedge_2$	9 $\wedge_1$
9	1.731	1.731 $\circ$	1.731 $\circ$	9	14 $\wedge_4$	10 $\vee_1$
4	1.732	1.711 $-$	1.761 $+$	4	10 $\vee_2$	4 $=$
20	1.740	1.728 $-$	1.788 $+$	20	20 $=$	20 $=$
16	1.746	1.728 $-$	1.796 $+$	16	16 $=$	16 $=$
14	1.748	1.716 $-$	1.807 $+$	14	9 $\vee_4$	12 $\wedge_2$
18	1.788	1.750 $-$	1.810 $+$	18	18 $=$	5 $\wedge_2$
12	1.797	1.797 $\circ$	1.797 $\circ$	12	12 $=$	14 $\vee_2$
5	1.798	1.798 $\circ$	1.798 $\circ$	5	5 $=$	18 $\vee_2$
13	1.855	1.855 $\circ$	1.855 $\circ$	13	15 $\wedge_1$	13 $=$
15	1.872	1.816 $-$	1.871 $-$	15	13 $\vee_1$	15 $=$
17	1.917	1.857 $-$	1.905 $-$	17	17 $=$	17 $=$

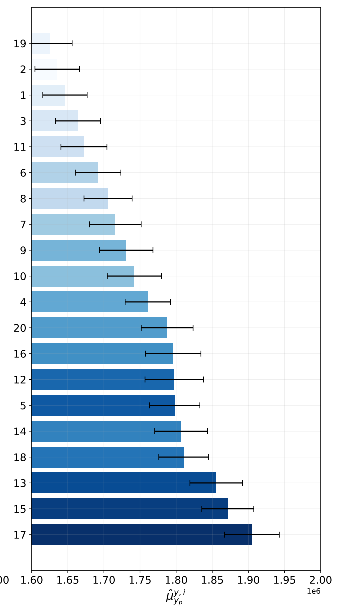
δ^i	$\hat{\mu}_{y,2000}^i$ $(\times 10^6)$	$\hat{\mu}_{yp}^i$ $(\times 10^6)$	$\hat{\mu}_{yp}^{y,i}$ $(\times 10^6)$	$r(\hat{\mu}_{y,2000}^i)$	$r(\hat{\mu}_{yp}^i)$	$r(\hat{\mu}_{yp}^{y,i})$
19	1.670	1.625 $-$	1.626 $-$	19	19 $=$	19 $=$
2	1.672	1.639 $-$	1.635 $-$	2	2 $=$	2 $=$
1	1.675	1.681 $+$	1.646 $-$	1	6 $\wedge_4$	1 $=$
11	1.685	1.643 $-$	1.672 $-$	11	11 $=$	3 $\wedge_1$
3	1.696	1.664 $-$	1.664 $-$	3	8 $\wedge_1$	11 $\vee_1$
8	1.709	1.662 $-$	1.706 $-$	8	3 $\vee_1$	8 $=$
6	1.709	1.641 $-$	1.721 $+$	6	9 $\wedge_2$	9 $\wedge_2$
7	1.749	1.700 $-$	1.716 $-$	7	1 $\vee_5$	7 $=$
9	1.750	1.664 $-$	1.712 $-$	9	7 $\vee_1$	6 $\vee_2$
10	1.770	1.725 $-$	1.742 $-$	10	4 $\wedge_1$	10 $=$
4	1.781	1.711 $-$	1.761 $-$	4	14 $\wedge_4$	4 $=$
20	1.794	1.728 $-$	1.788 $-$	20	10 $\vee_2$	5 $\wedge_2$
16	1.795	1.728 $-$	1.796 $+$	16	20 $\vee_1$	20 $\vee_1$
5	1.796	1.769 $-$	1.762 $-$	5	16 $\vee_1$	16 $\vee_1$
14	1.809	1.716 $-$	1.807 $-$	14	18 $\wedge_1$	14 $=$
18	1.821	1.750 $-$	1.810 $-$	18	5 $\vee_2$	18 $=$
12	1.853	1.813 $-$	1.827 $-$	12	13 $\wedge_2$	12 $=$
15	1.896	1.816 $-$	1.871 $-$	15	12 $\vee_1$	15 $=$
13	1.922	1.811 $-$	1.872 $-$	13	15 $\vee_1$	13 $=$
17	1.935	1.857 $-$	1.905 $-$	17	17 $=$	17 $=$



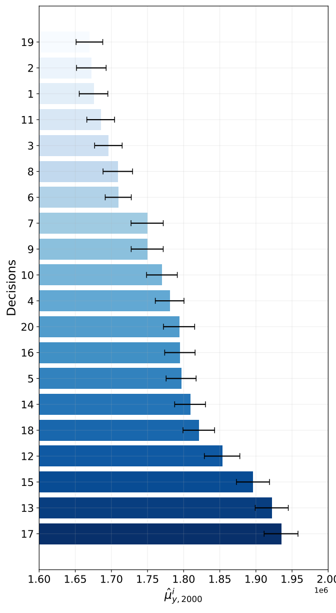
(a) Ground truth $\hat{\mu}^i_y$



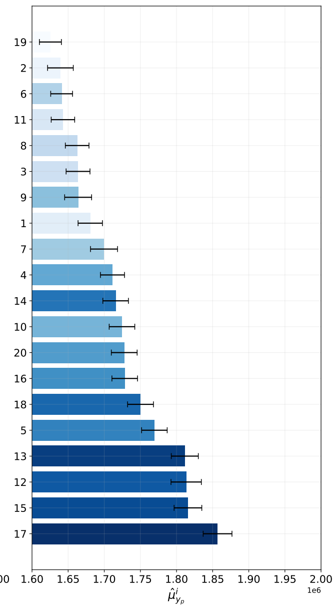
(b) Proxy $\hat{\mu}^i_{y_p}$



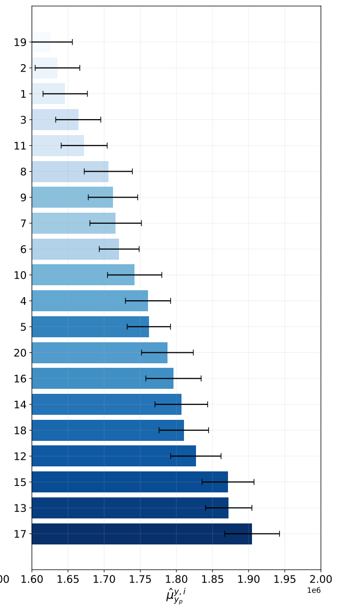
(c) CV approach $\hat{\mu}^{y,i}_{yp}$



(d) Ground truth $\hat{\mu}^i_{y,2000}$



(e) Proxy $\hat{\mu}^i_{y_p}$



(f) CV approach $\hat{\mu}^{y,i}_{yp}$

Figure C.2: Figure (a) and (d) are the ground truths, computed respectively with 600 and 2000 scenarios. In both figures, the decisions are sorted in crescent order of expected total preventive control cost and the colors of the decisions are of crescent intensity while following the order of the decisions. Figures (b)-(e) and (c)-(f) correspond respectively to the proxy method and the control variates method, and the decisions are sorted in crescent order. The colors of the decisions of (b) and (c) are identical to (a) and the ones of (e) and (f) are identical to (d) to help visualizing the differences in the obtained rankings compared to the two ground truths. The error bars show the $\pm\hat{\sigma}^i/\sqrt{n}$ interval, where $\hat{\sigma}^i$ is the sample estimate of the standard deviation of y^i (a,d), y^i_p (b,e) and $(y^i - y^i_p)$ (c,f) and n is respectively equal to 600 (a), 2000 (b,d,e) and 100 (c,f).