Multi-domain performance analysis with scores tailored to user preferences

Sébastien Piérard, Adrien Deliège, and Marc Van Droogenbroeck *

Montefiore Institute, University of Liège, Liège, Belgium

Abstract.   The performance of algorithms, methods, and models tends to depend heavily on the distribution of cases on which they are applied, this distribution being specific to the applicative domain. After performing an evaluation in several domains, it is highly informative to compute a (weighted) mean performance and, as shown in this paper, to scrutinize what happens during this averaging. To achieve this goal, we adopt a probabilistic framework and consider a performance as a probability measure (*e.g.*, a normalized confusion matrix for a classification task). It appears that the corresponding weighted mean is known to be the summarization, and that only some remarkable scores assign to the summarized performance a value equal to a weighted arithmetic mean of the values assigned to the domain-specific performances. These scores include the family of ranking scores, a continuum parameterized by user preferences, and that the weights to consider in the arithmetic mean depend on the user preferences. Based on this, we rigorously define four domains, named easiest, most difficult, preponderant, and bottleneck domains, as functions of user preferences. After establishing the theory in a general setting, regardless of the task, we develop new visual tools for two-class classification.

1   Introduction

The development of algorithms, methods, and models is often complicated by two types of uncertainty. First, the *applicative domain*, and thus the distribution of inputs, is rarely unique and precisely known during development. Second, *user preferences* regarding tradeoffs affecting the achievable performance (*e.g.*, to what degree do they prefer false negatives to false positives in classification?) are also generally not precisely known and can differ from one user to another.

Developers can, of course, implement various techniques (continual learning, test-time adaptation, domain adaptation [1], domain generalization [2], . . .) to make their *entity* —the term used in this paper for algorithm, method, or model— usable in various domains, but the achievable performance still varies from one domain to another. After evaluating the developed entity in several domains, the developer is left with many questions, such as: Which domains are the strengths and weaknesses of the developed entities? Which domains have the greatest influence on average performance? Which domains constitute the bottlenecks for the average performance improvement? We need to address these questions from the end-user preferences perspective.

The objective of this paper is to rigorously demonstrate that, no matter what the task is, by taking an adequate way of averaging the domain-specific performances (*i.e.*, the *summarization* introduced in [3]) as well as an adequate family of scores (*i.e.*, the *ranking scores* introduced in [4]), the *easiest domain*, *most difficult domain*, *preponderant domain*, and *bottleneck domain* become all well-defined and relative to some easy to interpret user preferences.

Our contributions are threefold:

1. Theoretical contribution on scores. We highlight the fact that the *expected value scores* (including all *unconditional probabilistic scores*) and the *expected value ratio scores* (including all *probabilistic conditional scores* and all *ranking scores*) have the remarkable property that the value taken for the averaged performance obtained by the *summarization* technique [3] is equal to some weighted arithmetic mean of the values taken for the domain-specific performances, and provide closed-form formulas to compute these weights.

2. Contribution on user-centered multi-domain performance analysis. Capitalizing on the remarkable property and focusing on the family of *ranking scores* [4], we rigorously define the *easiest*, *most difficult*, *preponderant*, and *bottleneck* domains as functions of a random variable that can be used to encode some user preferences for any task.

3. Contribution on visualization tools. In the case of two-class crisp classification, we introduce new variants of the Tile, called *flavors* [5, 6], that allow us to visualize in a square how the *easiest*, *most difficult*, *preponderant*, and *bottleneck* domains depend on the user preferences.

Notations. Following [3, 4, 6], we consider a probabilistic framework. A performance is a probability measure (*e.g.*, a normalized confusion matrix for a classification task) on a measurable space (Ω, Σ), where Ω is the sample space and Σ the event space. The set of all performances is denoted by $\mathbb{P}_{(\Omega,\Sigma)}$. We denote the set of domains by $\mathbb{D}$, the performance in the domain d by P_d, and the weight arbitrarily given to the domain d by λ_d.

2 Averaging performances with the summarization

In this paper, we build upon the summarization [3], which is a probabilistic performance averaging technique that preserves the relationships between scores and preserves the probabilistic meaning of both unconditional and conditional probabilistic scores. It is applicable regardless of the task. The summarized performance $\overline{P}$ is the following mixture of performances:

$$\overline{P} = \frac{\sum_{d \in \mathbb{D}} \lambda_d P_d}{\sum_{d \in \mathbb{D}} \lambda_d} \qquad \text{with} \qquad \lambda_d \geq 0 \, \forall d \in \mathbb{D} \,. \tag{1}$$

As noticed in [3], for some remarkable scores X, the summarization leads to

$$X(\overline{P}) = \sum_{d \in \mathbb{D}} \omega_{X,d} X(P_d) \qquad \text{with} \qquad \begin{cases} \omega_{X,d} \geq 0 \, \forall d \in \mathbb{D} \\ \sum_{d \in \mathbb{D}} \omega_{X,d} = 1 \end{cases} \,. \tag{2}$$

We call λ_d the *domain weight* and $\omega_{X,d}$ the *summarization weight*.

Formulas for the weights for expected value scores. We parameterize these scores by a random variable V and define them as $X_V^{EV} : \mathbb{P}_{(\Omega,\Sigma)} \to \mathbb{R} : P \mapsto X_V^{EV}(P) = \mathbf{E}_P[V]$, where $\mathbf{E}$ is the mathematical expectation. With these scores, eq. (2) holds and the summarization weights do not depend on the performance:

$$\omega_{X_V^{EV},d} = \frac{\lambda_d}{\sum_{d\in\mathbb{D}} \lambda_d}. \tag{3}$$

Formulas for the weights for expected value ratio scores. We parameterize these scores by two random variables, V_1 and $V_2 \neq 0$, and define them as $X_{V_1,V_2}^{EVR} : \mathrm{dom}(X_{V_1,V_2}^{EVR}) \to \mathbb{R} : P \mapsto X_{V_1,V_2}^{EVR}(P) = \frac{\mathbf{E}_P[V_1]}{\mathbf{E}_P[V_2]}$, where $\mathrm{dom}(X_{V_1,V_2}^{EVR}) = \{P \in \mathbb{P}_{(\Omega,\Sigma)} : \mathbf{E}_P[V_2] \neq 0\}$. With these scores, eq. (2) also holds, but contrary to the expected value scores, the summarization weights depend on the performance:

$$\omega_{X_{V_1,V_2}^{EVR},d} = \frac{\lambda_d X_{V_2}^{EV}(P_d)}{\sum_{d\in\mathbb{D}} \lambda_d X_{V_2}^{EV}(P_d)} = \frac{\lambda_d}{\sum_{d\in\mathbb{D}} \lambda_d} \frac{X_{V_2}^{EV}(P_d)}{X_{V_2}^{EV}(\overline{P})}. \tag{4}$$

Equations (3) and (4) generalize the ones given in [3], in the particular case of *unconditional* and *conditional* probabilistic scores, respectively.

3 Taking user preferences into account through scores

In addition, we also build upon the ranking scores [4], which establish a connection between the performances P, the task (modeled through a random variable S called *satisfaction*), and some user preferences (modeled through a nonnegative random variable I called *importance*). These scores induce meaningful performance orderings [4], in the sense that these orderings satisfy the fundamental axioms of performance-based ranking. They are defined as:

$$R_I : \mathrm{dom}(R_I) \to \mathbb{R} : P \mapsto R_I(P) = \frac{\mathbf{E}_P[IS]}{\mathbf{E}_P[I]}, \tag{5}$$

where $\mathrm{dom}(R_I) = \{P \in \mathbb{P}_{(\Omega,\Sigma)} : \mathbf{E}_P[I] \neq 0\}$. As $R_I = X_{IS,I}^{EVR}$, all ranking scores are particular cases of expected value ratio scores and eq. (4) also holds for them:

$$\omega_{R_I,d} = \frac{\lambda_d X_I^{EV}(P_d)}{\sum_{d\in\mathbb{D}} \lambda_d X_I^{EV}(P_d)} = \frac{\lambda_d \mathbf{E}_{P_d}[I]}{\sum_{d\in\mathbb{D}} \lambda_d \mathbf{E}_{P_d}[I]} = \frac{\lambda_d}{\sum_{d\in\mathbb{D}} \lambda_d} \frac{\mathbf{E}_{P_d}[I]}{\mathbf{E}_{\overline{P}}[I]}. \tag{6}$$

Thanks to these scores, we can now analyze, in the perspective of user preferences, the performance of an entity that has been evaluated in several domains.

For a given entity, we can rank the domains, from the easiest (with the best performance) to the most difficult (with the worst performance). This ranking depends on the user preferences as follows:

$$\text{easiest-domain}(I) = \arg\max_{d\in\mathbb{D}} R_I(P_d), \tag{7}$$

$$\text{most-difficult-domain}(I) = \underset{d\in\mathbb{D}}{\arg\min}\, R_I(P_d)\,. \tag{8}$$

We can also determine the preponderant domain *w.r.t.* user preferences. It is the domain for which the summarization weight is the largest:

$$\text{preponderant-domain}(I) = \underset{d\in\mathbb{D}}{\arg\max}\, \frac{\lambda_d \mathbf{E}_{P_d}[I]}{\sum_{d\in\mathbb{D}} \lambda_d \mathbf{E}_{P_d}[I]} = \underset{d\in\mathbb{D}}{\arg\max}\, \lambda_d \mathbf{E}_{P_d}[I]\,. \tag{9}$$

A common practice in software engineering is to iteratively identify the parts that are bottlenecks and improve these parts. Similarly, if the objective is to design an entity that performs the best on average, one must be able to identify the domains that are bottlenecks. These are the domains for which the summarization weights are high and the values taken by the ranking scores are weak. We unify these two—sometimes contradictory—objectives by domain ablation: we define the bottleneck domain as the one that has to be removed from the set of domains to maximize a given ranking score:

$$\text{bottleneck-domain}(I) = \underset{d\in\mathbb{D}}{\arg\max}\, R_I\left(\frac{\sum_{d'\in\mathbb{D}\setminus\{d\}} \lambda_{d'} P_{d'}}{\sum_{d'\in\mathbb{D}\setminus\{d\}} \lambda_{d'}} \right)\,. \tag{10}$$

4 Visualization tools for two-class crisp classification

Lastly, we build upon the Tile [5], which is a tool for two-class crisp classification to visualize functions of the importance I on a square space. The horizontal axis indicates the relative importance considered by the user for the *true positives w.r.t.* the *true negatives*. The vertical axis indicates the relative importance for the *false negatives w.r.t.* the *false positives*.

Defining the task. From the performance perspective [4], two-class crisp classification is defined by the sample space $\Omega = \{tn, fp, fn, tp\}$ and the satisfaction $S = \mathbf{1}_{\{tn,tp\}}$. The samples tn, fp, fn, and tp are traditionally interpreted as a true negative, false positive, false negative, and true positive, respectively.

Setting the user preferences. In our setting, the user can freely specify some preferences through the importance values $I(tn)$, $I(fp)$, $I(fn)$, and $I(tp)$. The generic formula for the ranking scores given in eq. (5) is particularized as follows:

$$R_I(P) = \frac{P(\{tn\})\, I(tn) + P(\{tp\})\, I(tp)}{P(\{tn\})\, I(tn) + P(\{fp\})\, I(fp) + P(\{fn\})\, I(fn) + P(\{tp\})\, I(tp)}\,.$$

Using the Tile. As the ranking scores sharing the same values for $a(I) = \frac{I(tp)}{I(tn)+I(tp)}$ and $b(I) = \frac{I(fn)}{I(fp)+I(fn)}$ induce the same ranking, it has been proposed in [5] to focus on the *canonical ranking scores* such that $I(tn) + I(tp) = I(fp) + I(fn)$ and to map them on the $(a,b) \in [0,1]^2$ space called *Tile*. So, $I(tn) = 1-a$, $I(fp) = 1-b$, $I(fn) = b$, and $I(tp) = a$. Note that the accuracy, the true negative rate, the true positive rate, the negative predictive value, the positive predictive value, and F_1 are particular cases of canonical ranking scores.
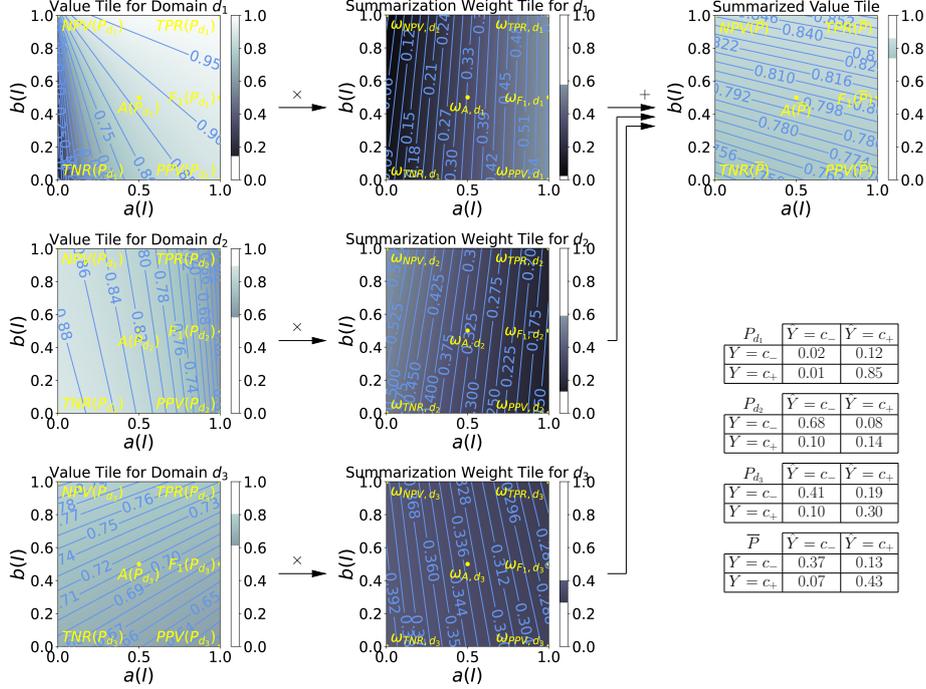
Fig. 1: Tiles showing the summarization on 3 domains: by multiplying each domain-specific *Value Tile* by the corresponding *Summarization Weight Tile* and adding the results together, one obtains the *Summarized Value Tile*, *i.e.* the *Value Tile* for $\overline{P}$. This Tile is exactly the same as the *Value Tile* that would be obtained from $\overline{P}$ after computing it with eq. (1). However, by scrutinizing what happens during the performance averaging, it becomes clear that the actual weights to consider strongly depend on the user preferences (*i.e.*, point in the Tile), which is something hidden in eq. (1).

Visualizing the summarization. Figure 1 depicts the summarization, *i.e.* eq. (1), with Tiles. The values taken by the canonical ranking scores and the weights $\omega_{R_I,d}$ are shown, respectively, on *Value Tiles* and *Summarization Weight Tiles*.

Introducing new flavors. We extend the catalog of flavors (*i.e.*, functions of the importance I) compiled in [6] and implemented in the SORBETTO library [7], and propose to depict on the Tile the easiest, the most difficult, the preponderant, and the bottleneck domains (see fig. 2). Practically, to improve on the worst case scenario, the developer should improve on the most difficult domain, and to improve on average, improve on the bottleneck domain. Currently, only the central point is generally considered (*i.e.* the accuracy), such as in [2], overlooking entirely the dimension of user preferences.

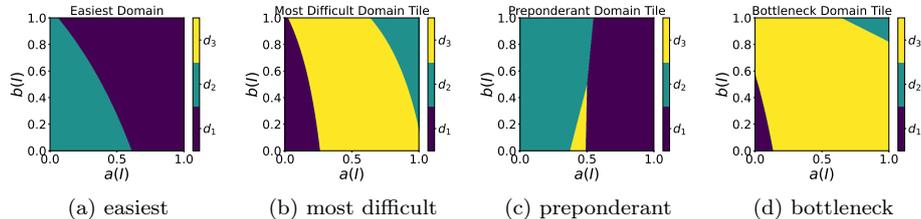|			
(a) easiest	(b) most difficult	(c) preponderant	(d) bottleneck

Fig. 2: In this work, we propose four new flavors for the Tile, as visual tools to perform multi-domain performance analyses *w.r.t.* user preferences. These are the Tiles obtained for the example of fig. 1. Notably, this analysis covers the two sources of uncertainty met in a model development: the domain dimension (the colors on the Tiles), and the user preferences (the position on the Tiles).

5 Conclusion

By considering the technique for averaging domain-specific performances known as the *summarization* [3] and the family of scores known as the *ranking scores* [4], we rigorously defined the *easiest, most difficult, preponderant*, and *bottleneck* domains as functions of some user preferences. First, we did it in a way that makes it possible for our theoretical results to be used for any task. Then, we particularized our results to the task of two-class crisp classification, demonstrating how the dependence of the four defined domains regarding the user preferences can be visualized on a square space that is known as the Tile [5]. By doing so, we extended the catalog of Tile flavors compiled in [6] and currently implemented in the SORBETTO library [7].

References

[1] G. Wilson and D. J. Cook, "A survey of unsupervised deep domain adaptation," *ACM Trans. Intell. Syst. Technol.*, vol. 11, pp. 1–46, Jul. 2020.

[2] I. Gulrajani and D. Lopez-Paz, "In search of lost domain generalization," *arXiv*, vol. abs/2007.01434, 2020.

[3] S. Piérard and M. Van Droogenbroeck, "Summarizing the performances of a background subtraction algorithm measured on several videos," in *IEEE Int. Conf. Image Process. (ICIP)*, (Abu Dhabi, United Arab Emirates), pp. 3234–3238, Oct. 2020.

[4] S. Piérard, A. Halin, A. Cioppa, A. Deliège, and M. Van Droogenbroeck, "Foundations of the theory of performance-based ranking," in *IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, (Nashville, TN, USA), pp. 14293–14302, IEEE, Jun. 2025.

[5] S. Piérard, A. Halin, A. Cioppa, A. Deliège, and M. Van Droogenbroeck, "The Tile: A 2D map of ranking scores for two-class classification," *arXiv*, vol. abs/2412.04309, 2024.

[6] A. Halin, S. Piérard, A. Cioppa, and M. Van Droogenbroeck, "A hitchhiker's guide to understanding performances of two-class classifiers," *arXiv*, vol. abs/2412.04377, 2024.

[7] S. Piérard, A. Halin, F. Marelli, S. Pernas, and J. Pierre, "Sorbetto: a Python library for producing classification tiles," 2025. https://github.com/uliege-performance/sorbetto, DOI: 10.5281/zenodo.17591788.