Automatic learning approaches for electric power systems

L. Wehenkel
University of Liège
Department of Electrical and Computer Engineering
Sart-Tilman B28, B-4000 Liège, Belgium

August 28, 2001

Abstract

This chapter describes a methodology based on the combination of probabilistic reasoning, automatic learning and Monte-Carlo simulations, which has been used extensively for the study of electric power systems. The first part describes the generic approach together with the principles of the main classes of automatic learning methods. The second part discusses a few real-life applications and some new research directions. The chapter concludes with a discussion of the usefulness of the proposed approach and its applicability to the study of complex systems, in general.

1 Context and statement of scope of the paper

This paper summarizes research and development work which started in the mid eighties.

The practical problem which has driven the research was electric power systems dynamic security assessment. Electric power systems are essentially large, complex nonlinear systems which have significantly grown in size and importance during the last 50 years. At the beginning of this century, electric power systems were rather small isolated systems, connecting customers and power plants inside small geographical areas. When the technology became mature, it was realized that economies of scale and increased reliability would be possible if the small systems were interconnected among each other. Indeed, interconnection made possible to share very large power plants among different users, and reduced the amount of necessary stand-by reserves to serve plant outages. This was the beginning of a long period of transmission systems enhancement, resulting in bulk power systems stretching over whole continents. For example, the Western European interconnected system (UCPTE) covers a geographical area of about 2000 kilometers North to South by about 4000 kilometers West to East, and presently deserves energy to about 300 million citizens through more than 10 countries.

The very large size of electric power systems makes their understanding and the mastering of their reliability a quite complex problem. An interconnected system is operated by a number of independent companies which have to take decisions without knowing precisely what will be the strategy of their neighbors. Thus, electric power systems are also a good example of distributed decision making under uncertainties. Uncertainties are related to the external environment of the system as well as to its internal behavior. The external environments acting on the power system are both physical (e.g. meteorological effects) and socio-economic (demand for electricity, social role of electric energy, ecological trends, etc.); both are quite uncertain from the viewpoint of power engineers. For a decision maker, the internal behavior of the electric power system is uncertain because of partial knowledge about the system, existence of human factors, as well as because of the high system complexity.

Since the electric power systems are built by engineers, there has been a long tradition of analytical modeling, of system theory applications to power systems, and, in the last 30 years, of computer based numerical simulations.

In the late seventies and early eighties, it was believed that *Artificial Intelligence* (AI) techniques could provide solutions to help improve the performance of electric power systems. In this period, many pioneering papers have been published proposing to apply expert system approaches to various problems in designing and operating the power system. Some of this work has yielded actual applications in the mean-while. Pattern recognition, another AI technique, was first proposed for dynamic security assessment by Tom DyLiacco, in the late sixties [1, 2], and, since then, many researchers have worked on the topic, applying different techniques (statistical pattern recognition, neural networks and machine learning) to different power system problems (load forecasting, system identification and state estimation, stability assessment and control).

During the last ten years, an important amount of research work was also carried out in the field of automatic learning per se. In particular, automatic learning theory has reached by now a certain level of maturity and resulted in unifying the work carried out by different research schools, such as statistics, connectionnist systems, computer science and artificial intelligence. Data Mining is the most recent trend which contributed to render automatic learning popular among non-specialists. Data mining has emerged in the mid nineties, as an interdisciplinary field of applied research in response to the need for extracting meaningful information from ever growing databases. It has already some success stories, at least enough to make the field popular among practitioners.

Within this context, the research of the author of this contribution may be positioned as follows: in the mid eighties, we started to work on the application of machine learning to electric power system transient stability assessment; in the early nineties, we started collaborating with Electricité de France (a major utility in Europe) in order to find out how to apply the resulting methods to practical problems of economic and technical significance. This collaboration is still pursued at the time of writing this paper, almost ten years later. More importantly, a large amount of the work done at the University has actually been transferred towards practice and is today used within Electricité de France. More recently, a certain number of other utilities have started to consider the resulting approaches as valuable alternatives to their usual way of working. On our side, we believe that the present trend in liberalization and un-bundling of electric power systems will make the methodologies using automatic learning even more useful than in the past.

The methodology which is described in this paper is basically a "computer experiment" type of method. For a given power system and for a given practical problem, it uses Monte-Carlo simulations (coupled with existing power system simulators) to sample a large number of relevant power system dynamic scenarios, together with automatic learning (and data mining toolboxes) to extract synthetic information from the resulting databases.

Given the scope of the Academic Press Theme volumes within which this work is published, we aim at describing to non-specialists the following aspects : the basic principles underlying the approach; the main techniques which are used (with a stronger focus on those which may be classified under the AI theme); the practical interest of this work in the context of electric power systems; and the possible uses in a more general setting in order to make computer experiments to study complex systems from simulations. The last objective is motivated by our feeling that many other fields could benefit from the approach : mechanical engineering, chemical engineering, computer systems, telecommunications, etc.

The paper is organized as follows. Section 2 is a general description of the framework in a sufficiently general language to make it clear that it can be applied to a large number of problems; it focuses on the framework description. Section 3 describes in some detail a certain number of complementary automatic learning methods of general interest. Section 4 reviews a certain number of practical applications in the context of electric power system engineering and Section 5 provides general conclusions, directions for further research and discusses the applicability of this work outside the field of electric power systems.

2 Framework

This section introduces a general framework for the study of complex systems. The framework combines system theory methods with probabilistic reasoning and automatic learning from artificial intelligence. The overall approach is able to build simple models and decision rules

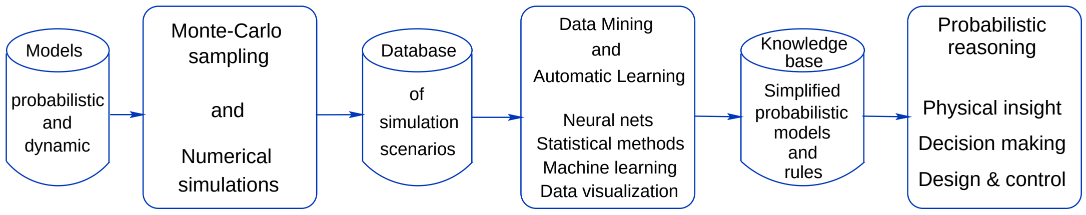


Figure 1: Overall probabilistic framework based on automatic learning

which may be used for the design, monitoring, and control of complex systems in uncertain environments.

The principle of the framework is depicted in figure 1. It starts with the specification of models (probabilistic and system theory type models), proceeds with the automatic generation of a database of system behaviors by Monte-Carlo simulations, and further with the extraction of synthetic information from this database by automatic learning. The last step of the approach consists in using the extracted information in order to acquire a better understanding of the system behavior and to make decisions for design and control.

Note. Although we focus in this paper mainly on situations where, as depicted in figure 1, the databases are generated by Monte-carlo simulations, it is of course possible to apply automatic learning to databases collected from actual measurements. This distinction will be further illustrated in section 4.

2.1 Probabilistic reasoning

As we will see, the core of the framework is provided by probabilistic reasoning, which we will introduce in an abstract although intuitive way. Probabilistic reasoning uses a probabilistic model of the reality and probability theory (theorems and techniques) to exploit this model.

2.1.1 Probabilistic models

Probabilistic modeling starts with the definition of a set of possible objects called the *universe*. These objects represent the possible behaviors[1] of the physical world which is studied. To make things as simple as possible, we will assume that the universe is finite (even though it may be very large). This is justified by the fact that we will consider the modeling of system behavior using digital computers, which can only represent a finite number of possible behaviors.

Hence, in the context of the analysis and control of a given complex system, the universe represents all the possible ways this system may behave during a certain time period, taking into account the possible external inputs and internal uncertainties. In the rest of this text we will also use the term *scenario* to denote such a hypothetical behavior of a given system.

The second step in probabilistic modeling consists in defining a family of subsets of the universe which may be observed (these are called *events* in the probabilistic literature). Again, we will assume for the sake of simplicity that in our case this collection contains every possible subset of the universe[2].

The last step of probabilistic modeling consists of assuming a probability measure. This is a function[3] which assigns to each subset in the collection of observable subsets a number between 0 and 1 representing its probability, where 0 means impossibility and 1 certainty.

2.1.2 Conditioning

Conditioning is the basic tool in probability theory to exploit observations.

Let A be a event (a subset of the universe). Then if A occurs (which means that the observed or assumed system behavior belongs to this subset), the probabilities of all events are updated in the following way :

$$P(B) \to P(B|A) = \frac{P(B \cap A)}{P(A)}, \tag{1}$$

where $P(B|A)$ denotes the probability of event B given the fact that event A has occurred.

Thus all events which are disjoint from A become impossible. On the other hand, among the events which intersect with A, some become more probable and some other less. The events whose probability remains unchanged ($P(B) = P(B|A)$) are said to be independent of A.

One can argue that this way of defining the conditional probability measure is actually the only rational possibility, yielding completely consistent reasoning procedures [3]. Probability theory thus provides a consistent and sound framework to model reasoning under uncertainties.

2.1.3 Interpretations

As such, probability calculus does not impose any particular physical interpretation. Actually, several interpretations coexist and are used in practice. We will merely recall the classical and the Bayesian views.

In the classical “objectivistic” interpretation, probabilities are viewed as limit values of observed frequencies, and could be determined (up to certain level of accuracy) by observing the physical system behaviors and counting the number of occurrences which belong to the different subsets of the universe.

In the Bayesian “subjectivistic” interpretation, probabilities are used to model the state of knowledge of a user. Thus, different users may use different probability measures if they have different knowledge about a system. Also, a given user may in principle revise his probability measure, when his state of knowledge evolves. Thus, in the Bayesian interpretation, probability theory is a tool (among other candidate tools which have been proposed in the literature) to model reasoning under uncertainty.

In addition, the Bayesian framework allows one to reason about probability measures (models) using meta-probabilities. Thus a user may start with a prior distribution of candidate probability measures, and update this as new observations come in. Eventually, after a sufficient number of observations, this model will converge to the objectivistic frequency based model.

Although the Bayesian and classical views have been subject to very animated discussions during a long period, we do not believe that in practice there is opposition between the two approaches, and, in the context of this paper, ingredients of both paradigms will be used.

2.1.4 Random variables, time and random processes

In practice, the behavior of a complex system is generally not completely observable. What is available to observations are measurements, alarms and events which occur during time. For example, in an electric power system it is possible to measure the voltage at a node, or the power flow through a line but normally one doesn't have a full picture of the system behavior.

Such observations are random variables, i.e. real-valued or discrete valued functions defined on the universe of system behaviors[4]. Since the system is dynamic, we assume that most random variables are time tagged : when we talk about a measurement we actually talk about the value assumed by this measurement at a certain time.

A random process is basically a collection of similar random variables taken at different time steps (we will assume for simplicity that time is also discrete). For example, the successive values of voltage at a given node of a power system would define a random process. Depending on the particular scenario, a different realization (a time series) will be observed at the chosen node.

Once a probabilistic model has been defined, it induces probabilistic models for any collection of random variables or processes which are defined on the universe of scenarios.

2.1.5 Reasoning

Probabilistic reasoning aims at exploiting information which may be observed on a system in order to infer information about unobserved variables. Let us enumerate a number of practical examples, in the context of electric power systems.

System identification. Given the values of measurements of voltages and power flows in the system at different time steps, what are the most likely values of parameters of a linear circuit theory model representing the system ?

State estimation. Given the values at time t of a certain number of measurements, and a linear circuit theory model, what is the most likely value of voltages at each node of the system ?

Load forecasting. Given the values of hourly consumptions at a certain node of the system during the last two years, what is the expected value of hourly consumption during the next two weeks ?

Stability prediction. Given the values of rotor angles during the last 300 milliseconds, what is the probability that there will be a loss of synchronism within the next few seconds ?

Planning. Given the values of water inflows and temperatures during the last fifty years, assuming a certain economical growth and system structure, what is the expected cost of operating the power system during the next five years ?

Thus, the standard pattern is "Given some observation, say something about some unobserved (past, present or future) feature of the system". The basic tool to answer these questions is provided by conditioning, which allows to compute the conditional probability distributions of the variables of interest. Indeed, having defined the basic probability model of system scenarios, it is in principle possible to compute the conditional probability model of any random variable (or process) given any type of assumptions on some other random variables (or processes).

Once the conditional distributions are available, it is possible to extract synthetic information from them for decision making, such as expected values or most likely values.

2.1.6 Analytical computations versus Monte-Carlo simulations

In order to go one step further in our discussion, it is necessary to add some structure to our model. This will be done more carefully in section 4.2.2, but for the time being we will start with an intuitive discussion.

Let us suppose that our system scenarios may be described completely using a certain number of parameters together with some equations which allow to compute all random variables and processes once the parameter values are given. Without any deep restriction, we may assume that each parameter belongs to a finite set of possible values, and that the universe of possible scenarios is merely the Cartesian product of these sets. The basic probability measure may then be defined by assigning a positive number to each combination of parameter values, in such a way that the total mass sums up to one.

Clearly, whatever the complexity of the functions which define our random variables, it would in principle be possible to use analytical derivations in order to express their probability distributions as a function of the ground probabilities.

Further, if an observation is made on a certain random variable (say, that the voltage at a certain node and time assumes a given value), the basic probability measure may be replaced by a conditional probability measure. Indeed, the observation (unless the random variable is

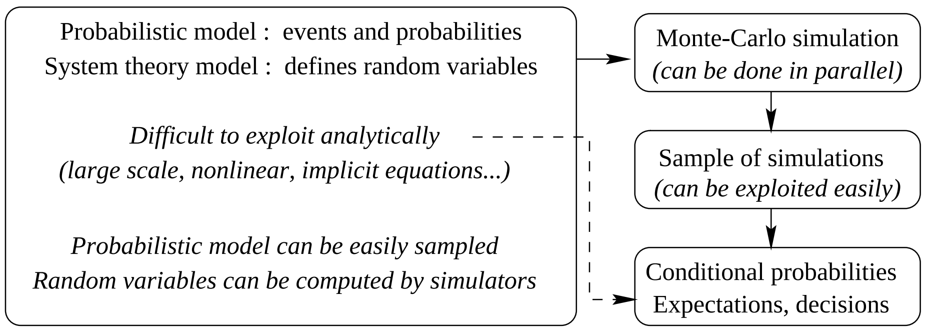

Probabilistic model : events and probabilities
System theory model : defines random variables
Difficult to exploit analytically
(large scale, nonlinear, implicit equations...)
Probabilistic model can be easily sampled
Random variables can be computed by simulators
Monte-Carlo simulation
(can be done in parallel)
Sample of simulations
(can be exploited easily)
Conditional probabilities
Expectations, decisions

Figure 2: Use of Monte-Carlo simulations instead of analytical derivations

constant over the universe) will correspond to a non-trivial event : the subset of the universe where the random variable assumes the observed value. This event can then be used to define the conditional probability measure and hence refresh all the induced probability measures of all other random variables.

Clearly, for a real complex system, the above analytical procedure would quickly become extremely cumbersome, to the extend to jeopardize feasibility. This is further complicated by the fact that in decision making it is generally necessary to find decisions which are optimal in some sense, which in turn requires the computation of conditional probability distributions with respect to a potentially very large number of alternative events corresponding to the possible decisions.

In situations where the analytical computations are intractable, one may use the Monte-Carlo approach [4, 5]. This basically consists in sampling (subsets of) the universe of scenarios according to the basic probability measure and to compute the values of each selected random variable. The technique may be used in order to compute expected values of random variables or values corresponding to some optimality criterion. It may also be used so as to generate samples which can then be compiled into conditional probability distribution models by automatic learning.

Thus, using the Monte-Carlo approach one can pre-compute conditional probabilities of some random variables given values of some others, by using sampling, then use the precompiled models later on for decision making. Figure 2 sketches the idea of using Monte-Carlo simulations instead of analytical derivations to extract conditional probability models. This approach is often economic for the following reasons:

- most of the computations may be carried out in advance, while decision making may be done efficiently in real-time;
- the most heavy part of the Monte-Carlo approach may be easily carried out in parallel;
- the conditional probability distributions may be approximated to the desired degree of accuracy (using automatically learned models);

- only the variables which actually influence each other need to be represented in the conditional probability models derived by automatic learning;
- the models extracted by automatic learning may be formulated in such a way that they can be interpreted physically, which makes it possible to enrich human expertise and also to use human expertise to validate the results.

The overall scheme, combining automatic learning with Monte-Carlo simulations will be further discussed and illustrated in section 4.2.

2.2 Automatic learning and data mining

In this section we will focus on supervised learning, which aims at developing input/output models from observed databases. From a theoretical viewpoint, automatic learning is essentially a generalization of statistical estimation. Below we first formulate this problem as a function approximation problem, then we reformulate it as a problem of building probabilistic models, so as to make explicit its relationship with the previous section. Then we will briefly discuss the contributions of the recently emerged field called data mining. We postpone the discussion of technical and theoretical details to section 3.

2.2.1 Supervised learning as function approximation

In a formal setting, and independently of any particular assumption, the supervised automatic learning problem is generally formulated as follows.

Definition 1 *Given a sample of input/output pairs, say $S = \{(x_1, y_1), \dots, (x_n, y_n)\}$, and a set of candidate input/output models (or hypotheses) $\mathcal{H}$, find an optimal model $H^* \in \mathcal{H}$, such that $\hat{y}_i \stackrel{\triangle}{=} H^*(x_i)$ is as close as possible to y_i for the observed pairs, as well as for any other possible input/output pair which may be observed.*

In this general setting, both x and y may be vectors of real numbers or discrete (symbolic) attributes. If y is symbolic, we will talk about classification problems, if y is a real number (or a vector of real number) we will talk about regression problems.

In the automatic learning literature, $\mathcal{H}$ is called the hypothesis space. It is a set of mappings from the input space towards the output space. Examples of hypothesis spaces are as follows : the set of linear input/output functions defined by a set of matrices (parameters), the set of decision trees of variable complexity, the set of neural networks of given structure, etc.

Thus, supervised learning aims at first choosing an appropriate hypothesis space $\mathcal{H}$ and then selecting a predictive model in $\mathcal{H}$ which may be used in order to guess certain output variables as a *function* of some other input variables.

In order to make our definition more precise, it is necessary to define more precisely what is meant by a good approximation. This will be further elaborated in the next subsection. Then in section 3 we will discuss the main principles which allow one to identify a good hypothesis from a learning sample.

As we will show in section 3, different automatic learning methods essentially differ in the type of hypothesis space they use and in the search method they use in order to find the optimal hypothesis. On the other hand, the criteria used to measure hypothesis quality are essentially independent of the type of automatic learning method.

2.2.2 Learning as identification of probabilistic models

Let us make the link with the probabilistic models introduced in section 2.1.

We start by assuming that each observation corresponds to some object in the universe of possible objects. Hence, input and output variables are actually random variables defined on the universe, and function approximation aims at building an approximation of some random variables (y) as a functional combination of some other random variables (x).

However, according to our discussion in section 2.1, all the information provided about the output variables by the input variables may be encoded appropriately in a conditional probability model

$$P(y|x). \tag{2}$$

Now, let us choose a deviance criterion $d(y, y')$ in order to measure the difference between two output values. The precise form of the deviance criterion is not relevant to our present discussion, but it should have properties similar to classical distance measures. Using any such measure, any hypothesis may be compared to the target random variable by computing its *expected risk*, defined as follows

$$R(H) \overset{\triangle}{=} E\{d(y, H(x))\} = \int_{x,y} d(y, H(x))dP(x, y), \tag{3}$$

where the expectation is taken along the joint probability distribution of input-output pairs. The lower R, the more accurate the hypothesis according to the chosen deviance criterion.

Thus, it is straightforward to derive from the conditional probability distribution of eqn. (2) an optimal input/output model by the following equation [6]

$$y_B(x) = \arg_y \min \int_{y'} d(y, y')dP(y'|x). \tag{4}$$

This optimal model is called in the automatic learning literature the *Bayes* model. For a given choice of input and output variables, it depends of course on the deviance criterion used. The expected risk of the Bayes model, defined by

$$R_B \overset{\triangle}{=} E\{d(y, y_B(x))\}, \tag{5}$$

is by definition the ultimate lower bound of the risk (it is usually called residual risk, or residual error in the literature on automatic learning).

For example, in classification problems (discrete output), the conditional probability distribution will be discrete. The usual deviance criterion is misclassification probability in this case. Thus, the conditional probability of y given x may be used in order to identify the most probable value of y given x. This would correspond to using eqn. (4) with the discrete deviance measure, i.e. $d(y, y') = 1 - \delta(y, y')$, which is equal to zero if $y = y'$ and equal to one otherwise. The resulting Bayes rule has a minimum error probability.

If the output variable is numerical, the usual deviance criterion is the euclidian norm, leading to minimum square error learners, which essentially aim at approximating y by its conditional expectation $E\{y|x\}$. Various hypotheses (e.g. Gaussian noise) may be used in order to justify such criteria, but this discussion would lead us too far from our topic.

If the input-output relationship is deterministic (i.e. if there exists a function $f(\cdot)$ such that $y = f(x)$) then, with these deviance measures, it is also true that

$$y_B(x) = f(x) \text{ and } R_B = 0. \tag{6}$$

However, in general, the Bayes model, although it does provide an optimal guess for the outputs in terms of the inputs, does not provide full information about the probabilistic relationship among them. Thus, the point we would like to stress here is that rather than viewing automatic learning as function approximation as in section 2.2.1, it is advantageous to see it as probabilistic modeling. This leads to the following rephrasing of the automatic learning problem.

Definition 2 *Given a database of input output/pairs, say $\{(x_1, y_1), \ldots, (x_n, y_n)\}$, and a set of candidate input/output probabilistic models $\mathcal{P}$, find an optimal model $P^* \in \mathcal{P}$, such that $P^*(y|x)$ is as close as possible in the average to the true conditional probability $P(y|x)$.*

Note that this definition is slightly more general than the first definition : the objective now is not only to provide a good way to guess what would be the output given the inputs, but also to describe the probabilistic nature of the input-output behavior of the system. However, as is stated by eqn. (4), once a good probabilistic model of the system is available, it is straightforward to derive a good approximation function from it.

In section 3.2 we will review the main principles which lead to the formulation of automatic learning algorithms, i.e. algorithms which try to solve the learning problems stated by these two definitions with the sole information provided by a sample (i.e. without knowing the conditional probability distributions explicitely).

2.2.3 Data mining

Anticipating on section 3.2, let us say that the two main lines of theoretical progress in automatic learning in the last twenty years are, on the one hand, generalization of statistical estimation to flexible families of models (decision trees, neural networks, projection pursuit regression), and a better theoretical understanding of non-parametric (variable complexity) automatic learning methods, in the small to medium sample size case, on the other hand.

From the practical point of view, automatic learning algorithms have been improved significantly, which makes it possible to handle larger and larger databases. In the same time database and data collection technologies have progressed even more quickly, to such an extent that the traditional tools (structured query languages) offered by database management systems are not sufficiently powerful anymore to make the best use of the information contained in the existing databases.

The need for smarter tools to extract synthetic information from databases has led to the development of so called data mining platforms. These are software environments which generally combine traditional database management systems with an automatic learning toolbox, sophisticated user interfaces, and visualization techniques [7].

The same need has also led to a new research area called knowledge discovery in databases (KDD), which aims at developing methods to extract and validate useful knowledge from very large databases. In particular, the added value of KDD with respect to raw automatic learning is to help users to formulate more flexible criteria (automatic learning research generally focuses on accuracy only) to extract useful information from databases.

On the other hand, under research is also the development of parallel automatic learning algorithms, in order to be able to treat very large data sets with acceptable response times [8].

3 Automatic learning methods

The present section focuses on the description of a subset of complementary automatic learning methods. Throughout this section we will use simple hypothetical examples related to an academic example database generated for power system stability assessment and control, which we first introduce. Next we discuss the main theoretical considerations which lead to the principles of many modern automatic learning methods. Then we review the three main families of *supervised* learning methods, and their combination into hybrid techniques. Finally, we end with a brief discussion of *unsupervised* learning methods and principles.

Table 1: Spreadsheet view of a small part of the OMIB database

	Pu	Qu	Pl	Vl	Xinf	Vinf	CCT	STATUS
SC5001	876.029	-193.660	-98.179	1.067	54.598	1.076	0.261	SECURE
SC5002	1110.880	-423.190	-119.300	1.119	58.228	1.112	0.162	SECURE
SC5003	980.132	79.722	-122.600	1.063	62.537	1.062	0.213	SECURE
SC5004	974.139	217.073	-100.520	1.015	64.428	1.010	0.190	SECURE
SC5005	927.198	-618.470	-100.000	1.020	42.557	1.017	0.174	SECURE
SC5006	1192.590	617.266	-103.460	1.073	51.230	1.065	0.199	SECURE
SC5007	1069.120	7.137	-109.460	1.016	66.446	1.010	0.129	INSECURE
SC5008	1189.200	905.121	-87.433	1.056	66.052	1.071	0.167	SECURE
SC5009	999.084	685.442	-107.870	1.109	59.726	1.110	0.272	SECURE
SC5010	1241.880	-442.250	-105.680	1.078	58.917	1.079	0.091	INSECURE
SC5011	845.574	816.962	-106.180	1.103	57.922	1.098	0.352	SECURE
SC5012	1151.250	10.003	-86.426	1.108	64.069	1.113	0.162	SECURE
SC5013	963.600	-312.430	-96.890	0.988	61.517	0.986	0.131	INSECURE
SC5014	721.150	155.468	-95.262	1.050	63.566	1.027	0.347	SECURE
SC5015	1135.190	320.912	-117.480	1.049	56.051	1.041	0.176	SECURE
SC5016	939.189	234.557	-109.450	1.100	49.913	1.110	0.293	SECURE
SC5017	923.754	294.472	-100.300	1.106	74.579	1.120	0.252	SECURE
SC5018	886.942	446.574	-87.805	0.950	57.044	0.950	0.230	SECURE

3.1 Academic example database

We will use an example of power system security assessment to illustrate automatic learning methods. We will discuss this problem and its subproblems in section 4. Let us merely say that the security of a power system denotes its capacity to react in a satisfactory way to unforeseen events (short-circuits, outages, mis-operations, etc.). In preventive mode, power system security assessment and control aims at taking decisions in order to obtain a satisfactory level of security while reducing operating costs as much as possible. For a large power system, this activity is quite complex. Below, we will use automatic learning in order to derive as simple as possible rules for preventive security assessment and control. For illustration purposes, we will use a very simple academic example, which is not at all representative of the complexity of real large scale systems, but simple enough to make explanations easy.

Table 1 shows a spreadsheet view of a small part of a database related to power system transient stability assessment. The lines of the table correspond to the different objects of the database, which are representing different power system scenarios. Each column corresponds to a variable characterizing the scenarios : the first six columns will be used in our illustrations as input variables, and the last two as output variables.

Physically, the database corresponds to an academic *one-machine infinite-bus (OMIB)* power system, composed of a generator, a load, and an equivalent transmission line connected to an “infinite bus”. These latter two elements are a simplified representation of the remaining power system to which the generator is connected (see figure 3). The first two variables measure the active (Pu) and reactive (Qu) power generated, while the third and forth variable measure the active power consumed locally (Pl) and the voltage magnitude at the load bus (Vl). The next two variables represent the equivalent system strength in terms of voltage (Vinf) and transmis-

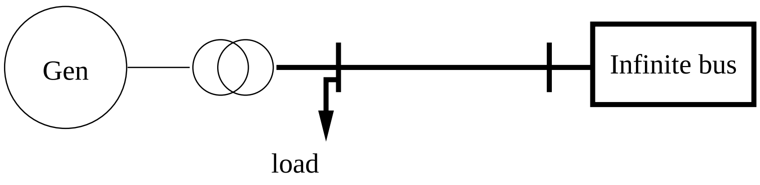


Figure 3: Simple one-machine infinite-bus system (OMIB)

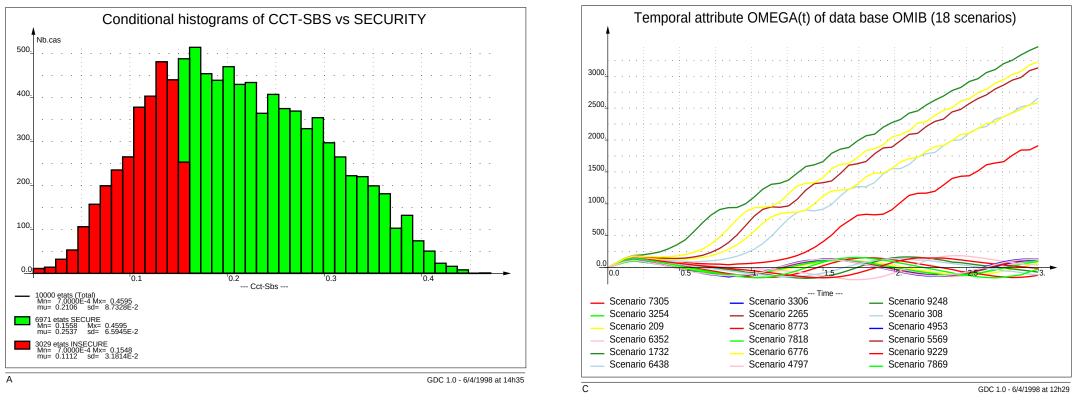


Figure 4: Illustration of OMIB database content

sion system (Xinf). The last two variables shown in table 1 are measuring the degree of stability of the system with respect to a given (fixed) fault at the generator bus: CCT denotes the critical clearing time, which is the maximum duration of the fault without loss of synchronism; “Status” denotes whether the system is sufficiently secure or not.

The complete database is composed of 10000 random scenarios simulated for this small system. It is composed of two parts corresponding to two different assumptions : scenarios 1 to 5000 correspond to a specific hypothesis where variables Pl, Vl, Xinf and Vinf are constant; scenarios 5001 to 10000 relax this hypothesis (see Table 1). Thus the first part of the database corresponds to a simpler problem than the second part of the database : the input space actually reduces to two dimensions, since only the first two parameters are variable.

Figure 4 shows two graphics illustrating the content of the whole database : left hand part contains the histogram of the distribution of values of CCTs, and its right hand part shows some temporal attributes (curves) related to the mechanical rotor speed of the generator when submitted to a short-circuit (one can easily figure out the stable scenarios among those which are shown as those which rotor speed remains bounded).

We will start by using parts of this simpler database in order to illustrate automatic learning methods. Figure 5 shows two and three dimensional scatterplots of these first 5000 scenarios of the database. The left hand part of the figure shows the (simple) relationship between inputs (Pu and Qu) and symbolic output (security status); we will use this output variable to illustrate classification methods. The left hand part, shows the relationship between the inputs and the continuous output CCT; we will use this output variable to illustrate regression techniques.

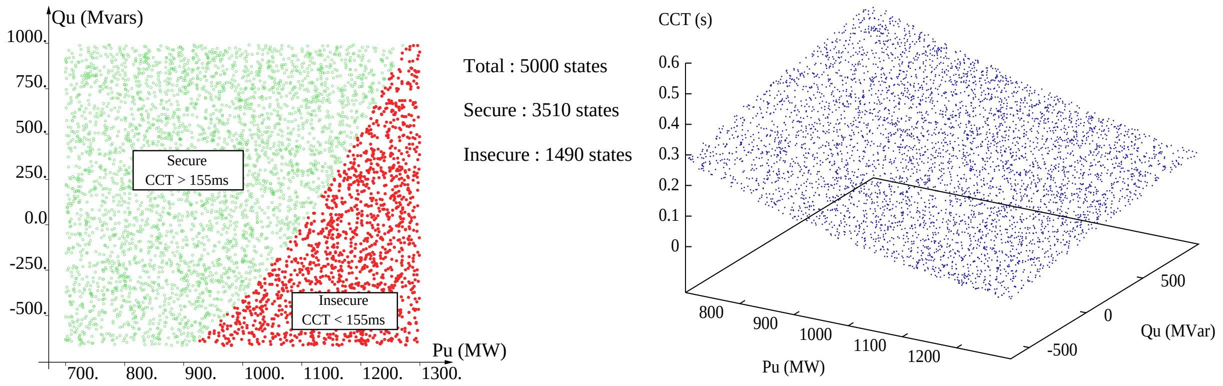


Figure 5: First 5000 random scenarios of the OMIB database (adapted from [21])

In order to be able to evaluate the capability of the models derived by automatic learning to predict correctly unseen states, we split our sub-database in two parts : (i) as a learning sample ($\boldsymbol{LS}$) we use scenarios 1 to 3000 in order to build models; (ii) as a test sample ($\boldsymbol{TS}$) we use the scenarios 3001 to 5000 to evaluate accuracy of models on unseen states.

3.2 Theoretical considerations

In this section we will briefly discuss the main principles behind automatic learning. We recommend to read references [9, 10, 11] for a more comprehensive discussion of what is briefly sketched below.

3.2.1 Learning principles

Whether automatic learning is formulated as function approximation or as conditional probability distribution approximation, the main difficulty is to exploit correctly the information provided in a learning sample so as to choose a correct approximation in the chosen hypothesis space.

In this context, it is useful to distinguish among the situation where the hypothesis space is small compared to the size of the sample, and the opposite situation where the hypothesis space is very large. Actually, the former case is the usual situation considered in standard parametric asymptotic statistics. The latter case is the one generally considered in the more recent theoretical work on automatic learning, and has led to new learning principles, for example for neural networks and decision trees.

In the large sample (and small hypothesis space) case, the prevailing principle in automatic learning consists of minimizing the empirical risk. This amounts to choosing among all candidate models the one which yields the smallest deviance in the learning set. It is rather straightforward to show that, under the assumption of independent learning samples, this model will converge towards the best choice (within the given hypothesis space) when the sample becomes infinitely large. (See, for example, [11] for a precise discussion of the meaning of “small” hypothesis space.)

In the small sample case (or when the hypothesis space is very large), it is generally advisable to tradeoff the empirical risk with a measure of the complexity of the chosen hypothesis. For example, in the case of decision tree induction this leads to so called pruning methods, as will be discussed later on. The necessity of this compromize is related to the so-called “overfitting” problem : a small empirical risk measured on the learning set does not necessarily imply a small actual risk, measured outside the training set. This will be further illustrated below for various types of automatic learning methods.

Risk complexity tradeoff. The two above extreme cases may be combined using a general quality measure for automatic learning, which is of the following form

$$Q(H,S) = R_e(H,S) + \beta C(H), \tag{7}$$

where $R_e(H,S)$ denotes the empirical risk as measured on the learning sample S, $C(H)$ is a measure of the complexity of the hypothesis H, and β is a non-negative parameter depending on the sought compromise. Typically, β would decrease when the learning sample size increases. The objective of automatic learning is then to select H^* in $\mathcal{H}$ which minimizes Q, given the learning sample S.

The precise definition of the empirical risk will depend on the exact problem formulation. For example, if the problem is seen as a function approximation problem, with a mean square error criterion, then we would use the following definition of the empirical risk

$$R_e(H,S) = \frac{1}{n}\sum_{i=1}^{n} ||H(x_i) - y_i||^2. \tag{8}$$

On the other hand, if the purpose is to develop a conditional probability model, the usual criterion would be sample log-likelihood, i.e.

$$R_e(P,S) = \frac{1}{n}\sum_{i=1}^{n} \log P(y_i|x_i). \tag{9}$$

Bayesian principle. In the Bayesian framework, the hypothesis space is first “decorated” with a prior probability distribution, and the Bayesian learning principle consists of computing, from these priors and from the learning sample, the posterior probability of any hypothesis, which serves as a criterion to evaluate candidate hypotheses. This turns out in the following formulation

$$Q_B(H,S) = \log P(H|S) = \log P(S|H) + \log P(H) - \log P(S), \tag{10}$$

where the last term is usually dropped since is does not depend on the chosen hypothesis H. The two first terms of this criterion are of similar nature than the two terms of equation (7): the first term measures how well the sample is explained by the hypothesis, and the second term depends only on the prior properties of the hypothesis (complexity in eqn. (7), probability in eqn. (10)). Also, a closer look at criterion (10) shows that the smaller the learning sample, the bigger the weight of this latter term, and thus the less complex (or the more a priori probable) the optimal hypothesis.

Minimum description length principle (MDL). The MDL principle stems from information theory, and more specifically, from data compression coding theory [12]. It states that the best hypothesis is the one which allows to represent the output information of the learning states in the most compact form, taking into account the coding of approximation errors in the learning set and the cost of coding the hypothesis itself. Thus the description length of the data and a model is expressed as follows

$$DL(H, S) = DL(S|H) + DL(H), \tag{11}$$

where the first term is the length of coding the errors of the model in the learning set (it is equal to zero when the empirical risk is equal to zero) and the second term is the number of bits required to code the model itself (it is smaller for less complex models). The MDL principle has been applied to many different automatic learning methods, and in particular in the context of decision tree induction [13, 14, 15] and pruning [16, 17].

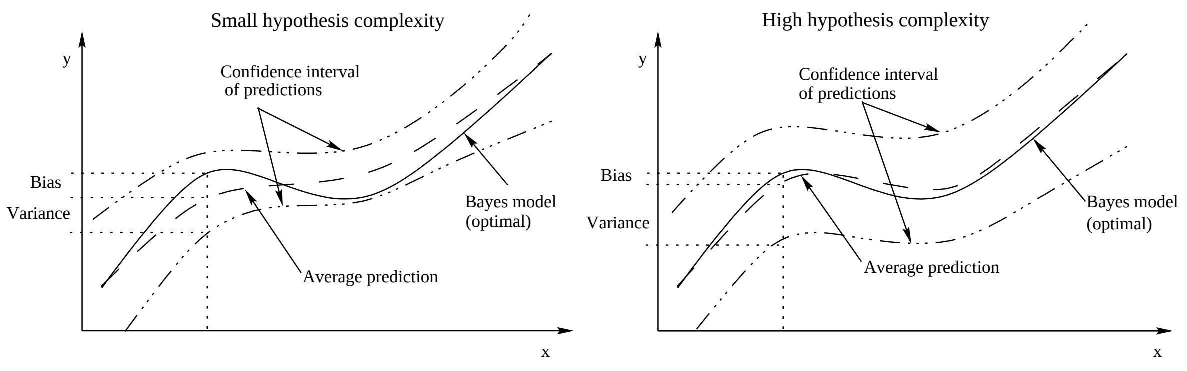
It is clear that all three criteria essentially try to reach the same type of compromise : choosing more complex models only if the explanation they provide for the learning states is significantly better.

3.2.2 Bias and variance

The recent work in automatic learning led to the rediscovery of a well known phenomenon in statistics, namely the “bias-variance tradeoff”.

The bias-variance tradeoff, which actually provides an interpretation for the learning principles described in section 3.2.1, may be simply stated as follows :

- any automatic learning algorithm computes a hypothesis which is a function of the learning sample;
- the learning sample is random in nature, hence also the result of the automatic learning algorithm;
- the prediction at a certain point of the input space by an hypothesis found by automatic learning is thus a random variable;



Small hypothesis complexity
y
Confidence interval of predictions
Bias
Variance
Bayes model (optimal)
Average prediction
x
High hypothesis complexity
y
Confidence interval of predictions
Bias
Variance
Bayes model (optimal)
Average prediction
x

Figure 6: Illustration of bias and variance

- the bias (at a certain location in the input space) is defined as the difference between the desired value at this point (say the value predicted by the Bayes model), and the expected prediction of the hypotheses produced by automatic learning for different samples of a given size;
- the variance (at a certain location in the input space) is the square difference between the average predictions of hypotheses and individual ones;
- both bias and variance contribute to sub-optimality, in the sense that both result in an hypothesis being different in the average from the Bayes model;
- for fixed hypothesis complexity, variance generally decreases when the sample size increases, whereas bias remains constant;
- for fixed sample size, variance generally increases when the hypothesis complexity increases, whereas bias will generally decrease.

This is further depicted in figure 6 using a simple example where both input and output spaces are one-dimensional. The two graphs are drawn for identical conditions (same output information, same learning sample size) except that in the left hand graph, the complexity of the hypothesis is smaller than in the right hand graph.

The first observation which can be made from figure 6 is that bias and variance vary from one input location to another. Thus, typically, variance is higher in regions where the probability to observe input variables is lower (e.g. close to the boundaries of the input space). On the other hand, bias is higher in the regions where the Bayes rule has a higher curvature.

The comparison of the left and right graphs of figure 6 illustrates how variance and bias vary in opposite direction when the complexity of the hypothesis is increased.

In the recent years, the improved understanding of the bias-variance tradeoff led to the proposal of new algorithms aiming at fighting against bias with reduced variance models and fighting against variance with reduced bias models (see for example the literature on bagging and boosting [18, 19, 20]).

Bias-variance tradeoff by cross-validation. Whatever the approach leading to the chosen quality measure, in many practical situations it is difficult to define precisely the tradeoff between empirical risk and complexity. In other words, the form of the measure is known up to the value of the parameter β, which is typically problem dependent : intuitively, if the problem is more complex, the value of β should be larger, and vice-versa.

Thus the appropriate value of β must be identified from the available samples, using cross-validation. In practice this leads to the following procedure :

- divide the available sample in two parts;
- using the first part of the sample and a sequence of candidate values of β, build a corresponding sequence of hypotheses;
- using the second part of the sample, evaluate the generalization capability of each hypothesis and select the one which obtains the best performance.

This procedure will be illustrated below. Let us notice that it may be further refined in order to make the best use of the available samples. Clearly, it results in a certain increase in computational burden since instead of building a single hypothesis it requires the building of several ones.

3.2.3 Learning algorithms

Once the hypothesis space and quality measure have been defined, automatic learning merely amounts to an optimization (or search) problem. Depending on the structure of the hypothesis space, different optimization techniques may be used : direct solutions by linear equation solvers (linear models with quadratic quality measures), nonlinear optimization, heuristic search, enumerative search, genetic algorithms, or any combination of these techniques.

It is also interesting to distinguish between incremental and batch learning strategies. Incremental learning proceeds by using the individual learning samples in a sequential manner in order to progressively adapt the hypothesis. Batch learning methods proceed by using the whole sample as was suggested in the preceding description. The advantage of incremental learning is to avoid storage of learning samples, which may become cumbersome in some applications. Its main disadvantage with respect to batch learning is sub-optimality in terms of speed and accuracy.

Part of the research work carried out in the recent years led to the development of efficient optimization procedures, as we will outline below.

3.3 Main classes of supervised learning methods

Let us recall the objective of supervised learning :

> *Given a set of examples (the learning set (LS)) of associated input/output pairs, derive a general rule representing the underlying input/output relationship, which may be used to explain the observed pairs and/or predict output values for any new unseen input.*

In automatic learning we use the term *attribute* to denote the parameters (or variables) used to describe the input information.

In the context of electric power system security assessment, an *example* would thus correspond to a state of a particular power system, or more generally to a simulated dynamic scenario. The input attributes would be relevant parameters describing its electrical state and topology, which can be either directly measured in real-time or can be computed in some way from real-time measurements. Outputs could be information concerning its security, for example in the form of a discrete classification (e.g. secure / insecure), or a numerical security margin such as the CCT in our OMIB example.

Note. For the sake of simplicity, we will only describe the principles of the various automatic learning methods. For precisions concerning technical details of the methods illustrated below, we kindly refer the reader to [21] where all the information required to reproduce the results shown here is provided.

3.3.1 Machine learning

Machine learning (ML) is a subfield of automatic learning concerned with the automatic design of rules similar to those used by human experts (e.g. if-then rules). We will describe only *Top down induction of decision trees* (TDIDT) and some of its variants, which form one of the most successful classes of such methods [22, 23].

Decision trees. Before describing how TDIDT proceeds to build decision trees let us explain what a decision tree is and how it is used to classify a state. Figure 7 shows a hypothetical binary decision tree (DT) for our problem using the two attributes Pu and Qu. The bold arrows on the tree suggest how a hypothetical state (Pu = 1000 MW and Qu=-500 MVAr) traverses the tree in a top down fashion to reach a terminal node. One starts at the topnode and applies sequentially the dichotomous tests encountered to select the appropriate successor. When a terminal node is reached, the output information stored there is retrieved. Thus, for our hypothetical state the conclusion is “insecure”. Note that the tree may be translated into an equivalent set of if-then rules, one for each terminal node. E.g. the tree in figure 7 translates into the rules indicated beneath it.

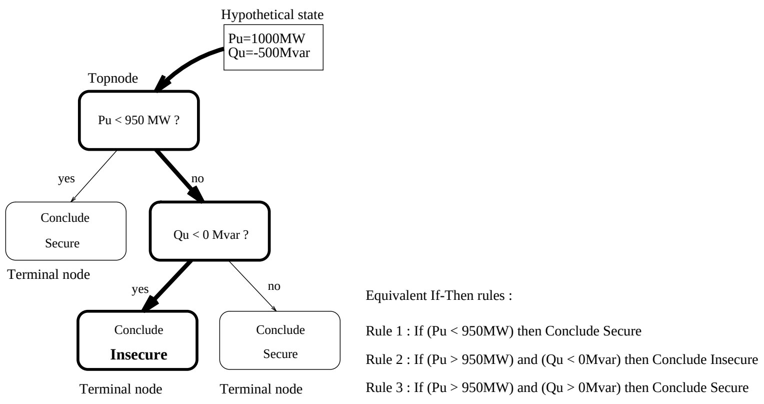

Hypothetical state
Pu=1000MW
Qu=-500Mvar
Topnode
Pu < 950 MW ?
yes
no
Conclude Secure
Qu < 0 Mvar ?
Terminal node
yes
no
Conclude Insecure
Conclude Secure
Terminal node
Terminal node
Equivalent If-Then rules :
Rule 1 : If (Pu < 950MW) then Conclude Secure
Rule 2 : If (Pu > 950MW) and (Qu < 0Mvar) then Conclude Insecure
Rule 3 : If (Pu > 950MW) and (Qu > 0Mvar) then Conclude Secure

Figure 7: Hypothetical decision tree and equivalent if-then rules (taken from [21])

Decision tree growing. Now, let us illustrate on our example how the TDIDT method will extract from our learning set a number of classification rules in the form of a decision tree.

Figure 8 illustrates the successive node splitting procedure. The procedure is initialized by creating the topnode of the tree, which corresponds to the full $\boldsymbol{LS}$ as shown in figure 8a. Note that the relative size of the dark and light areas of the box used to represent the topnode corresponds to the proportion of insecure and secure states in the full learning set (909 insecure states vs 2091 secure states).

The method starts with a list of attributes (also called *candidate attributes*) in terms of which it will formulate the tree tests. In our example we use only two candidate attributes (Pu and Qu) since all other input parameters are constant in the first part of the database.

To develop the topnode, each candidate attribute (here Pu and Qu) is considered in turn, in order to determine an appropriate threshold. To this end, the learning set is sorted by increasing order of the considered attribute values, then for each successive attribute value, a dichotomic test is formulated and the method determines how well this test separates secure and insecure states, using an information theoretic score measure. The score measure is normalized, between 0 (no separation at all) and 1 (perfect separation). Figure 8b shows how the score varies in terms of the threshold both for Pu and Qu at the topnode. Thus, the optimal threshold for Pu is found to be 1096.2 MW (with a score of 0.36) and the optimal threshold for Qu is found to be -125MVAr (with a score of 0.09). Finally, the overall best test is identified at the topnode to be Pu$>$1096.2 MW.

Once the optimal test is found, the next step consists of creating two successor nodes corresponding to the two possible issues of the test; the learning set is then partitioned into corresponding subsets by applying the test to its states. The result of this step is represented at figure 8c. Note that the number on the top of each node represents the number of corresponding learning states : 3000 at the topnode, 1026 at the first successor and 1974 at the second successor. Note also that the first successor contains a strong majority of insecure states, while the second successor contains a very strong majority of secure states.

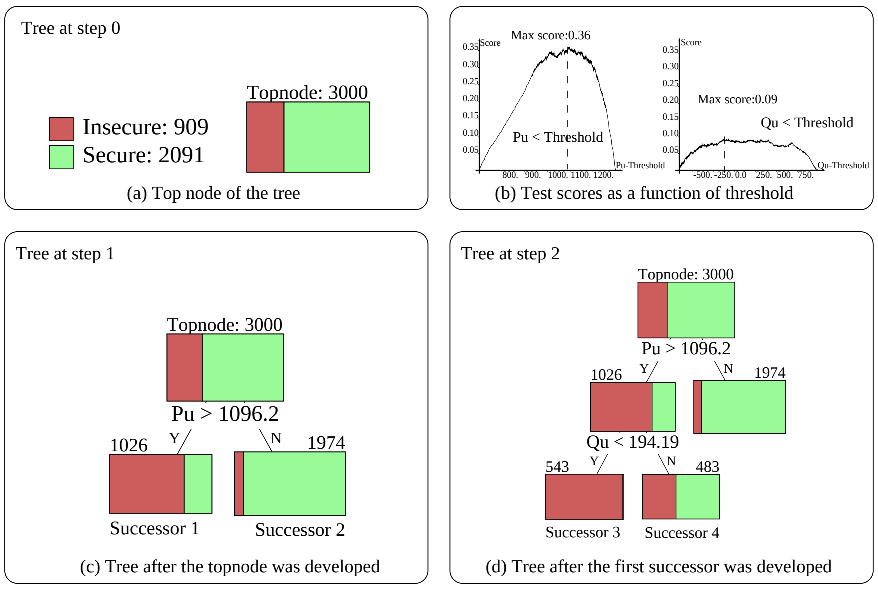


Figure 8: Three first steps of decision tree growing (taken from [21])

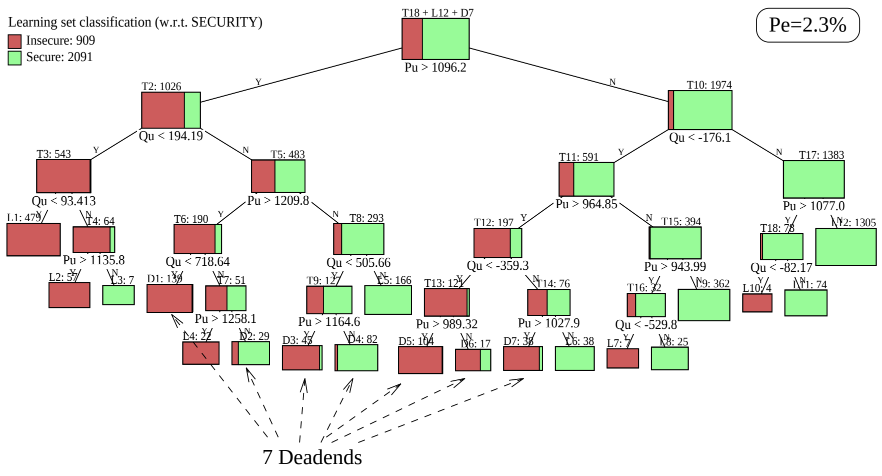


Figure 9: “Orthogonal” decision tree (end result) (taken from [21])

Stopping to split criterion. As is illustrated on figure 8d, the procedure continues recursively to split the recently created successors, gradually separating the secure and insecure states until a stop splitting criterion is met. The stop splitting criterion decides whether a node should indeed be further developed or not. There are two possible reasons to stop splitting a node, which yield two types of terminal nodes : *leaves* and *deadends*. A leaf is a node which corresponds to a sufficiently pure subset (e.g. all states belong to the same class). A deadend is a node where there is not enough statistical support for choosing an appropriate test. Stop splitting at deadend nodes prevents the tree from over-fitting the learning set and hence allows the method to reach a good compromise between accuracy and simplicity.

The end result of this procedure is the tree shown at figure 9, which partitions the learning set into subregions defined by line segments orthogonal to the Pu or Qu axes; this “orthogonal” tree is composed of 18 test nodes, 12 leaves and 7 deadends.

Validation. Since the tree is grown to reach a good compromise between simplicity and separation of secure and insecure *learning* states it provides a kind of summary of the relationship observed in the learning set between Pu and Qu attributes and security class. But, how well does it generalize to unseen states ? To answer this question, we use the test set of 2000 states different from the learning states and compare the security class predicted by the tree with the one derived from the CCT computed by numerical simulation.

Thus each test state is directed towards a terminal node on the basis of its input attribute values (Pu and Qu) and applying sequentially the dichotomous tests encountered to select the appropriate successor. When a terminal node is reached, the output majority class of the corresponding learning subset stored there is retrieved and the test state is classified into this class. E.g. states reaching terminal nodes L1, L2, D1, L4, D3, D5, D6, D7, L7 and L10 are predicted to be insecure, while those reaching terminal nodes L3, D2, D4, L5, L6, L8, L9, L11 and L12 are predicted to be secure. Among the 2000 test states, this yields 1954 correct classifications, 15 insecure states declared erroneously secure, and 31 false alarms, i.e. an error rate Pe of 2.3%.

Refinements. There are many refinements of the TDIDT method of interest in the context of security assessment. First of all, decision trees may exploit easily discrete attributes (e.g. to represent power system topology, or events) together with numerical ones. They may also be generalized to an arbitrary number of (security) classes and to tests with more than two outcomes.

Another interesting extension consists of using linear combinations instead of single attribute (orthogonal) splits, yielding so-called "oblique" decision trees. They are useful when there are strong interactions among different candidate attributes. For example, in our illustrative problem we could use linear combinations among Pu and Qu, which should provide a more efficient separation between secure and insecure states.

Figure 10 shows a tree obtained in this fashion. During tree building, we search for splits in the form of "Pu + Weight*Qu$<$Threshold" instead of searching for single attribute splits (in the form of "Pu$<$Threshold" and "Qu$<$Threshold"). The optimal splitting procedure is modified in order to determine automatically both an appropriate weight and the optimal threshold at each test node.

The fact that the resulting "oblique" tree is significantly simpler than the "orthogonal" one of figure 9 (only 6 test nodes, 6 leaves and 1 deadend) confirms our intuition. The tree is also much more reliable (no non-detections and only two false alarms among the 2000 test states, i.e. an error rate of 0.1%).

Figure 10 further illustrates the difference between the two classification boundaries induced

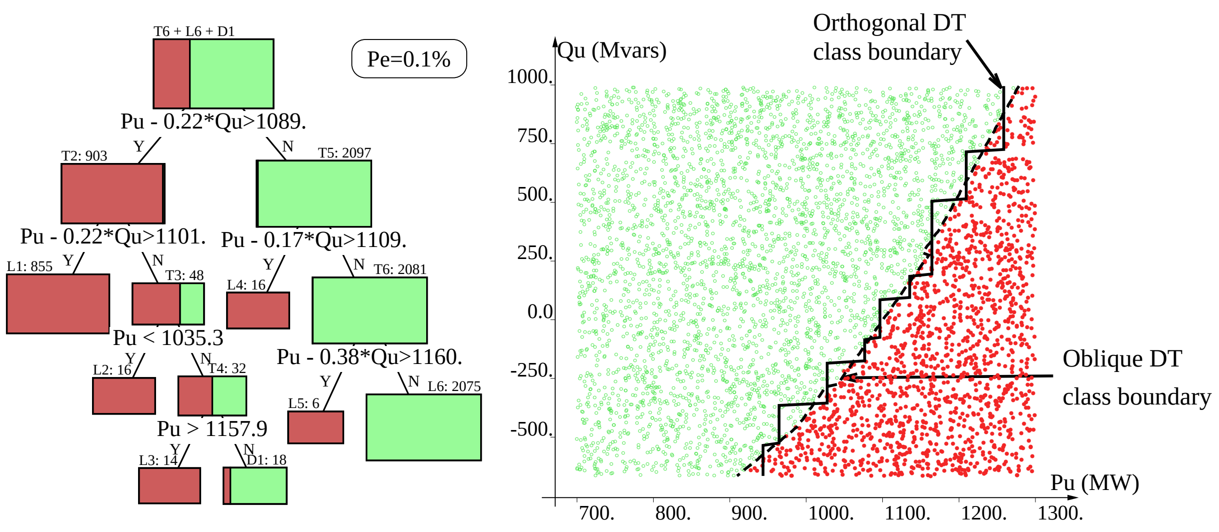


Figure 10: “Oblique” decision tree and classification boundaries (adapted from [21])

by the two trees : a rather rough staircase approximation for the “orthogonal” tree vs a much smoother boundary for the “oblique” one.

The only price to pay for this improvement is an increase in CPU time at the tree growing stage, since searching for linear combinations is more intricate than searching for optimal thresholds. E.g. in our example it took 30 seconds[5] to grow the “oblique” tree and only 3 seconds to grow the “orthogonal” one.

In addition to “oblique” trees, other interesting extensions are *regression* trees which infer information about a numerical output variable, and *fuzzy* trees which use fuzzy logic instead of crisp logic to represent output information in a smooth fashion [24]. Both approaches allow us to infer information about security margins, similarly to the techniques discussed below in §§3.3.2 and 3.3.3.

For example, figure 11 shows a (partial view of a) regression tree which approximates the CCT of the OMIB system as a piecewise constant function of the machine rotor angle in the during fault period. Note that the shaded area and the number in each box provide information about the mean and standard deviations of CCTs of the scenarios corresponding to each node. The top-node corresponds to all possible scenarios, whereas the terminal nodes correspond to a subset of scenarios falling within a certain range of $\delta(t)$ values. For example, the lower left node corresponds to scenarios such that $\delta(150) > 54.2^\circ$ and $\delta(100) > 67.4^\circ$; for this kind of scenarios the mean CCT value is 68ms and its standard deviation is equal to 19ms. Similarly, the lower right node corresponds to scenarios such that $\delta(150) \leq 39.5^\circ$; they have a mean CCT of 383ms and a standard deviation of 21ms. Thus, the regression tree allows one to approximate the CCT from rotor angles in the during fault period; in our example this approximation is actually quite accurate, the mean absolute error being about 2ms.

Finally, a recent research field in the context of decision tree induction consists of temporal

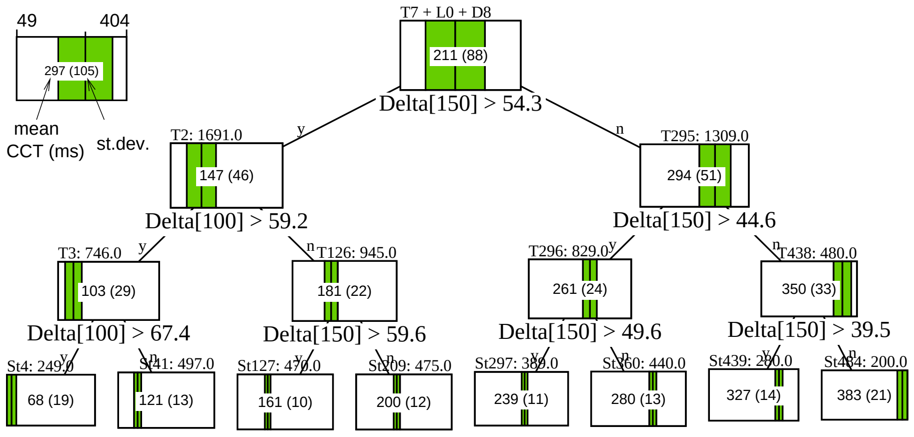


Figure 11: Regression tree (CCT function of $\delta(t)$)

trees [25]. These are decision trees which may exploit directly temporal attributes, in order to classify a situation as early as possible in a certain class.

Salient features of decision trees. The main strength of decision trees is their interpretability. By merely looking at the test nodes of a tree one can easily sort out the most salient attributes (i.e. those which most strongly influence the output) and find out how they influence the output. Furthermore, at the tree growing stage the method provides a great deal of additional information, e.g. about scores of different candidate attributes, their correlations, and the overall information they provide to the tree.

Another very important asset is the ability of the method to identify the most relevant attributes for each problem. Our toy problem was too simple to illustrate this feature, but in large-scale applications less than twenty percent of the candidate attributes are typically selected while growing a tree.

The last characteristic of decision trees is computational efficiency : tree growing computational complexity is practically linear in the number of candidate attributes and in the number of learning states, allowing one to tackle easily problems with a few hundred candidate attributes and a few thousand learning states. The use of a tree as a classification algorithm is ultrafast, compatible with any real-time constraints.

Computational efficiency together with interpretability enable the method to be used in an interactive trial and error fashion, so as to discover interesting information contained in a database and thereby gain physical insight into a problem. Below, we describe methods which are essentially complementary to decision trees and may be combined with them in hybrid approaches.

3.3.2 Linear and nonlinear regression

This type of method aims at building essentially smooth input/output models. Most of the smooth regression techniques may be viewed as a parametric model-fitting approach. In these approaches the hypothesis space is defined as a set of functions

$$\mathcal{H} = \{f(\cdot, \boldsymbol{w}_m) | \boldsymbol{w}_m \in W\}, \tag{12}$$

where $\boldsymbol{w}_m$ is an $m-$dimensional vector of parameters (or weights) and W is a subset of $\mathbb{R}^m$. The objective of learning thus reduces to the search of an optimal value $\boldsymbol{w}_m^*$ of the parameter vector which would minimize the empirical risk (typically the mean square error)

$$R_e(\boldsymbol{w}_m) \triangleq \frac{1}{n} \sum_{i=1}^{n} d(f(\boldsymbol{x}_i, \boldsymbol{w}_m), y_i). \tag{13}$$

If the function $f(\cdot, \cdot)$ depends in a smooth way on the parameter values and if the deviance measure is smooth (these two requirements generally hold true), then the empirical risk (13) is also a smooth function of the parameter values. Therefore, these techniques generally use gradient descent type of search techniques so as to minimize R_e with respect to the parameters $\boldsymbol{w}_m$.

One of the main difficulties with these methods is to determine the appropriate family of functions suitable for a given problem. This problem is generally solved by defining a nested family of hypothesis spaces

$$\mathcal{H}_0 \subset \mathcal{H}_1 \subset \mathcal{H}_2 \subset \dots \tag{14}$$

such that the complete superset of these hypothesis spaces has some kind of “universal approximation capability” in a large enough hypothesis space $\mathcal{H}$.

This means that for any ϵ and any function $g \in \mathcal{H}$, there exists at least one i and one function $f \in \mathcal{H}_i$ which is ϵ-close to g, i.e. such that

$$R(f, g) = E_x\{d(g(x), f(x))\} \leq \epsilon \tag{15}$$

where the expectations are taken with respect to the probability measure defined on the input space. Below we will provide examples of such hypothesis spaces in the context of multilayer perceptrons and projection pursuit.

Once such a family of hypothesis spaces have been defined, the supervised learning problem then amounts to the two following subproblems :

Choice of structure : choice of one of the hypothesis spaces in the family, say $\mathcal{H}_{i^*}$

Parameter optimization : for a given choice of $\mathcal{H}_i$, choice of the correct parameter vector $\boldsymbol{w}_m^*$.

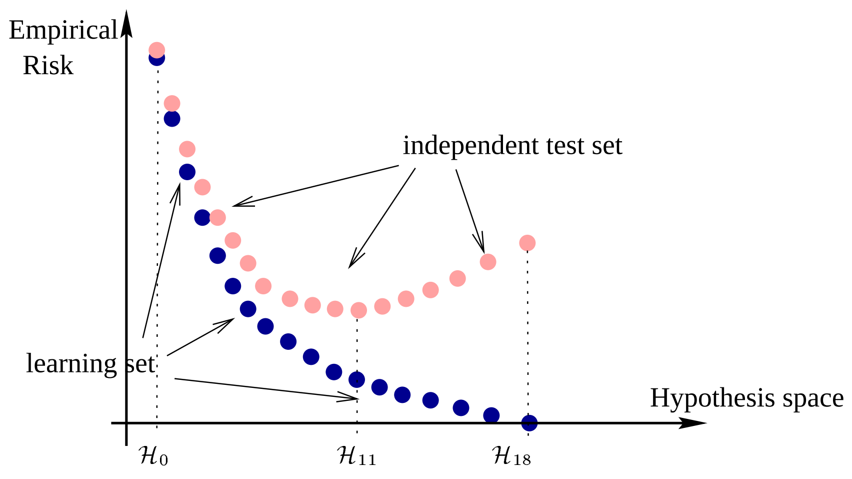


Figure 12: Structure selection by cross-validation

Clearly, the empirical risk minimization principle can not be applied to search directly for a model in the complete hypothesis space, since this would systematically lead to over-fitting. However, what can be done (and what is usually done) is to build a sequence of models for growing values of i by using the empirical risk minimization principle, and then to select among these latter the one which yields the best performance on an independent test set.

Figure 12 shows the typical behavior of empirical risk and test set risk as the size of the hypothesis space increases. One observes that the larger the hypothesis space the smaller the value of the empirical risk on the learning set, which is normal since the hypothesis spaces are nested. However, on an independent test set the risk increases significantly for overly large hypothesis spaces. For example, according to figure 12, the appropriate hypothesis would be the one found in $\mathcal{H}_{11}$, since above this level performance starts to decrease on the test set (this is similar to the cross-validation technique outlined earlier to select the appropriate value of β in the quality measure).

Nesting strategies. There are mainly two types of strategies to define a nested sequence of hypothesis spaces : bounding the structure, or bounding the norm of the parameter vector. Both may be best viewed as the truncation of an initially very large hypothesis space (i.e. using a very large number of parameters). They lead to the following definition of the sequence of hypothesis spaces

$$\mathcal{H}_i \triangleq \{f(\cdot, \boldsymbol{w}_m) | \boldsymbol{w}_m \in W_i\} \tag{16}$$

where $i < j \Rightarrow W_i \subset W_j$.

Bounding the structure the hypothesis space consists of forcing some components of the weight vector to zero, which would correspond to W_i being a subset of $\boldsymbol{w}_m$ vectors with at most i non zero components. This turns out to be equivalent to using a penalized quality measure like in eqn. (7), where the complexity of the model is proportional to the number of non-zero weights.

Bounding the norm of the parameter vector would consist of W_i being a subset of the form

$$W_i \overset{\triangle}{=} \{\boldsymbol{w}_m \in \mathbb{R}^m \ : ||\boldsymbol{w}_m|| \leq \lambda_i\}, \tag{17}$$

where λ_i is an increasing sequence of real numbers. This latter strategy is equivalent to using a penalized quality measure as in eqn. (7), where the complexity measure is proportional to the norm of the weight vector.

Notes.

1. The careful reader may have observed that the hypothesis space nesting principle that we have described here is similar to the principle which underlies decision tree pruning.

2. In the remaining part of this section we assume that the input variables are numerical or have been coded as real numbers.

Linear regression. Within the above framework, linear regression amounts to using the following model

$$f(\boldsymbol{x}, \boldsymbol{w}') \overset{\triangle}{=} \boldsymbol{x}^T \boldsymbol{w} + w_0. \tag{18}$$

Since this model is linear in the parameters, using a quadratic deviance measure leads to a quadratic empirical risk measure. The components of the gradient of the empirical risk with respect to the weight vector are therefore linear functions of the weight vector. Thus the optimal weight vector may be found by a linear equation solver, by solving the set of m linear equations

$$\frac{\partial R_e}{\partial w_i} = 0, \ \forall i = 1, \ldots, m, \tag{19}$$

where w_i denotes the i-th component of the weight vector.

In many applications linear models may provide very useful results. Also, in many cases the number of learning samples may be considered to be large with respect to the hypothesis space, and no complexity tradeoff is necessary. If this is not the case, one may resort to stage-wise linear regression, which amounts to applying the above nesting strategies to this problem.

The two main advantages of linear regression are computational efficiency, and the fact that it doesn't need very large learning sets to be tuned. For example, on the OMIB database, a simple linear model provides actually a rather good approximation of the CCT in terms of the two input attributes:

$$C\hat{C}T = -0.0004362Pu + 0.0000776Qu + 0.635 \tag{20}$$

which yields a mean absolute error of 8ms. The total CPU time required to learn this model using the 3000 learning states is of 200ms. Notice that actually, using only 100 learning states would have been largely sufficient: this results in the same accuracy but in a CPU time reduced to 10ms.

Generalized linear models. Unfortunately, many automatic learning problems may not be handled well by linear regression only. Hence, we will review a sequence of generalizations of this idea, starting with so-called generalized linear models, which are the most straightforward enhancement of linear models.

A generalized linear model yields the following form of hypothesis

$$f(\boldsymbol{x}, \boldsymbol{w}) \overset{\triangle}{=} \sum_{i=1}^{m} w_i f_i(x^{j_i}), \tag{21}$$

where the $f_i(\cdot)$ are a priori defined (possibly nonlinear) functions of one variable, and x^{j_i} denote particular input variables.

The main difference with respect to the basic linear model is that this model is not restricted to be linear in terms of the input attributes. However, it remains linear with respect to the parameter vector. Thus the same learning algorithm (direct solution of linear equations) is also applicable here.

The functions $f_i(\cdot)$ are generally called the basis functions of the generalized linear model. The set of basis functions used by a specific version of this method is called the dictionary of this method. In analogy to the orthogonal decision trees, we call this an "orthogonal" nonlinear model, because the nonlinearity acts only along the axes of the input space.

Generalized linear models are actually not at all restricted to orthogonal ones. They can be generalized one step further by

$$f(\boldsymbol{x}, \boldsymbol{w}) \overset{\triangle}{=} \sum_{i=1}^{m} w_i f_i(\boldsymbol{x}), \tag{22}$$

where the basis functions are defined in the multidimensional input space. This latter form yields the orthogonal model as a particular case.

For example, in the case of our example data base, we may obtain by an appropriate choice of these functions, the following quadratic model

$$\begin{aligned} C\hat{C}T &= 0.962 - 2.69\,10^{-8}(Qu)^2 + 3.35\,10^{-7}(Pu)^2 - 2.68\,10^{-8}(Pu)(Qu) \\ &\quad -0.001102Pu + 0.0001Qu, \end{aligned}$$

which is significantly more accurate than the previous linear model (mean absolute error of 1ms).

Let us mention that generalized linear models provide actually a very broad family of models, including trigonometric approximation and multidimensional spline approximation. The main difficulty with these methods is related to the fact that in high dimensional input spaces, they require prohibitively large dictionaries to yield good approximation properties. For example, a full quadratic model requires $\frac{k(k+1)}{2} + 1$ parameters (where k denotes the number of input

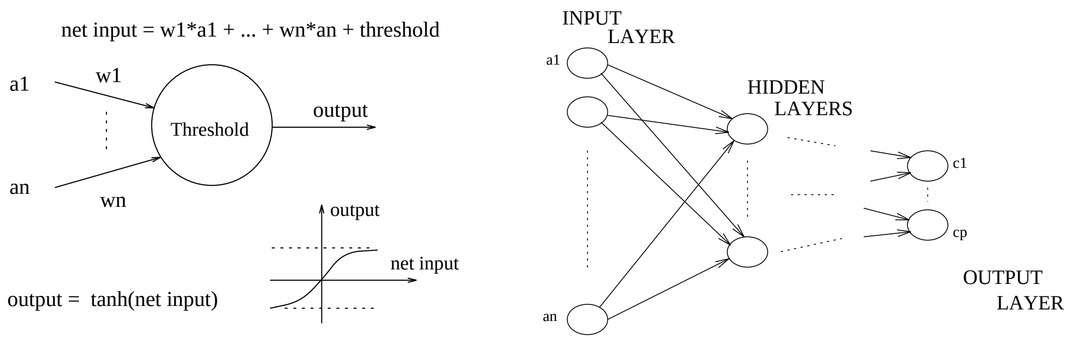


Figure 13: Perceptron (neuron) and multilayer perceptron (adapted from [21])

variables), which becomes prohibitive as soon as the number of input variables becomes larger than say a few tens.

This main weakness is circumvented both by multilayer perceptrons and by the projection pursuit regression technique, which we describe next.

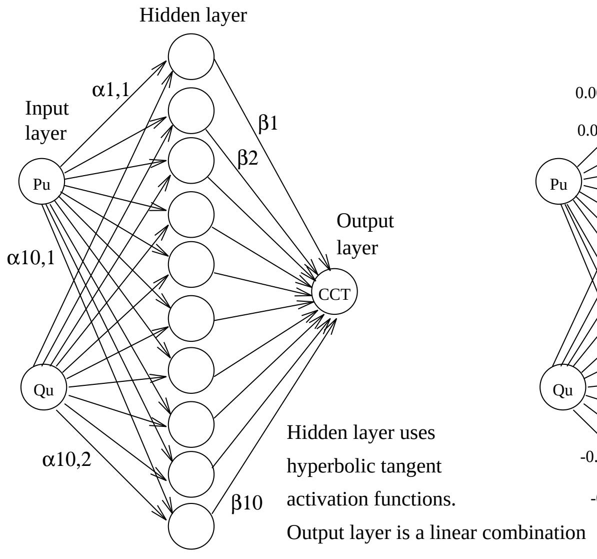
Multilayer perceptrons. Artificial neural networks' development started several decades ago with the work on perceptrons. Figure 13 illustrates the perceptron; this is basically a simple linear threshold unit, thus able to represent only linear functions. Its limited representation capabilities have motivated the consideration of more complex models composed of multiple interconnected layers of perceptrons (multilayer perceptrons), which provide a very flexible family of models.

Figure 13 illustrates a typical multilayer perceptron (MLP). Each neuron is a perceptron : input layer neurons are fed with the input attributes; hidden and output layer neurons receive linear combinations of outputs from neurons in the preceding layers.

MLP learning. In the context of multilayer perceptrons, the learning stage consists of determining an appropriate structure of the MLP and of identifying appropriate values of the different parameters (weights and thresholds).

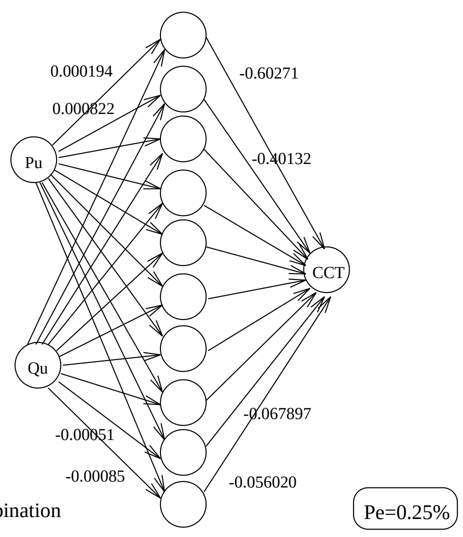
Choosing the structure consists of defining the number of neurons, the topology of their interconnection, and the type of activation functions they use. Usually, it is determined by a trial and error procedure. However, nowadays there exist various algorithms to determine the structure automatically.

The parameter identification task amounts to a (complex) nonlinear numerical optimization problem, which may be solved by various techniques. Historically, the first method which was proposed was the so called "back-propagation" algorithm, which is equivalent to a fixed step gradient descent technique. It is interesting from a biological point of view, but rather inefficient from a computational point of view. Nowadays, one uses generally second order quasi-Newton methods.



Hidden layer
Input layer
α1,1
β1
β2
Pu
Output layer
α10,1
CCT
Qu
α10,2
β10
Hidden layer uses hyperbolic tangent activation functions.
Output layer is a linear combination

(a) Initial values of the weights are random



(b) After training the weights are tuned to fit the problem

Figure 14: Single hidden layer perceptrons (adapted from [21])

MLP parameter identification. In our illustrative problem MLPs can be exploited interestingly to approximate the CCT as a closed form function of Pu and Qu. Generally, for the approximation of a continuous function, MLPs with a single hidden layer may provide good approximators. Thus, let us try to approximate the CCT with such a structure. The learning set used will be the same 3000 input states that we used up to now, and as in the preceding examples, we associate as output information to each state its CCT, rather than the security class.

Figure 14 graphically sketches the MLP that we have used, containing 10 hidden neurons. Each hidden neuron i has an input/output relationship of the form

$$\text{Output}_i = \tanh(\alpha_{i,1}Pu + \alpha_{i,2}Qu + \theta_i), \tag{23}$$

where $\alpha_{i,1}$ (resp. $\alpha_{i,2}$) is the connection weight between the neuron and the Pu (resp. Qu) input, and θ_i its threshold.

The output of the MLP is a linear combination of the preceding functions, i.e.

$$\text{C}\hat{\text{C}}\text{T} = \sum_{i=1...10} \beta_i \tanh(\alpha_{i,1}Pu + \alpha_{i,2}Qu + \theta_i), \tag{24}$$

where β_i represents the contribution of neuron i in the overall output.

The parameter identification thus aims at choosing appropriate values of the 40 parameters $(\alpha_{i,j}, \theta_i, \beta_i)$ in order to fit, for each learning state, the MLP output to the CCT value determined by numerical simulation. The empirical risk criterion we use is the mean square error

$$R_e = \frac{1}{3000}\sum_{i=1}^{3000} |\text{CCT}_i - \text{C}\hat{\text{C}}\text{T}_i|^2, \tag{25}$$

which is a smooth, although complex and nonlinear function of the parameter values, which needs to be minimized.

Before starting the learning procedure the parameters are all initialized at random, then they are progressively adapted in order to minimize the empirical risk. In our example we used the Broyden-Fletcher-Goldfarb-Shanno (BFGS) optimization method.

At initialization, the random initial parameter values of the MLP yielded a value of $R_e = 0.0689568683$, corresponding to a root mean square (RMS) approximation error of 0.2626 seconds, which is three times larger than the standard deviation of CCT values in the learning set. In short, the initial output values are random.

After 46 iterations the algorithm stops at a local minimum, having reduced the value to $R_e = 0.0000003136$, which corresponds to a very small RMS approximation error of 0.00056 seconds. The mean absolute error in the learning set is equal to 0.4 milliseconds which is actually smaller than the accuracy of the CCT values determined by numerical simulation. Thus, we deem that we are close to the global minimum.

All in all, the parameter adaptation process took about 200 CPU seconds on an UltraSparc workstation.

The resulting closed form approximation of the MLP input/output function corresponding to the final parameter values is as follows

$$\begin{aligned}
\hat{\text{CCT}} = \; & -0.602710 \tanh(0.000194Pu - 0.00034Qu - 0.93219) \\
& -0.401320 \tanh(0.000822Pu - 0.00020Qu - 0.76681) \\
& +0.318249 \tanh(0.000239Pu - 0.00050Qu - 0.29351) \\
& -0.287230 \tanh(0.002004Pu - 0.00034Qu - 1.20080) \\
& +0.184522 \tanh(0.000131Pu - 0.00057Qu - 0.03152) \\
& +0.177701 \tanh(0.001799Pu - 0.00011Qu - 2.08190) \\
& -0.150720 \tanh(0.001530Pu - 0.00056Qu - 1.68040) \\
& +0.142678 \tanh(0.002152Pu - 0.00046Qu - 1.72280) \\
& -0.067897 \tanh(0.001910Pu - 0.00051Qu - 1.71343) \\
& -0.056020 \tanh(0.000202Pu - 0.00085Qu - 0.39876)
\end{aligned} \quad (26)$$

MLP validation. In order to evaluate the reliability of this approximation, we have used the MLP to predict the CCTs of the 2000 test states. Figure 15 shows the distribution of errors; it is clear that in this simple example the MLP approximates the CCT with very high accuracy. Furthermore, we observe that the mean absolute error in the test set (0.4ms) is equal to the mean absolute error in the learning set. Thus, the MLP generalizes very well to unseen states. However, the improvement in accuracy is marginal with respect to the above quadratic model.

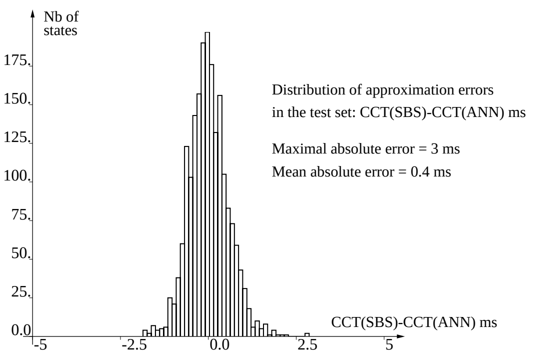


Figure 15: Distribution of MLP approximation errors (test set) (taken from [21])

Consequently, the MLP can be used in order to classify states with respect to a threshold. For example, with respect to the threshold of 155ms used in the decision trees, it classifies 5 insecure states as secure (their CCT is however very close to the threshold of 155ms) and makes no false alarms, i.e. its error rate is of 0.25%.

Note that since the MLP provides a very accurate closed form approximation of the CCT, its sensitivities with respect to Pu and Qu may be computed analytically. These could in turn be used to find out preventive control actions to increase the value of the CCT whenever it is found to be too small.

Refinements. Although in many problems a single hidden layer is sufficient, it is straightforward to generalize the MLP by adding any number of further layers. It is also possible to use other activation functions than the hyperbolic tangent, e.g. Gaussian or trigonometric functions.

Another extension consists of growing the neural network by progressively adding neurons and/or layers. Pruning techniques, with fancy names like “optimal brain damage” and “optimal brain surgeon”, were also designed to reduce over-fitting by removing useless connections and neurons. There are even techniques (e.g. the projection pursuit regression method discussed below) able to adapt automatically the shape of the activation functions to the problem features.

Finally, let us mention that the MLP learning algorithm may be used in an adaptive on-line scheme, so as to adapt parameters whenever new learning states become available.

Salient features of MLPs. Notice that the very high accuracy obtained in our OMIB example is due to the fact that we used a very large learning set in order to approximate a rather smooth function of only two input parameters, and that we have used a moderate number of parameters in the MLP. Further, our OMIB database is free of noise.

However, in many practical large scale applications the conditions are generally less favorable and it is not possible to reach this level of accuracy. Nevertheless, MLPs are often among the most reliable automatic learning methods.

Thus, the main characteristic of MLPs is flexibility in approximating nonlinear functions in multi-dimensional spaces.

This flexibility is however obtained at the expense of high computational burden. In real-life problems, when the number of inputs and hidden neurons is large, training times are typically of several hours to several days. At the same time, it becomes rather difficult to appraise and interpret the type of input/output relationship represented by such an MLP, which thus behaves like a black box.

Projection pursuit regression. Projection pursuit regression has been proposed by Friedman in the early eighties [26]. It uses the following form of hypothesis

$$H(\boldsymbol{x}) = \sum_{i=1}^{M} \beta_i H_i(\boldsymbol{x}^T \boldsymbol{w}^i), \tag{27}$$

where the functions H_i belong to a one-dimensional hypothesis space. The learning algorithm, which we will not describe here for the sake of space, selects the following “parameters” : M (the order of the model), and for each $i \leq M$, β_i (the weights of individual directions), $\boldsymbol{w}^i$ (the individual directions), and $H_i(\cdot)$ (the individual one-dimensional models).

Thus, the model is similar in structure to a single hidden layer perceptron, with the important difference that the activation functions are learned rather than chosen a priori by the user. It is not astonishing that the main drawback of this very sophisticated method is high computational requirements.

However, the method is very flexible and has the main advantage with respect to multilayer perceptrons to provide more interpretable results. In particular, the method may be downgraded into an orthogonal projection pursuit model, which is quite easy to interprete and may be implemented in a very efficient way [27].

Let us mention that in the recent years, work in projection pursuit regression has mainly focussed on the definition of appropriate one-dimensional hypothesis spaces. For example, references [27, 28] describe an efficient method to derive one-dimensional piecewise linear models H_i, and how to exploit them effectively in the multi-dimensional learning problem.

3.3.3 Nearest neighbor methods

All the methods described up to now (decision trees, multilayer perceptrons, projection pursuit . . .) essentially compress detailed information contained in the learning set into general, more or less global models (rules or real-valued functions).

Additional information may however be provided in a case by case fashion, by matching an unseen situation with similar situations found in the database. This may be achieved by defining generalized distances so as to evaluate similarities among objects, together with appropriate fast

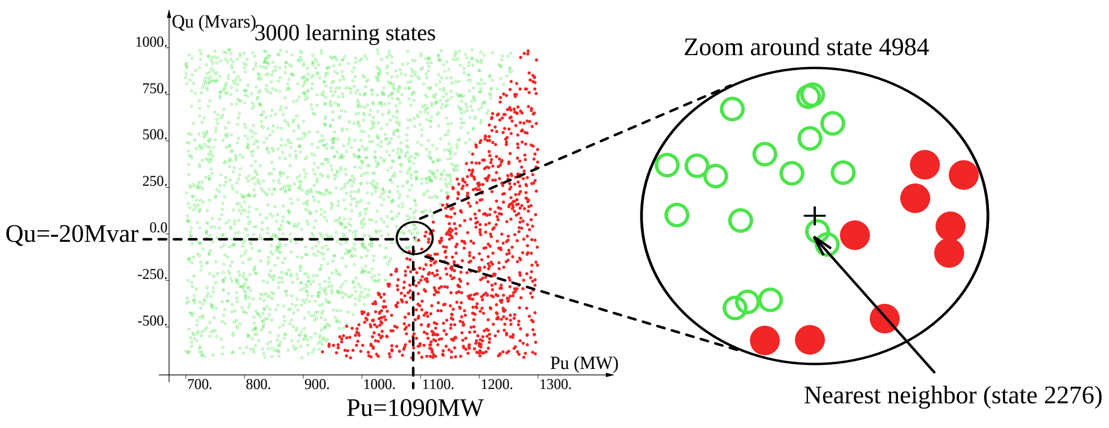


Figure 16: 3000 learning states and nearest neighbors of state 4984 (taken from [21])

database search algorithms. Let us briefly illustrate the so-called "K nearest neighbors" (KNN) method.

KNN consists of classifying a state into the majority class among its K nearest neighbors in the learning set. In its most simple version, the learning stage of the KNN method merely consists of storing the learning states in a table. The actual work (computing the distances and sorting out the K nearest neighbors) is done when the method is used to make predictions for unseen states.

For example, in our illustration let us consider the state no 4984 of our database (a test state). Its values of Pu and Qu are respectively of 1090 MW and -20 MVAr. Figure 16 shows in its left hand part the location of this state in the attribute space together with the learning states. In the right hand part we have zoomed on the nearest neighbors of the state. Note that the points on the borderline of the zoom region are equidistant (Euclidean distance) to the test state.[6] One may identify on figure 16 the nearest neighbor, i.e. the learning state closest to the test state (state no 2276 : Pu=1090 MW, Qu=-31 MVAr, and CCT=0.157s). Thus, according to the 1 nearest neighbor (1NN) rule, the CCT of the test state will be approximated to 0.157s and it would be classified into the secure class. Note that its actual CCT is equal to 0.158s; hence the state is correctly classified, in spite of being very close to the security boundary.

Validation. Repeating this procedure for all 2000 test states yields an error rate of 0.9%. Figure 17 shows the distribution of CCT approximation errors. Comparing with figure 15, we notice that the 1NN approximation is slightly less accurate than the MLP approximation. On the other hand, the 1NN provides additional information to that of the MLP and the DT : the distance to the nearest neighbors, attribute values of the nearest neighbors, and more generally any type of information attached to the nearest neighbors, like, for example, optimal preventive or emergency control strategies.

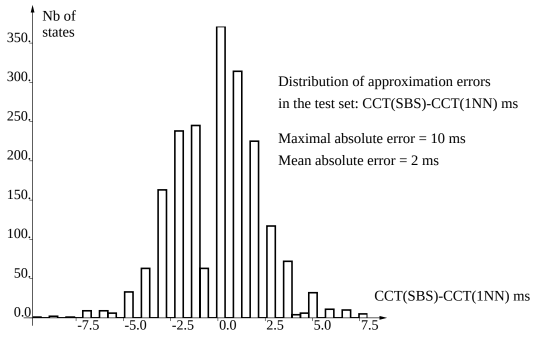


Figure 17: Distribution of 1NN approximation errors (test set) (taken from [21])

Refinements. The basic refinement consists of using K neighbors instead of a single one, K being determined on the basis of the learning set to increase reliability. Then, since the nearest neighbor rule is quite sensitive to the distance chosen, in many practical problems it is necessary to reduce the weights of less relevant attributes with respect to more relevant ones. Thus, distance learning algorithms have been devised so as to choose automatically the weights on the basis of a learning set. A further refinement consists of using different distance definitions in different regions of the attribute space.

Salient features of KNN. The main characteristics of this method are high simplicity but also sensitivity to the type of distances used. In particular, to be practical, ad hoc algorithms must be used to choose the distances on the basis of the learning set.

The fact that the KNN approach is quite similar to human reasoning (recalling similar situations seen in the past) makes it also interpretable by human operators.

Combined use and hybrid methods. Now that we have described so many methods, which all seem to be in competition, what is our message ?

The first remark is that, in terms of accuracy, while the ranking shown on our illustrative problem is quite representative of many problems, there are also many exceptions. Thus, although in general MLPs are among the more accurate methods, and decision or regression trees among the rougher ones, this is not always the case.

The second remark, is that the methods that we have presented are essentially complementary in terms of functionalities : decision trees and simple linear models are easy to interprete, and can be used effectively on very large databases in an interactive scheme. On the other hand, MLPs are slow at the learning time, and even though computers become faster and faster, this remains essentially a batch type of procedure, which is also true for full-fledged projection pursuit regression. In the same way KNN is complementary to both types of methods, but rather slow in practice.

The third remark is that nothing prevents one from using all methods in parallel on a database, and to select (if accuracy is the main focus) the method which provides the best results on an independent test set.

Finally, it is also possible to combine the methods in hybrid approaches. For example fuzzy decision trees result in an intimate coupling of techniques from recursive partitioning and numerical optimization à la MLP.

More simply, it is possible to use classical decision or regression tree induction in order to identify the most relevant attributes and then use these attributes as input variables to the computationally more laborious methods. This helps to increase interpretability and reduces computational burden. It may even lead to improved performances in terms of accuracy [21].

3.4 Unsupervised learning

Unsupervised learning aims at discovering interesting patterns in a database, without specifying in advance which attributes should be considered as input variables and which attributes should be used as output variables. In the literature on automatic learning, many unsupervised learning methods have been proposed, such as clustering techniques, mixture distribution fitting, correlation analysis, self-organizing feature maps, Bayesian networks, associations rules, etc.

In what follows we focus on some methods which may be used in order to identify coherent groups of objects or variables. Before describing the methods, let us briefly discuss some potential practical uses of these applications in the context of power system analysis.

3.4.1 Potential uses of unsupervised learning methods

Correlation analysis. This aims at identifying among a set of candidate attributes, groups of attributes which are strongly correlated (see the illustration in section 3.4.3). In power systems this is quite useful, since in large scale applications the number of attributes describing power system scenarios may easily reach several hundreds of variables (e.g. the voltages at all the buses of the system, the power flows in all the lines of the system, etc.).

Thus, unsupervised learning may be used in order to find coherent groups of variables, i.e. variables which essentially provide the same information about the scenarios. This in turn allows one to reduce significantly the number of attributes to consider for further analysis, which results in better interpretability, lower computational requirements, and often increased accuracy.

Similarity analysis. Here the objective is to identify groups of similar behaviors. For example, considering the left hand part of figure 4, one can easily observe that in the OMIB database there are mainly two types of rotor angle behaviors (bounded and unbounded) which actually

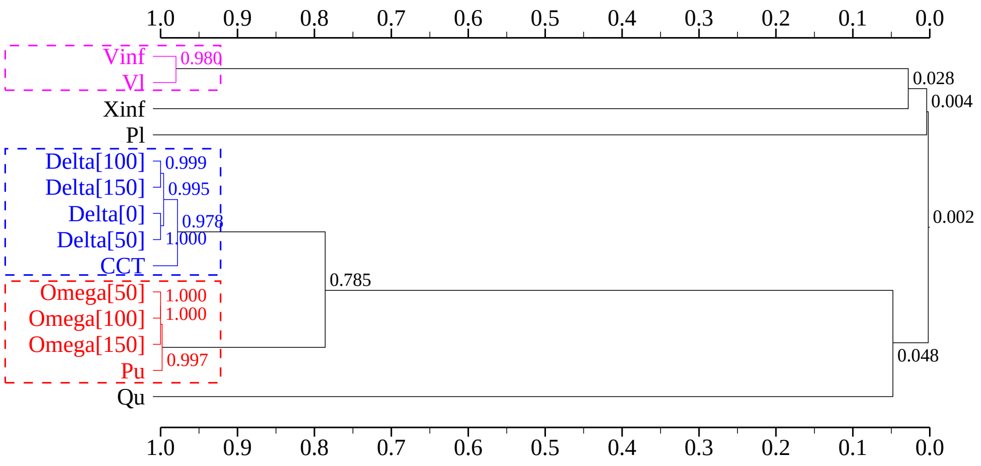


Figure 18: Dendrogram : correlation analysis of OMIB database attributes

correspond to the secure and insecure classes. In section 4 we mention a more sophisticated application of clustering techniques in the identification of blackout modes of a real large scale power system.

3.4.2 Unsupervised learning principles

In unsupervised learning there are essentially two basic principles.

The first principle is based on a probabilistic framework : it states that unsupervised learning is a matter of modeling (unconditional) joint probability distributions in the attribute space as a mixture of basic parametric distributions. A well known representative of this approach is the AUTOCLASS method [29], which is based on a Bayesian approach to probabilistic modeling. The principles behind mixture distribution fitting are essentially the same than those that we discussed earlier in the context of conditional probability model learning.

The second principle (the one which we will illustrate in what follows) is purely geometric. It starts by defining a distance measure between objects, and then considers that a group is a cluster of objects which are as close as possible to each other, and at the same time far away from objects not belonging to the group. The unsupervised learning algorithms based on this principle thus essentially consist of searching the data for a certain number of “good” clusters. Below we will describe two different search strategies which may be used.

3.4.3 Hierarchical agglomerative clustering

This method is particularly attractive when the number of objects to group is not too large.

Starting with a set of objects, the idea is to build the clustering in a bottom up fashion, by grouping progressively the most similar objects (or groups of objects) until only a single group remains containing all objects.

The application of this algorithm to the OMIB database is illustrated on figure 18, which shows a dendrogram summarizing a correlation analysis among various attributes : the six input attributes, the CCT, rotor angle and speed at different time instants (“delta(t)” and “omega(t)” denote the value at time t in milliseconds).

The dendrogram was built in the following fashion :

- computation of the linear correlation coefficients among all pairs of variables (the numbers on the diagram represent the absolute value of correlation coefficients);
- definition of the similarity of two variables as the absolute value of their correlation;
- definition of the similarity of two groups of variables as the minimum correlation coefficient of pairs of variables across the two groups;
- iterative grouping : starting with one group for each variable, recursively group the two most similar groups of variables until all the variables belong to the same group.

Thus, figure 18 shows, in particular, three groups of strongly correlated attributes, from top to bottom

- the two voltages Vinf and Vl (correlation coefficient of 0.98);
- rotor angles and CCT (minimum correlation coefficient of 0.978);
- rotor speeds and Pu (minimum correlation coefficient of 0.997).

3.4.4 K-means

While the agglomerative clustering method is quite attractive for interpretation purposes, it is mainly useful in order to group a moderate number of objects. The K-means method, on the other hand allows one to group a very large number of objects into a given number of groups.

The algorithm (see e.g. [21]) is basically an iterative optimization technique. For a fixed number of groups (say k), it first divides the set of objects into k disjoint subsets (using a random guess), then it iteratively moves objects from one group to another in order to improve the quality of the overall clustering. The latter quality is measured by the total within scatter of the k clusters : the scatter of a cluster is the sum of the euclidian distances from the objects of this cluster to the cluster center. The algorithm stop at the first local minimum, i.e. when it is impossible to improve the quality by swapping any object.

Figure 19 illustrates how curve coloring can be useful to analyze the results of a clustering algorithm. A random sample of the OMIB database was automatically partitioned by the K-means method into 15 classes, according to the values taken by the temporal attribute $\delta(t)$, i.e. rotor angle temporal behavior. The $\delta(t)$ curves of the scenarios belonging to three of these clusters are shown in figure 19.

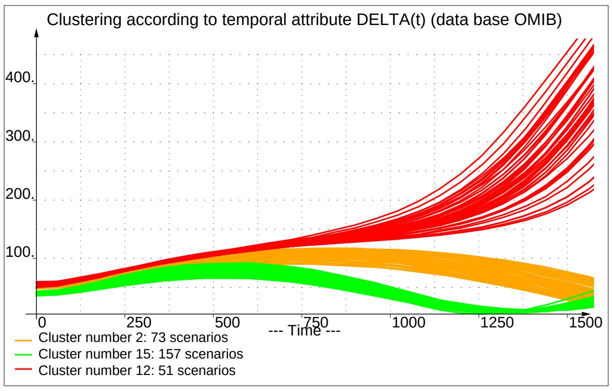


Figure 19: Clustering of OMIB scenarios according to temporal behavior of rotor angle

3.4.5 Hybrid methods

One of the difficulties with the K-means method is due to the fact that the user must define in advance the number of clusters to search for. But this number is usually not known, which leads to a trial and error procedure to determine the right number of clusters.

Another possibility consists of combining K-means and agglomerative clustering in order to identify a correct number of clusters in the database. This leads to the following procedure :

- first use the K-means algorithm with a rather large number of clusters, say between 20 and 100 (depending on the database size);
- then use the hierarchical agglomerative clustering method in order to analyze the distances among the resulting cluster centers and decide for a new value of k;
- run again the K-means algorithm with the new (smaller) value of k;
- iterate this one or two times.

In order to illustrate this approach we have applied it to our OMIB data base. First, we used a set of 3000 objects of the first part of the data base, and used an initial number of 20 clusters for the K-means algorithm. Applying the agglomerative clustering algorithm to the latter 20 clusters yielded the left hand dendrogram of figure 20, which suggests that an appropriate number of clusters would be 9.

Then we applied the same approach to 3000 objects of the second part of the data base (which is in principle richer, since more parameters are variable). Again starting with 20 initial clusters, the approach yields the right hand dendrogram of figure 20, which suggests that the appropriate number of clusters is rather 10 for this part of the database.

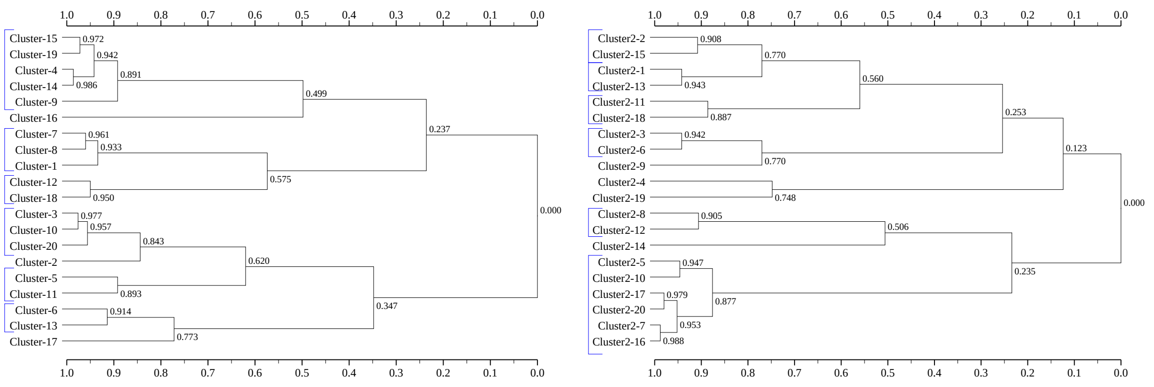


Figure 20: Dendrograms of K-means results

3.4.6 Similar episodes and time-series

As we have illustrated above, in our applications there are many situations where the similarity measure relates to similar dynamic behavior patterns. In some cases, such as those already illustrated, it is possible to translate time series (curves) into vectors of real numbers and then to use the euclidian distance between these vectors as a measure of similarity. This approach was used above, and was possible, mainly because the curves are all synchronized (they have a common origin of time and a common duration).

However, in many applications the data is not represented in such a systematic way in the database. Thus, it is necessary to define more flexible criteria to compare time series among each other. Techniques which might be used for that purpose are for example signal transformation techniques (fast Fourier transform, wavelets, etc.), signal correlation, or deformation based curve matching [30, 31, 32].

3.5 Other data mining tools

In this section we have screened various complementary techniques to extract information from databases, focusing on nontrivial automatic learning algorithms. Our objective was to provide a comprehensive and self-contained description of the main useful techniques which are today at our disposal.

Of course, to provide a complete description of existing automatic learning algorithms is impossible and not very useful. We have provided some references for further reading at some places of the text, but here also we have limited our references to either very basic ones or pointers to very recent work which might not yet be known.

Of course, data mining does not only use automatic learning algorithms, even though they are at the center of the process. Typically, data mining software contains also a large number of auxiliary tools and fancy user interfaces. In particlar, they should provide facilities to select subsets of scenarios and variables, to visualize them easily and to show data mining results to the user. Many of these features have been illustrated through the examples that we have provided. Other features which are necessary for data mining, and which we did not discuss are :

Data transformation and cleaning techniques : replacement of missing values, normalization, windowing, identification of outliers;

Interactive definition of attributes : in order to allow the user to combine different ground attributes into more sophisticated ones;

Automatic learned models : automatic learning builds new functional attributes and creates new objects from those contained in the initial database; these new variables and objects should be incorporated in the environment in such a way that one can use them in the same way as the other variables and objects.

4 Applications in power systems

We now discuss more in detail the applications of automatic learning and data mining to power system problems. We start, in section 4.1 with an overview of those applications which can use automatic learning in one way or another. Then, in section 4.2 we will discuss more in detail the "study" applications which may use the framework of Figure 1, combining Monte-Carlo simulation with automatic learning. In this latter part we will provide some references to publications reporting different studies in detail.

4.1 Overview of main application fields

In this section we screen through the main applications of automatic learning and data mining in the context of electric power systems. Our objective is to illustrate the diversity of possible applications and to suggest new directions not yet explored in detail.

4.1.1 Dynamic Security Assessment

Power system security assessment has the special feature that the data bases are generated by computer based simulation, and this because the actual experience of instabilities is fortunately very small for most power systems. This is a laborious task but in the same time an advantage, being possible to generate as many scenarios as are deemed necessary to solve a particular task.

Possible uses of automatic learning to security assessment would be:

- providing continuous security margins for a given scenario: CCTs, load power margins, severities, global stability limits;

- identifying the relevant attributes for the security class prediction of a given power system;
- determining if an operating state is stable or not with respect to a given fault, or to a set of contingencies;
- determining which operating points lead to a voltage collapse for a given contingency;
- determining for a given operating state the faults likely to create instability;
- contingencies ranking in terms of their severity for a range of operating conditions;
- identifying conditions characterizing pre-fault attributes of stable operating states for a given set of possible faults;
- partitioning all the possible scenarios into families of dynamic behaviors: stable and failure modes; determining criteria to identify failure modes in real-time;
- partitioning the power system map into voltage coherent behavior regions;
- searching for a specific kind of scenario;
- early detection of voltage risks for a given scenario;
- inferring means for control if an incipient instability is detected early enough;
- extracting if-then rules from all the problems solved by means of machine learning.

4.1.2 Controller and Protection Design

In power systems, design problems mainly concern the identification of appropriate locations for some devices, their dimensioning and the tuning of some of their parameters. Typical applications are: study concerning a new static VAr compensator (SVC) (the best geographical location, type and capacity, possibly parameters tuning), re-tuning the parameters of power system stabilizers equipping some plants, selecting the number and location for new measurement devices, defining pilot nodes for secondary voltage control systems, tuning a transformer tap-changer blocking device.

There is a variety of control problems, concerning centralized/local controllers operating at different time-frames and acting on different physical variables. These controllers need to be robust with respect to changing characteristics of a power system, due to variation of load level and topology. Since the latter are generally non-linear it is necessary to use simulation tools in order to validate controller robustness and sometimes tune them. Automatic learning can be used in order to make such studies more systematically, taking better advantage of simulations.

A automatic learning process applied to the above SVC design problem should start with the identification of the electrically “coherent” regions in the system. A small number of voltage coherent regions may be found out by means of unsupervised learning, analyzing correlations among voltages at different nodes of the system. Then, a more refined analysis of the individual voltages in each region would allow to identify “central” buses where the overall behavior of the zone is best observed. To study the best substation to place the SVC and to find out which parameters of the operating point would influence most strongly the behavior of the device, a new data base is generated by random sampling different possible locations, operating conditions, assumptions and modelings (steady state, small signal, large disturbances), and various automatic learning tools may be implied. In order to finalize the design, a third more refined study could be carried out for fixing SVC parameters: several possible sizes and parameters could be screened out.

Adaptive (control and protection) systems are an alternative to robust designs, in order to cope with the variety of operating conditions under which such systems must operate. Automatic learning would allow one to determine the parameters to which the optimal control law of a plant is most sensitive and then learn the scheduling rules to select the appropriate control law in real-time.

4.1.3 Modeling

In order to carry out simulations, be it for security assessment or design, it is of course necessary to build up models; while many components in the power system may be modeled rather accurately, by choosing model structure from first principles and using classical identification techniques to determine parameters, some other parts are intrinsically difficult to model, either because they need to be represented in a reduced way (this is typically the case of load models and external system models), or because they change with time (due to aging and/or modifications carried out during maintenance operations) or they vary significantly from one geographical region to another (due to weather conditions). We believe that automatic learning could be a very useful tool in order to improve the latter type of models, and also to monitor the former ones in a systematic fashion in order to develop better models.

Data to be mined can be available in this case by recording information from real-time measurement systems and disturbance records. By data mining tools, it would be possible to estimate how accurately their derived model approximates real-life, i.e. how much uncertainty remains which is not explained by the model. And this information may be used in the context of problems using data base generated by means of these models. For example, a certain standard deviation of the security margin in a generated data base, due to uncertainty in the modeled load distribution, translates into inevitable classification errors when trying to assess the security class of a scenario. Then we know that, for minimizing these errors we should try to model more accurately the load, rather than to further search for improving data mining methods.

4.1.4 Estimation, Identification, Forecasting

In order to build up and maintain statistical models about load patterns, measurement and signaling errors, data gathered in control centers archives may be exploited by automatic learning tools.

Short-term load forecasting is one of the areas very investigated from the point of view of neural networks application (supervised and unsupervised). In this context, a data base may be easily updated to contain past load demands and past data about some typical factors: local/global weather conditions (temperatures, winds, humidity), type of the hour, type of the day, tariff strategies, social events like national days off, strikes. Mining this type of data gives a better understanding about what parameters are more important for the load evolution, how different geographical regions act from the point of view of weather conditions and how this influences the load, and accurately predict the load demand for a given new situation [33].

An interesting area for research concerns very short term *nodal* load forecasting, to be used in enhancing real-time system monitoring and security assessment. Such predicted values would enrich the information exploited by state estimators, providing the required redundancy to detect topological errors. Further, obtaining a 30-minute prediction of the individual load demands would permit security assessment on predicted state rather than on past information.

4.1.5 Monte-Carlo based Planning

In the off-line planning environment, engineers use Monte-Carlo simulations in order to estimate expected values of quantities such as operating costs and load-curtailment for future operating conditions. It consists of using random sampling to screen possible scenarios according to their probability, together with numerical methods to compute for each scenario the operating cost and amount of load curtailed, and averaging the values for the simulated scenarios. Presently, most tools used within this context compute only global sample statistics (averages, standard deviations, quantiles) of important quantities chosen by the engineer, but throw away the details concerning each individual sample.

Preliminary investigations show that much richer information can be extracted by storing the results from individual samples (simulated operating scenarios) in a data base, and further analyzing the latter by automatic learning. For example, this approach allows one to detect outliers (which are often responsible of high variance and bias in the estimates) and lets the engineer judge whether or not they must be taken into account. Also, by carrying out systematic studies of relationships (correlations) between different output parameters, better insight into the physical structure of costs and other decision variables can be gained, which eventually will lead to better strategic decisions.

Automatic learning can also be used in order to compare the distributions of simulated scenarios and those encountered in real life, which in turn leads to improvements in the probabilistic models used in the Monte-Carlo simulations. Furthermore, by carrying out the analysis of variance of output parameters, it is possible to find out in which region of the sampling space variance is higher, and thereby to define more efficient sampling schemes. In the same way, regression techniques may be used in order to build automatically approximate input/output models which can then be used as control variates in order to further reduce variance [34, 35].

4.1.6 Monitoring

Monitoring is a very broad application field, ranging from the monitoring of the individual devices and plants, to sophisticated system monitoring. The purpose is to detect as early as possible when a system departs from its normal conditions. It includes many of the activities of a control-room, in particular security assessment. Power plant monitoring (burner, reactor, dam), transformer monitoring, and system oscillations monitoring are applications which can make use of automatic learning.

In particular, recently temporal decision trees have been proposed as an appropriate tool for monitoring, given the dynamic time-varying nature of attributes and the ability of temporal trees to early predict modes of bad behavior (e.g. fast electromechanical transients, mid-term voltage collapses, cascades of overloaded line tripping, etc.) [25].

Some of these applications have already received attention from researchers, but many possibilities still remain to be explored. In particular, so as to respond to present un-bundling trends, future system monitoring will mostly rely on hierarchical approaches. In this hierarchy, the lower levels will correspond to local monitoring of plants or sets of plants of individual generating companies or load aggregators, and the higher levels will correspond to security and transaction monitoring at the system level. At the lower levels, the problem will be to monitor system changes which may impact the functioning of local plants, from local information and possibly global status information obtained from the upper level monitoring system. At the upper levels, the problem will be to use global information together with outputs of local monitoring systems in order to provide an overall picture of the system, and identify those abnormal states which cannot be identified from local information only. Which information to share between lower and upper levels, and how to make the best use of it will be the main questions to solve. We believe that probabilistic simulations, together with automatic learning and other intelligent information processing approaches may provide valuable tools in this context.

4.2 Application to power system design, analysis and control

Among the applications of automatic learning just discussed, we may distinguish two classes according to the nature of the databases used : (i) applications using data collected from the field (modeling, state-estimation and identification, monitoring); (ii) applications requiring numerical simulations in order to build databases (security assessment, controller design, planning).

Here we focus on the latter class of applications. They are on the one hand more complex, since they need software and models developed for the generation of the databases. On the other hand, due to the fact that the database generation is under the control of the engineer, in these applications there is increased flexibility.

Below we will outline the overall approach used to study electric power systems through computer experiments. The main idea is to take advantage of existing computer simulation tools and physical models together with automatic learning. The bridge between these two techniques is provided by Monte-Carlo simulations, which exploit existing simulation tools in order to produce information which may be exploited by automatic learning.

First of all let us discuss the nature of the different problems encountered in the field of power system design, analysis and control.

4.2.1 Introduction

Let us first recall that power systems are complex, large scale, highly nonlinear and uncertain systems. The satisfactory operation of a power system requires extensive studies with highly diverse horizons. To illustrate the complexity of the decision making process, let us focus on security assessment and control, which is certainly one of the most difficult and at the same time important problems faced by power system engineers.

Security assessment and control aims at making decisions in different time horizons so as to prevent the system to experience undesired situations, and in particular to avoid large catastrophic outages. Traditionally, security control has been divided in two main categories : preventive and emergency control (EC).

In preventive security control, the objective is to prepare the system when it is still in normal operation, so as to make it capable of facing future (uncertain) events in a satisfactory way. In emergency control, the disturbing events have already occurred, and thus the objective becomes to control the dynamics of the system in such a way that consequences are minimized.

Security related decision making starts in (national or international) regulation bodies which define security criteria and auditing principles, and ends by paying back customers who have suffered economic loss due to insecurity. Let us briefly enumerate the steps as they appear in a logical sequence.

Security standards. The explicit definition of security standards is needed in order to set targets for security control. Clearly, security standards will depend on the role played by the electric energy system in the overall economy, as well as on political choices. The standards definition must comprise the specification of the desired levels of reliability, of the means associated to reach the stated objectives, the rules for allocating resulting costs and the auditing mechanism used for verification. Generally, at this level it is also decided which entities (private or public) are technically in charge of meeting the security standards.

Research. Research basically aims at understanding how electric power systems behave, in order to provide predictive models which may be used at the different levels of decision making. Today, the trend in research is clearly towards exploiting information technology (computation, database management, communications), which provides means for improved modeling and security control. Let us notice that since the electric power system is essentially stochastic, modeling should address both physical and probabilistic aspects.

Long term investment. Formerly, in the integrated framework, investment decisions were generally taken in a coordinated way, combining expansion of generation and transmission subsystems in an optimal way. Today, in order to favor competition, in most systems the investment of generation is a matter of independent decision making based mainly on business opportunities. Hence, the transmission system investment must follow in order to maintain desired levels of security. Building new lines is for the time being almost impossible in most developed countries, thus investment might probably focus on improving power system monitoring and control, by exploiting modern communication possibilities and power electronics.

Maintenance planning. Decision making aims at choosing maintenance plans in such a way that availability is maximized in periods where higher traffic on the system is expected. Again, since the generation subsystem is operated independently by a certain number of agents, there will be increased needs for probability methods in order to manage uncertainties. Also, in order to reduce uncertainties it will be necessary to be more reactive, in order to adapt maintenance plans smoothly, as information becomes available.

Transaction planning. In short term (one day ahead, typically) transaction planning, the system structure is essentially fixed and the objective is to arbitrate among conflicting transactions without discrimination and while ensuring system security.

Operation. In the control room, the operator receives real-time information which is used in order to coordinate the scheduled transactions, while reacting to unforeseen events (disturbances, outages, unexpected behavior . . .). Part of the operators' job will be to handle 'slow' emergencies, e.g. related to thermal overload problems or slow voltage collapses.

Emergency control. In this category we focus on automatic control actions such as generation tripping, load-shedding, controlled islanding . . . We will further elaborate on this in the remaining part of the paper.

Restorative control. This aims at re-energizing the system after an event which has led to partial or total blackout. Efficient restorative control is necessary in order to minimize outage costs. Clearly, strategies for restorative control need to be coordinated with emergency control schemes.

Post-mortem analysis. Generally, after a major blackout it is necessary to find out the main causes and to evaluate the outage costs incurred by the users of the transmission system. Post-mortem analysis will be easier if the appropriate information has been stored during the disturbance and is made readily available to the analysts.

Financial compensation. After a major event, it is generally the case that some users should be paid for the economic losses they have incurred due to the consequences of the outage on their business. The inputs to these decisions are the results of post-mortem analysis, contractual agreements (possibly) and the rules for compensation defined in the security standards.

Thus the decision making process is by itself rather intricate. On the other hand, the types of problems considered in power system security are essentially rare events, difficult to model, difficult to predict, but they may potentially have very important human, ecological and economic consequences.

Given the complexity of the overall security problem, at each one of these decision steps it is generally decomposed into a number of simpler subproblems. Given the fact that the problems are most often related to rare events, they generally require simulation techniques (tailored to the considered subproblem) to answer "what if" type of questions.

However, in spite of the uncertain character of power systems, the use of probabilistic approaches was in the past essentially restricted to long term planning applications, and even in this context their use has been quite marginal. In most applications, problems used to be solved by combining human expertise and systematic screening techniques. However, in the last few years, the probabilistic framework described in this paper has received a growing number of practical applications.

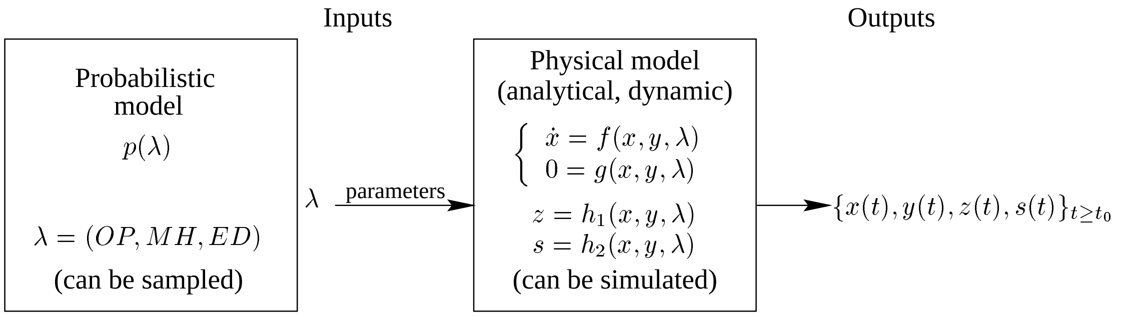


Figure 21: A priori information : probabilistic model and physical model

4.2.2 Computer experiments for power system analysis and design

The approach consists in two main steps : database generation using Monte-Carlo simulations of dynamic behavior scenarios, and database analysis using automatic learning [36, 37]. Note that the Monte-Carlo approach is well suited to the intrinsically probabilistic nature of the problem (think about the random nature of external disturbances, failures of protection devices, mis-tuned settings, etc.). On the other hand, automatic learning techniques are (by definition) designed so as to separate predictable information from the random component.

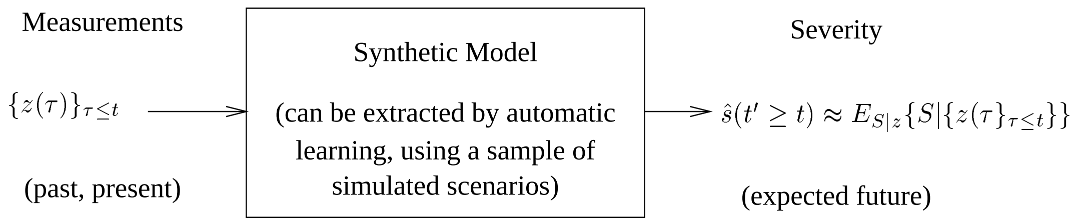
As sketched generically in figure 21, the Monte-Carlo simulations require two models : the probabilistic model (for random sampling), and the (dynamic) physical model (for numerical simulation).

The probabilistic model (see figure 21) represents a priori knowledge about initial operating points (OP), external disturbances and inputs (ED) and other modeling parameters used in a dynamic simulation (MH). Notice that this scheme allows one to represent probabilities of failures of protection devices, ranges of possible model parameters of external systems or aggregated load areas, etc. All this information is symbolized by a parameter vector (λ in figure 21) which defines a simulation scenario, and is fed later on into the physical model, in order to simulate the corresponding trajectory.

In our discussion below, we refer to the design of defense plans (i.e. the design of protection schemes against blackouts) in order to illustrate ideas. For a more detailed discussion of this problem, and of the actual study carried out on a real large scale system, we refer the interested reader to reference [37].

4.2.3 A priori models.

The design of the probabilistic model is generally the most difficult and at the same time the most important modeling step in this approach. The chosen probability distributions incorporate all prior knowledge as well as the definition of the range of conditions which the study aims to cover.



Measurements
$\{z(\tau)\}_{\tau\leq t}$
(past, present)
Synthetic Model
(can be extracted by automatic learning, using a sample of simulated scenarios)
Severity
$\hat{s}(t' \geq t) \approx E_{S|z}\{S|\{z(\tau\}_{\tau\leq t}\}\}$
(expected future)

Figure 22: Information extracted by AL : expected value of future severity given past measurements

As usually, the dynamic model is a set of differential and algebraic equations which define the analytical relationships among states, parameters, time, measurements (denoted by $z(t)$ in figure 21) and scenario severity indicators (denoted by $s(t)$ in figure 21).

Since there is a large variety of problems in power systems which may be solved by the discussed approach, the a priori models (both probabilistic and physical) will actually depend strongly on the particular application under consideration.

For example in the context of defense plan design, both models would be rather detailed [37]. Measurements would be all those variables which can be used as input to emergency control schemes, whereas indicators would denote the information which would be observed about the severity of the scenario under different hypotheses of control actions.

4.2.4 Database generation

In order to reduce the number of required simulation scenarios, the Monte-Carlo sampling may artificially increase the probabilities of various types of failures, and sample only combinations of severe disturbances. In other words, the probabilistic model may be biased in order to sample predominantly those regions in the measurement space where the variance of the severity indicators is high. In the literature on Monte-Carlo simulations there are several well known variance reduction techniques which may be used for that purpose (see e.g. [4], and also the literature on optimal experiment design [38]).

However, the coupling of such techniques with automatic learning is more intricate and there is still much work to be done in this field (see also the literature on query based learning and reinforcement learning).

For the time being we used the approach by biasing the probability distributions in an ad hoc way, so as to concentrate the simulations in the regions where prior expertise tells us that most information can be gained. Note that without biasing the random sampling it would be necessary to generate huge amounts of scenarios in order to gather some interesting blackout situations. However, while biased probabilities were used for the random sampling, track of the “actual” probabilities may be kept while generating the database so as to enable correct interpretation of the results.

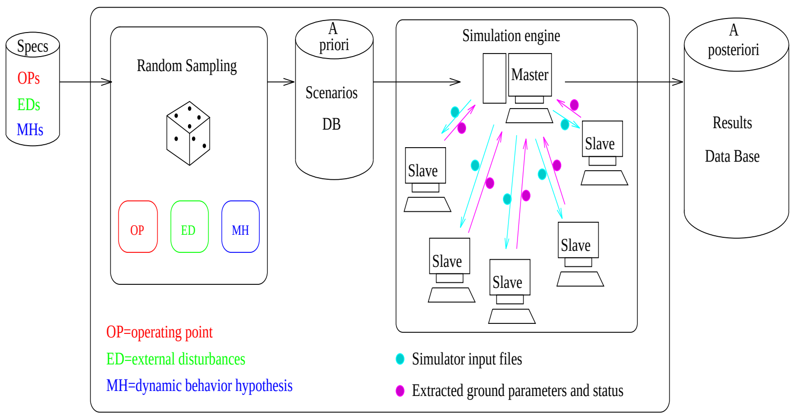


Figure 23: Parallel simulation of scenarios (taken from [21])

4.2.5 Numerical simulation tools

In principle the approach can be used with any numerical simulation tool deemed sufficiently accurate for the problem under consideration.

Note however that in the context of defense plan design, a rather detailed dynamic model should be used, able to simulate both slow and fast dynamics and various protective devices, so as to assess the performance of the system with good accuracy.

Thus, the database generation generally calls for parallel computations in order to be able to carry out several thousands of dynamic simulations with acceptable response times. In the study on the EHV system of Electricité de France described in [37], 12 CPUs were thus used in parallel for the simulations, using a master slave architecture depicted in figure 23.

4.2.6 Flexibility w.r.t. prior assumptions

Note that in traditional practice, in order to cope with the overall complexity of the problem, the experts use a divide-and-conquer approach, where the problem is **a priori** decomposed into simpler subproblems on a geographical and/or phenomena-wise base. If the system is undergoing changes this may be misleading, since expertise becomes more quickly obsolete and the “chance” of missing some potential risks is increased.

In the proposed approach, the principle is quite different. The problem is addressed a priori in a global way : the Monte-Carlo probabilistic model is designed in order to represent all reasonably possible causes of collapse (together with their relative probability) and the dynamic simulation model is designed so as to allow the study of both slow and fast phenomena. The geographical and phenomena-wise decomposition is carried out **a posteriori** by looking at the database of simulation results with the help of automatic learning. This allows one in principle to study interactions among various phenomena, if they happen, and to identify the most likely consequences in a more objective way.

However, during the database generation it is not necessary to restrict the amount of information which will be stored and available later for analysis. Actually, it is advisable to keep trace of all variables which could be used either as real-time measurements (inputs to emergency control device triggering criteria) or in order to define the scenario severity. The data mining tools offer the possibility to combine these variables in a very flexible way in order to build synthetic models. Note also that input and output variables may be either numerical continuous (analog states and measurements) or discrete events (fault occurence, relay tripping, breaker opering/reclosure . . .).

4.2.7 Extraction of knowledge for practical applications

Engineers trying to solve power system related problems should look at the whole toolbox of data mining methods, and not hesitate to combine different techniques to yield a full, practical solution. Data selection step may be performed with a decision tree, a clustering approach, or a correlation analysis and later on, the result may serve as input for other supervised techniques, possibly with the problem decomposed into simpler subproblems.

In order to show the interest of extracting synthetic information from a database of simulation scenarios, let us continue with our example of defense plan design.

The objective of a defense plan is to trigger control actions in real-time so as to avoid large disturbances to spread throughout the power system and to result in total blackout. The control actions used by defense plans are generally heroic, and most often they result in disconnecting generators and customers for a certain duration in order to bring the system back in a stable operation. Therefore, defense plans should be designed in such a way that they will act if and only if required, and it is of paramount importance to design effective triggering rules, using appropriate real-time information.

In order to design a triggering rule, it is necessary in some way to predict the future values of the scenario severity s, given present and past values of measurements z. In general, s and y are random variables (actually random processes). Thus, at some time t, a synthetic model is used to predict s at some future time, $t' \geq t$, using the already observed values of measurements $\{z(\tau)\}_{\tau \leq t}$. This is suggested in figure 22, where the estimate is provided by the conditional expectation of this random variable given the already observed values of z (notation $\hat{s}(t') \approx E_{S|z}\{S|\{z(\tau\}_{\tau \leq t}\}$).

Such a prediction model provides normally only an approximation of this conditional expectation, which, because it can not be computed analytically in practice, must be estimated from a random sample of input/output pairs. Thus, as it is suggested in figure 22, the design of such prediction models is carried out in the presented approach applying automatic learning to random samples generated by Monte-Carlo simulations. As we have already mentioned, this requires, for each variable which has to be predicted, the proper identification of relevant measurements (those parts of y which carry indeed information about the future value of z) and the design of a synthetic model which will estimate severity as precisely as possible.

Notice that in practical applications both z and s are vectors of variables which combine discrete (breaker status, relay trip, etc.) and numerical information (voltage magnitudes, amount of load shed, etc.).

In order to exploit a database such as the one used for the design of defense plans [37], several automatic learning techniques can be used :

- correlation analysis, in order to reduce the dimensionality of the input vectors (in the study described in [37], the number of candidate (temporal) input variables was more than 800, yielding a database of about 2 Gigabytes);
- clustering analysis, in order to find out the main modes of system breakdown (in the study described in [37], about 40 modes were identified, by repetitive applications of the K-means method);
- supervised learning in order to build models able to predict various modes of blackout from real-time measurements (in the study described in [37], temporal decision trees were grown in order to detect early enough voltage collapse in one part of the system).

4.3 Main contributions of the approach

The automatic learning based approach, combined with Monte-Carlo simulations, has been applied during last ten years to a number of large scale power system problems. These applications first started in the context of research collaborations between the University and some electric power utilities. Since a few years now, the approach is used by Electricité de France in order to carry out various operational studies.

Below we start by briefly enumerating the type of studies which have been carried out, providing pointers to related publications. Then we will discuss the main complementary features of this approach with respect to other techniques used in the context of power systems analysis, design and control.

4.3.1 Sample of large scale applications

Transient stability. The first large scale application was carried out in the context of transient stability assessment of a large nuclear power plant of Electricité de France [39]. This was followed by a similar study to determine transient stability limits of the James' Bay corridor of the Hydro-Québec system [40].

Neither of these early research projects has resulted in an actual implementation in the control center. However, they contributed to demonstrate the usefulness of the overall methodology, to develop software tools and bring the methodology to a sufficiently mature stage.

Voltage stability. Applications to voltage stability have been carried out mainly in collaboration with Electricité de France. The first study was carried out on the Western part of this system which experienced voltage collapse in the past. Most of the details of this research can be found in [41].

After this initial collaboration, Electricité de France started to use the method in various studies, and in particular in order to define new operating rules [42] for the South-Eastern part of their system : these new rules are in use in the control center since early 1998. In one year, they allowed Electricité de France to reduce operating costs by about 2 million dollars.

Emergency control design. The early work in emergency control focused on voltage collapse problems [43]. Later on, a long term collaboration between Electricité de France and the University of Liège started in order to apply the methodology to study power system blackout modes and enhance the overall emergency control system [37]. More recently, Electricité de France has used the approach to improve some of their emergency control schemes [44].

Under research. Presently there are other topics under investigation, in particular the design of global security limits for the Belgian power system, in collaboration with Electrabel, and the analysis of operating costs structures, in collaboration with Electricité de France.

4.3.2 Main complementary features

Numerical simulation tools provide very detailed information for each one of the simulated scenarios. Up to now, power system engineers used these tools in a very manual fashion. They had to build up by hand each simulation scenario, run the simulation tool, then look at the results in order to see how the system behaves in this particular simulation, change the scenario specification and input files and run the simulatore again to see the effect.

This iterative process has essentially the limitations of human experts : the number of simulations which can be run and exploited is quite limited, essentially by the human capabilities to analyse the results and synthesize them in a synthetic manner so as to define decision making criteria. Also, in this process the conditions under which the system is simulated are not defined explicitly : they depend on the successive steps and quite strongly on the expertise of the engineer in charge of the study. To make the whole process tractable, the engineers generally decompose a large problem into simpler subproblems according to the prior knowledge about the problem.

Today, power system behaviors become more and more complex. Expertise is thus becoming more quickly obsolete, and may very easily become misleading. On the other hand, the complexification of the systems makes their propper design and control even more necessary than in the past. However, the classical way of conducting studies does not allow to increase significantly the number of simulations which may be carried out.

The approach presented in this paper makes studies more systematic, and thus provides an appropriate way to enhance the decision making process. It can easily scale up to very large numbers of simulations, if required. Indeed, most of the laborious tasks are now carried out by the computer, or in parallel by several computers, if necessary. More importantly, the approach obliges the engineers to model the probabilistic nature of their problem in a very explicit fashion and to define precisely the scope of a study. Therefore, the results obtained are more transparent, can be easily reproduced and updated if necessary. They require also less strong prior assumptions and are therefore liable to provide more objective decision criteria. At the same time, using appropiate techniques to extract infromation from the simulation scenarios, the approach makes it possible to enhance human expertise.

Finally, the approach is intrinsically a probabilistic approach. It is therefore able to take into account the numerous uncertainties which make power system operation and control so difficult.

5 Conclusions

This paper has presented a new approach to use computer experiments to study complex systems. This approach combines probabilistic modeling and reasoning, with classical deterministic simulation techniques in order to analyse, design and control complex systems. The core of the approach is provided by a toolbox of automatic learning methods which are used to extract information from databases generated by Monte-Carlo simulations.

The approach has been developed in the context of electric power system applications during the last 15 years. In this field it has already provided practical solutions to various difficult problems.

Besides describing the main principles of the approach, one of the aims of this paper was to suggest that this approach could be applied to many problems in the general context of complex systems. Indeed, many other complex systems built by engineers lead to similar problems. For example, the design, operation and control of telecommunication networks, computer networks, chemical and mechanical plants, could take advantage of the combined use of simulation, Monte-Carlo sampling, and automatic learning. We hope that this chapter will contribute to raise interest in this methodology by experts of these application fields.

Nevertheless, although the approach has already evolved significantly during the last few years, there is still a lot of room for research. In particular, from the theoretical viewpoint we mention the combination of optimal experiment design ideas with automatic learning and the very broad area related to learning from temporal databases. From the practical point of view, we believe that one of the main outcomes of the approach will be to allow the use of probabilistic techniques in the context of non-linear complex system analysis, design and control.

References

[1] T. E. Dy Liacco. *Control of Power Systems via the Multi-Level Concept*. PhD thesis, Sys. Res. Center, Case Western Reserve Univ., 1968. Rep. SRC-68-19.

[2] T. E. DyLiacco. The adaptive reliability control system. *IEEE Trans. on Power Apparatus and Systems*, PAS-86(5):517–531, May 1967.

[3] E. T. Jaynes. *Probability Theory: the Logic of Science*. edited by E. T. Jaynes, available from http://omega.albany.edu:8008/JaynesBook.html, July 1995. Fragmentary edition.

[4] R. Y. Rubinstein. *Simulation and the Monte-Carlo Method*. J. Wiley, 1981.

[5] R. Y. Rubinstein and B. Melamed. *Modern simulation and modeling*. Wiley Series in probability and statistics, 1998.

[6] P. A. Chou. Optimal partitioning for classification and regression trees. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, PAMI-13(14):340–354, 1991.

[7] U. M. Fayyad, G. Piatetsky-Shapiro, P. Smyth, and R. Uthurusamy. *Advances in Knowledge Discovery and Data Mining*. AAAI Press/MIT Press, 1996.

[8] A. A. Freitas and S. H. Lavington. *Mining very large databases with parallel processing*. Kluwer Academic, 1998.

[9] V. N. Vapnik. *The nature of statistical learning theory*. Springer Verlag, 1995.

[10] M. Vidyasagar. *A theory of learning and generalization*. Springer, 1997.

[11] V. N. Vapnik. *Statistical learning theory*. Adaptive and learning systems for signal processing, communications and control. Wiley, 1998.

[12] J. Rissanen. Modeling by shortest data description. *Automatica*, 14:465–471, 1978.

[13] J. R. Quinlan and R.L. Rivest. Inferring decision trees using the minimum description length principle. *Information and Computation*, 80:227–248, 1989.

[14] L. Wehenkel. A probabilistic framework for the induction of decision trees. Technical report, University of Liège, 1992.

[15] C. S. Wallace and J. D. Patrick. Coding decision trees. *Machine Learning*, 11:7–22, 1993.

[16] L. Wehenkel. Decision tree pruning using an additive information quality measure. In B. Bouchon-Meunier, L. Valverde, and R.R. Yager, editors, *Uncertainty in Intelligent Systems*, pages 397–411. Elsevier - North Holland, 1993.

[17] M. Mehta, J. Rissanen, and R. Agrawal. Mdl-based decision tree pruning. In *Proc. of KDD-95*, Montréal, 1995.

[18] J. H. Friedman. Greedy function approximation: a gradient boosting machine. Technical report, Stanford University, 1999.

[19] L. Breiman. Arcing classifiers. Technical report, University of California at Berkeley, 1996.

[20] L. Breiman. Bagging predictors. *Machine learning*, 26(2):123–140, 1996.

[21] L. Wehenkel. *Automatic learning techniques in power systems*. Kluwer Academic, Boston, 1998.

[22] L. Breiman, J. H. Friedman, R. A. Olshen, and C. J. Stone. *Classification and Regression Trees*. Wadsworth International (California), 1984.

[23] J. R. Quinlan. *C4.5. Programs for Machine Learning*. Morgan Kaufman, 1993.

[24] X. Boyen and L. Wehenkel. Automatic induction of fuzzy decision trees and its application to power system security assessment. *Fuzzy sets and Systems*, 102:3–19, 1999.

[25] P. Geurts and L. Wehenkel. Early prediction of electric power system blackouts by temporal machine learning. In *Proc. of ECML-AAAI'98 Workshop on "AI Approaches to Time-series Analysis"*, Madison (Wisconsin), 1998.

[26] J. H. Friedman and W. Stuetzle. Projection pursuit regression. *Jour. of the Am. Stat. Ass.*, 76(376):817–823, December 1981.

[27] E. F. Sànchez-Ùbeda. *Models for data analysis: contributions to automatic learning*. PhD thesis, Universidad Pontificia Comillas, 1999.

[28] E. Sánchez-Úbeda and L. Wehenkel. The hinges model: A one-dimensional continuous piecewise polynomial model. In *Proc. of IPMU'98, Information Processing and Management of Uncertainty in Knowledge-Based Systems*, Paris, 1998.

[29] P. Cheeseman, M. Self, J. Kelly, and J. Stutz. Bayesian classification. In *Proc. of the 7th AAAI Conf.*, pages 607–611, 1988.

[30] G. Das, D. Gunopulos, and H. Mannila. Finding similar time series. In *Proc. of KDD'97*, 1997.

[31] H. Mannila, H. Toivonen, and A. I. Verkamo. Discovery of frequent episodes in event sequences. Technical Report C-1997-15, University of Helsinki, Department of Computer Science, 1997.

[32] R. Agrawal, C. Faloutsos, and A. Swami. Efficient similarity search in sequence databases. In *Proc. of 4th Int'l Conf. on foundations of data organization and algorithms*, Chicago, 1993.

[33] T. S Dillon. Artificial neural network applications to power systems and their relationship to symbolic methods. *Int. J. of Elec. Power and Energy Syst.*, 13(2):66–72, April 1991.

[34] A. Breipohl, F. N. Lee, J. Huang, and Q. Feng. Sample size reduction in stochastic production simulation. *IEEE Trans. on Power Syst.*, 5(3):984–992, August 1990.

[35] M. V. F. Pereira, M. E. P. Maceira, G. C. Oliveira, and L. M. V. G. Pinto. Combining analytical models and monte-carlo techniques in probabilistic power system analysis. *IEEE Trans. on Power Syst.*, PWRS-7:265–272, 1992.

[36] L. Wehenkel and M. Pavella. Decision trees and transient stability of electric power systems. *Automatica*, 27(1):115–134, 1991.

[37] L. Wehenkel, C. Lebrevelec, M. Trotignon, and J. Batut. Probabilistic design of power-system special stability controls. *Control Engineering Practice*, 7(2):183–194, 1999.

[38] J. R. Koehler and A. B. Owen. Computer experiments. Technical report, Standford University - Department of Statistics, 1995. http://playfair.Stanford.EDU/reports/owen/.

[39] L. Wehenkel, M. Pavella, E. Euxibie, and B. Heilbronn. Decision tree based transient stability method - a case study. *IEEE Trans. on Power Syst.*, 9(1):459–469, 1994.

[40] L. Wehenkel, I. Houben, M. Pavella, L. Riverin, and G. Versailles. Automatic learning approaches for on-line transient stability preventive control of the Hydro-Québec system - Part I. Decision tree approaches. In *Proc. of SIPOWER'95, 2nd IFAC Symp. on Control of Power Plants and Power Systems*, pages 231–236, December 1995.

[41] L. Wehenkel and Y. Jacquemart. *Automatic Learning Methods - Application to Dynamic Security Assessment*. Tutiral given at PICA'97, 1997.

[42] P. Cholley, C. Lebrevelec, S. Vitet, and M. de Pasquale. A statistical approach to assess voltage stability limits. In *Proc. of Bulk Power Syst. Dynamics and Contr. IV - Restructuring*, Santorini, Greece, 1998.

[43] T. Van Cutsem, L. Wehenkel, M. Pavella, B. Heilbronn, and M. Goubin. Decision trees for detecting emergency voltage conditions. In *Proc. of the 2nd Int. NSF Workshop on Bulk Power System Voltage Phenomena - Voltage Stability and Security, Deep Creek Lake, Ma*, pages 229–240, August 1991.

[44] C. Lebrevelec, P. Cholley, J.F. Quenet, and L. Wehenkel. A statistical analysis of the impact on security of a protection scheme on the French power system. In *Proc. of PowerCon 98*, Beijing, Aug 1998.

[45] P. Billingsley. *Probability and measure*. John Wiley and Sons, 1979.

Figure and table legends

Table 1. Spreadsheet view of a small part of the OMIB database

Figure 1. Overall probabilistic framework based on automatic learning.

Figure 2. Use of Monte-Carlo simulations instead of analytical derivations

Figure 3. Simple one-machine infinite-bus system (OMIB)

Figure 4. Illustration of OMIB database content

Figure 5. First 5000 random scenarios of the OMIB database (adapted from [21])

Figure 6. Illustration of bias and variance

Figure 7. Hypothetical decision tree and equivalent if-then rules (taken from [21])

Figure 8. Three first steps of decision tree growing (taken from [21])

Figure 9. "Orthogonal" decision tree (end result) (taken from [21])

Figure 10. "Oblique" decision tree and classification boundaries (adapted from [21])

Figure 11. Regression tree (CCT function of $\delta(t)$)

Figure 12. Structure selection by cross-validation

Figure 13. Perceptron (neuron) and multilayer perceptron (adapted from [21])

Figure 14. Single hidden layer perceptrons (adapted from [21])

Figure 15. Distribution of MLP approximation errors (test set) (taken from [21])

Figure 16. 3000 learning states and nearest neighbors of state 4984 (taken from [21])

Figure 17. Distribution of 1NN approximation errors (test set) (taken from [21])

Figure 18. Dendrogram : correlation analysis of OMIB database attributes

Figure 19. Clustering of OMIB scenarios according to temporal behavior of rotor angle

Figure 20. Dendrograms of K-means results

Figure 21. A priori information : probabilistic model and physical model

Figure 22. Information extracted by AL : expected value of future severity given past measurements

Figure 23. Parallel simulation of scenarios (taken from [21])

Footnotes

[1]Often one uses the term "state of nature" to denote this concept. We prefer to use the term behavior, in order to avoid confusion with the classical notion of state used in system theory.

[2]This completeness assumption is fully legitimate in the case of finite universes.

[3]which verifies the axioms of a probability measure [45].

[4]We suppose that the measurement devices are part of the studied system.

[5]CPU times on a SUN UltraSparc 300MHz workstation.

[6]The equidistant region is slightly oval due to the fact that we have normalized Pu and Qu by their standard deviation before computing the distance.