Agnostic Bayes

Thèse

Alexandre Lacoste

Doctorat en informatique
Philosophiæ doctor (Ph.D.)

Québec, Canada

Résumé

L'apprentissage automatique correspond à la science de l'apprentissage à partir d'exemples. Des algorithmes basés sur cette approche sont aujourd'hui omniprésents. Bien qu'il y ait eu un progrès significatif, ce domaine présente des défis importants. Par exemple, simplement sélectionner la fonction qui correspond le mieux aux données observées n'offre aucune garantie statistiques sur les exemples qui n'ont pas encore été observées. Quelques théories sur l'apprentissage automatique offrent des façons d'aborder ce problème. Parmi ceux-ci, nous présentons la modélisation bayésienne de l'apprentissage automatique et l'approche PAC-bayésienne pour l'apprentissage automatique dans une vue unifiée pour mettre en évidence d'importantes similarités. Le résultat de cette analyse suggère que de considérer les réponses de l'ensemble des modèles plutôt qu'un seul correspond à un des éléments-clés pour obtenir une bonne performance de généralisation. Malheureusement, cette approche vient avec un coût de calcul élevé, et trouver de bonnes approximations est un sujet de recherche actif.

Dans cette thèse, nous présentons une approche novatrice qui peut être appliquée avec un faible coût de calcul sur un large éventail de configurations d'apprentissage automatique. Pour atteindre cet objectif, nous appliquons la théorie de Bayes d'une manière différente de ce qui est conventionnellement fait pour l'apprentissage automatique. Spécifiquement, au lieu de chercher le vrai modèle à l'origine des données observées, nous cherchons le meilleur modèle selon une métrique donnée. Même si cette différence semble subtile, dans cette approche, nous ne faisons pas la supposition que le vrai modèle appartient à l'ensemble de modèles explorés. Par conséquent, nous disons que nous sommes agnostiques.

Plusieurs expérimentations montrent un gain de généralisation significatif en utilisant cette approche d'ensemble de modèles durant la phase de validation croisée. De plus, cet algorithme est simple à programmer et n'ajoute pas un coût de calcul significatif à la recherche d'hyperparamètres conventionnels. Finalement, cet outil probabiliste peut également être utilisé comme un test statistique pour évaluer la qualité des algorithmes sur plusieurs ensembles de données d'apprentissage.

Abstract

Machine learning is the science of learning from examples. Algorithms based on this approach are now ubiquitous. While there has been significant progress, this field presents important challenges. Namely, simply selecting the function that best fits the observed data was shown to have no statistical guarantee on the examples that have not yet been observed. There are a few learning theories that suggest how to address this problem. Among these, we present the Bayesian modeling of machine learning and the PAC-Bayesian approach to machine learning in a unified view to highlight important similarities. The outcome of this analysis suggests that model averaging is one of the key elements to obtain a good generalization performance. Specifically, one should perform predictions based on the outcome of every model instead of simply the one that best fits the observed data. Unfortunately, this approach comes with a high computational cost problem, and finding good approximations is the subject of active research.

In this thesis, we present an innovative approach that can be applied with a low computational cost on a wide range of machine learning setups. In order to achieve this, we apply the Bayes' theory in a different way than what is conventionally done for machine learning. Specifically, instead of searching for the *true* model at the origin of the observed data, we search for the *best* model according to a given metric. While the difference seems subtle, in this approach, we do not assume that the true model belongs to the set of explored model. Hence, we say that we are agnostic.

An extensive experimental setup shows a significant generalization performance gain when using this model averaging approach during the cross-validation phase. Moreover, this simple algorithm does not add a significant computational cost to the conventional search of hyper-parameters. Finally, this probabilistic tool can also be used as a statistical significance test to evaluate the quality of learning algorithms on multiple datasets.

Contents

List of Tables

List of Figures

Acknowledgement

My vast interest for machine learning pushed me to explore the different corners of this field. Unfortunately, some of these subdomains lie outside of the research interests of my advisors Mario Marchand and François Laviolette. Nonetheless, they followed me in these directions and helped me in any way possible. For that, I'm forever thankful. I would also like to thank them for putting together the GRAAL, a machine learning group in which I had the chance to meet amazing researchers. Among these, I would like to give special thanks to Jean-Francis Roy and Pascal Germain who continuously contributed to my development.

I cannot end this acknowledgement without mentioning Wendy Michelle Gill, my soul mate, who showed an incredible patience towards the slow convergence of this thesis and also corrected numerous grammar mistakes. Finally, Lyse Lacroix and Jacques Lacoste, my parents, offered an important financial and moral support during my whole doctorate studies.

Foreword

This thesis presents three selected works published during my doctorate studies. These publications stand out by the fact that they revolve around one particular topic and also by the fact that I am responsible for most of their realization. Here, I describe my contribution to these publications.

Included Publications

Bayesian comparison of machine learning algorithms on single and multiple data sets

Alexandre Lacoste, François Laviolette, and Mario Marchand
In *International Conference on Artificial Intelligence and Statistics*, 2012.

Published: Yes
Main Author: Alexandre Lacoste
My Contribution: Major

The main idea of this work came to me while I was running experiments comparing a few learning algorithms on several data sets. I then developed the model, the equations and the proofs. I also conducted the experiments and wrote the first draft of the paper. Mario Marchand and François Laviolette contributed mainly by teaching me how to write scientific papers. This valuable skill also helped me write all the following publications. Mario also rewrote some subsections to enhance the clarity of this work. Finally, François spent several hours discussing the ideas with me, which significantly oriented my thoughts.

Agnostic Bayesian Learning of Ensembles

Alexandre Lacoste, François Laviolette, Mario Marchand, and Hugo Larochelle
In *Proceedings of The 31st International Conference on Machine Learning*, 2014.

Published: Yes
Main Author: Alexandre Lacoste
My Contribution: Major

An early version of this work was also presented in the *NIPS workshop on Perturbations, Optimization, and Statistics workshop* in December 2012. In this work, I am responsible for the ideas, the models, the equations, the experiments and most of the writing. Mario and François helped me for clarifying the writing of the paper. After being rejected at ICML 2013, Hugo Larochelle joined the team and helped in rewriting some sections. He also helped me in refining my understanding of the Bayesian theory.

Sequential Model-Based Ensemble Optimization

Alexandre Lacoste, Hugo Larochelle, François Laviolette, and Mario Marchand
In *Proceedings of the 30th Conference on Uncertainty in Artificial Intelligence*, AUAI Press, 2014.

Published: Yes
Main Author: Alexandre Lacoste
My Contribution: Major

An early version of this work was also presented in the *ICML AutoML Workshop* in June 2014. In this work, I am responsible for the idea of the algorithm as well as all experiments and the first draft of the paper. Hugo significantly contributed to the discussion of the ideas and also contributed to the writing. He mostly took care of the introduction, the abstract and the conclusion. Mario and François contributed to the proofreading of this work.

Other Publications

I also participated in other publications during my doctorate studies. However, my contribution is minor and their topic diverges from that of this thesis. Hence, I briefly describe my contributions without including them in the thesis.

A PAC-Bayes Sample-Compression Approach to Kernel Methods

Pascal Germain, Alexandre Lacoste, François Laviolette, Mario Marchand, and Sara Shanian.
In *Proceedings of The 28th International Conference on Machine Learning*, 2011.

Published: Yes
Main Author: Sara Shanian (the author list is sorted alphabetically based on the last name)
My Contribution: Minor

During this time, I was working on a multi-kernel version of this work (unpublished). Since the proofs and experiments were similar, I was able to help writing and proofreading the theorems. I compared the experimental results with a different implementation of the algorithm and different optimization methods and conducted some of the experiments. Finally, I also contributed in the statistical significance methodology used in this work.

MHC-NP: Predicting Peptides Naturally Processed by the MHC

Sébastien Giguère, Alexandre Drouin, Alexandre Lacoste, Mario Marchand, Jacques Corbeil, and François Laviolette. In *Journal of immunological methods*, 400:30–36, 2013.

Published: Yes
Main Author: Sébastien Giguère
My Contribution: Minor

This publication describes the algorithm used to win the 2nd Machine Learning Competition in Immunology 2012.[1] Most of the work was executed by Sébastien Giguère and Alexandre Drouin. My main contribution was to perform model averaging over the set of hyperparameters to enhance the generalization performance of the final predictor. Interestingly, this triggered the main idea behind agnostic Bayes, the subject of this thesis. Sadly, for this contest, I only used a premature and suboptimal version of the agnostic Bayes ensemble. I also contributed by accelerating the execution of the core of the algorithm by a factor of ~ 16. This helped testing a wider range of hyperparameters.

[1] `http://bio.dfci.harvard.edu/DFRMLI/HTML/natural.php`

Chapter 1

Introduction to Machine Learning

For a long time, it was thought that *intelligence* was related to the skills intelligent people have. For example, playing chess and solving mathematical problems. Surprisingly, in the quest for artificial intelligence, these tasks were quickly tackled by early computer programs. It turns out that simpler tasks such as distinguishing the picture of a donkey from the picture of a monkey are much harder to program than what was previously thought. Simply associating a label to each possible image is intractable. For example, images of 100×100 pixels with 256 shades of gray yield $256^{100 \times 100}$ possible images (a number composed of 24083 digits), which is far beyond the number of atoms in the universe. To be more efficient, one can program a set of rules that distinguishes monkeys from donkeys based on the values of the pixels. Unfortunately, this approach is often tedious and highly sensitive to noise. For example, the monkey can be upside down, partly hidden behind foliage, with a tuxedo, sunglasses and a hat and still be a monkey. However, to achieve a reliable rule based algorithm for such a wide range of scenarios requires a complex set of rules that are carefully tested and tuned. Even then, it is likely to fail for scenarios that were not anticipated.

On the other hand most humans can *learn* such tasks after observing a few examples of each class. This led computer scientists to approach this kind of problem through learning by example and gave rise to the field of machine learning. In this paradigm, a **learning algorithm** is given a collection of observations and produces a function that *hopefully* generalizes to unseen examples with high accuracy. What is central to this idea is the concept of generalization, where it allows us to make predictions for situations we never encountered. We will see that, when the learning algorithm is designed properly, we can obtain statistical guarantees on the generalization performances of the learned model.

While machine learning was originally designed as a stepping stone towards artificial intelligence, the ability to generalize and make predictions turned out to be an amazingly useful ability and is now ubiquitous. Many services are now available in everyday life such as:

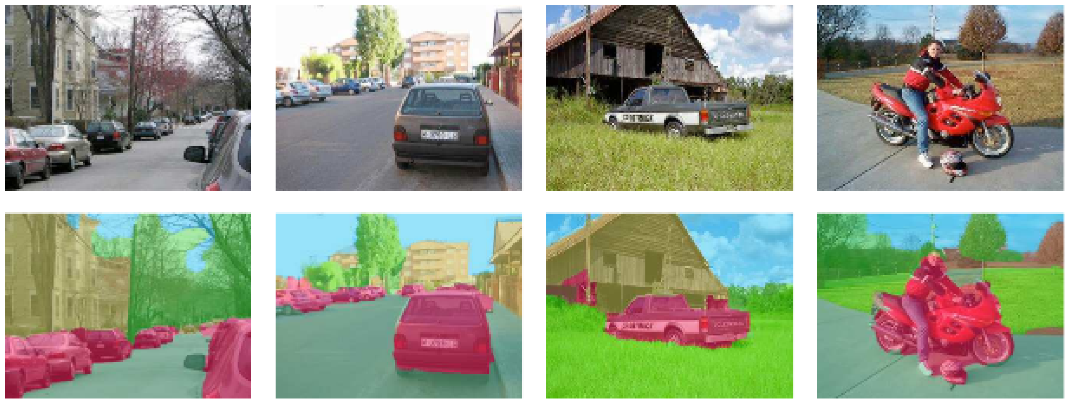


Figure 1.1: Scene labeling example. (Best seen in color.) The task consists of categorizing each pixel of an image into one of 8 predefined categories (sky, tree, road, grass, water, building, mountain, or foreground object). The first row corresponds to the original image, and a color mask is applied on the second row, to visualize the predictions made by the algorithm. Image taken from Farabet et al. [2013].

- Google's automatic translation service.[1]
- *Reliable* speech recognition systems on smartphones.
- Image search by content similarities[2].
- Netflix's movies recommendation system [Bell et al., 2007].

It is also often used to assist the scientific research, such as:

- Predictions of the complex interaction between proteins and molecules are used to assist the search for new drugs [Giguère et al., 2013].
- Estimating the distortion of galaxy images caused by the mass of dark matter, helps building a map of the dark matter in the universe.[3]
- Estimating the physical properties of the Higgs Boson.[4]

Finally it is worth noting the significant advances in tasks related to *perception* in artificial intelligence:

- Complex Scene Labeling (Figure 1.1).

[1]https://translate.google.com/
[2]https://images.google.com/
[3]https://www.kaggle.com/c/mdm
[4]http://www.kaggle.com/c/higgs-boson

- Solving language related tasks with almost no prior knowledge of the language [Collobert et al., 2011].
- Face detection and pose estimation in everyday pictures (Figure 1.2).

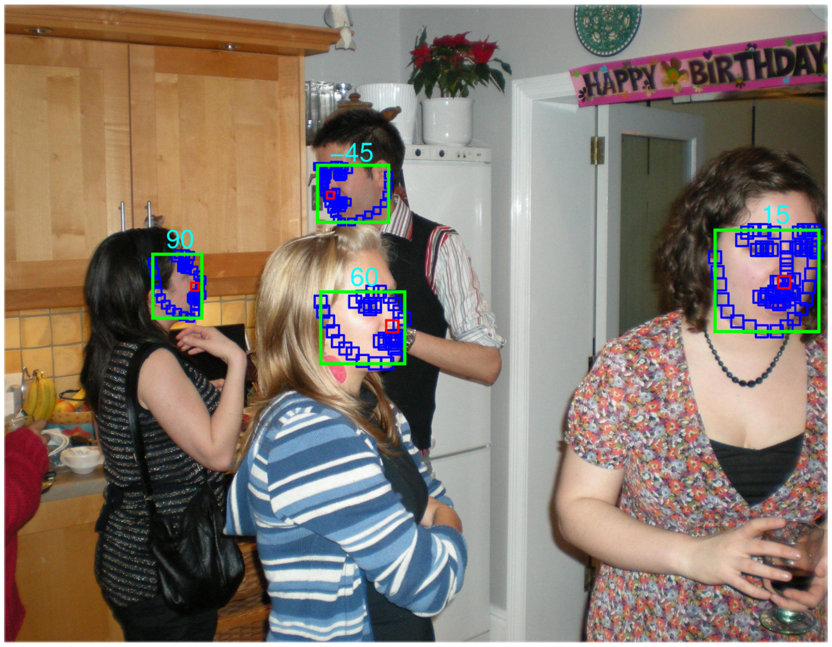


Figure 1.2: Face detection and pose estimation example. Image taken from Zhu and Ramanan [2012].

Some of these applications rely entirely on a supervised learning approach, the conventional formulation of machine learning previously described. However, the available data does not always conform to this paradigm and different formulations have been proposed to tackle the various situations encountered in practice. The content of this work will mainly focus on the supervised learning formulation, which will be presented in more details in the next section. To present a broader view of machine learning, we also briefly describe a few other commonly used machine learning paradigms.

1.1 Supervised Learning

We use $\mathcal{X}$ and $\mathcal{Y}$ to refer to the sets of all possible input and output values for a given task. In our example above, $\mathcal{X}$ corresponds to the set of all possible images composed of 100×100 grayscale pixels and $\mathcal{Y} = \{\text{'donkey'}, \text{'monkey'}\}$. In general $\mathcal{X}$ can be any set but it is commonly restricted to $\mathcal{X} \subseteq \mathbb{R}^k$ for simplifications. In this case, the elements composing the vector are called features and $\mathcal{X}$ can be referred to as the feature space.

A data set S is composed of m pairs (x_i, y_i), taking values in $\mathcal{X} \times \mathcal{Y}$ and $i \in \{1, 2, \ldots, m\}$. To conform to common statistical methods requirements, it is assumed that these m observations are independent and identically distributed (i.i.d.) samples coming from an unknown probability distribution D over $\mathcal{X} \times \mathcal{Y}$. This can also be denoted by $S \sim D^m$.

A learning algorithm $\mathcal{A}$ is simply a function that returns another function $h : \mathcal{X} \mapsto \mathcal{Y}$, based on the training set S, *i.e.*,

$$h = \mathcal{A}(S).$$

The generalization performance of h depends on the quality of the learning algorithm $\mathcal{A}$ but it also has a non deterministic component, *i.e.*, S is a random variable and a different training set $S' \sim D^m$ would yield a different predictor $h' = \mathcal{A}(S')$ with at different generalization performance. In order to quantify the generalization performance of the learned model, a given loss function is required. This function takes the form $\mathcal{L} : \mathcal{Y} \times \mathcal{Y} \mapsto \mathbb{R}$, which evaluates the cost of predicting $h(x)$ for a given $x \in \mathcal{X}$, when its true answer is $y \in \mathcal{Y}$. The expected loss over the distribution D is called the **true risk** of h, *i.e.*,

$$R_D(h) \overset{\text{def}}{=} \mathop{\mathbf{E}}_{x,y \sim D} \mathcal{L}(h(x), y).$$

With this metric[5] at hand, a machine learning task can be reduced to find, amongst a set of candidates $\mathcal{H}$, the predictor $h^\star$ minimizing the true risk, *i.e.*,

$$h^\star \overset{\text{def}}{=} \operatorname*{argmin}_{h \in \mathcal{H}} R_D(h).$$

Unfortunately, D is unknown and the risk can only be evaluated on the observed samples in S. This yields the **empirical risk**:

$$R_S(h) \overset{\text{def}}{=} \frac{1}{m} \sum_{i=1}^{m} \mathcal{L}\left(h\left(x_i\right), y_i\right).$$

As we will see in the next section, simply selecting the predictor minimizing $R_S(h)$ does not necessarily give a good generalization performance and solving this problem represents one of the main challenges of machine learning.

Historically, to simplify the analysis and to be able to exploit some mathematical properties, it has been convenient to work within restricted frameworks. Depending on the nature of the task to be solved, *i.e.*, the type of $\mathcal{Y}$, the following taxonomy was developed.

Classification: $\mathcal{Y} = \{0, 1, 2, \ldots, N\}$ e.g., distinguishing the different symbols of the alphabet given a handwritten version.

Binary Classification: $\mathcal{Y} = \{0, 1\}$ or $\mathcal{Y} = \{-1, 1\}$ e.g., is a mushroom poisonous or not, given some physical characteristics?

[5]There exists useful metrics that cannot be written as an expected loss. In this work, we mainly focus on the expected loss case.

Regression: $\mathcal{Y} = \mathbb{R}$ e.g., predicting the temperature given some meteorological conditions.

Structured Output: $\mathcal{Y}$ is any set e.g., language translation, where the output is a sentence in the target language.

1.2 Overfitting

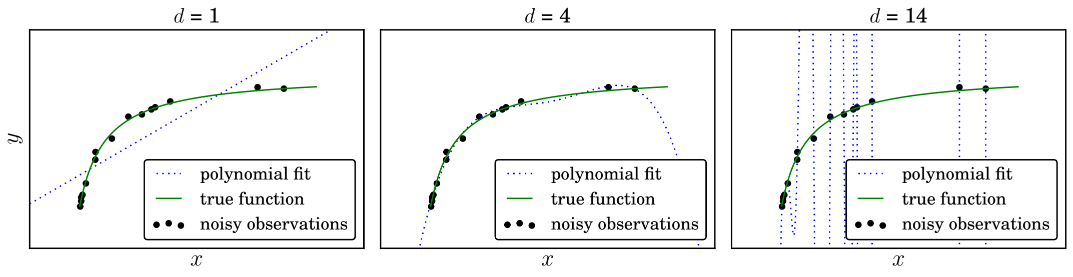


Figure 1.3: Example of underfitting and overfitting with polynomial curves. (Inspired from a scikit-learn tutorial.[6])

The formulation of supervised learning is a generalization of the more conventional idea of curve fitting, where a series of data points have been measured from a possibly noisy experiment and we are searching for the function that best fits these observations by minimizing the errors on the training set S. In this formulation, the input space $\mathcal{X}$ is of low dimensionality — usually $\mathcal{X} \subseteq \mathbb{R}$. This offers the opportunity to visualize the shape of the learned model and gain insight on its behavior. To this end, an example of polynomial curve fitting is depicted in Figure 1.3 for various values of d, the degree of the polynomial.[7] This example shows that when the degree of the polynomial is low, the model does not have enough *flexibility* to approximate the function. This is commonly referred to as **underfitting**. On the other hand, when the degree increases too much, it is possible to exactly fit all observed points in S. This sounds like a desirable property. However, in the rightmost image of Figure 1.3, we can observe that most predictions on the unobserved points are highly uncorrelated from the true underlying model. This is called **overfitting** and avoiding it is one of the main challenges in machine learning and will also be the main focus of this thesis.

1.3 Validation

In the previous example, the polynomial degree d seems to play an important role on the generalization ability of the fitted model. At first sight, finding the value of d that best fits the training set S seems like a good idea. Unfortunately, this would systematically select

[6]`http://www.astroml.org/sklearn_tutorial/practical.html`

[7]The true fucnction is $f(x) = 10 - 1/(x + 0.1)$ and the observed points are subjected to an additive gaussian noise with a standard deviation of 0.2.

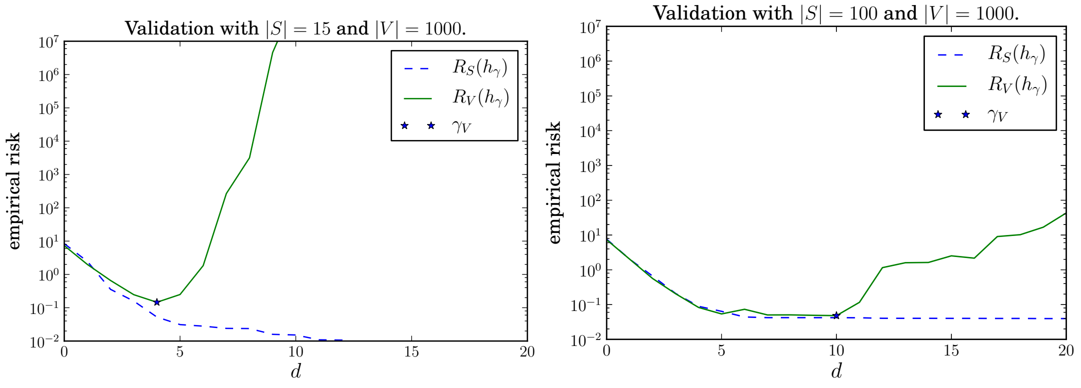


Figure 1.4: Comparison of validation risk $R_V(h_\gamma)$ and training risk $R_S(h_\gamma)$ for increasing values of the degree d in polynomial curve fitting. In the left figure, $|S| = 15$ and in the right figure, $|S| = 100$.

$d \geq m - 1$, which allows enough flexibility to exactly fit the observed data. Instead, what is commonly done is to *hide* a fraction of the data during training time and use it to evaluate the empirical risk for various values of d. More formally, the original data set is randomly partitioned into a training set S and a validation set V. They are both independent and considered to be composed of i.i.d. samples coming from D.

Parameters that need to be adjusted on a validation set are called hyperparameters and some learning algorithms can have more than one. In fact, the choice of the learning algorithm itself can also be seen as a hyperparameter. We use γ to represent a particular configuration of hyperparameter and $\mathcal{A}_\gamma$ to designate the corresponding learning algorithm. Then, we can train many different configurations γ from a predefined set Γ and choose the best predictor according the the validation set V and a given loss function $\mathcal{L}$. This yields $h_\gamma = \mathcal{A}_\gamma(S)$ and

$$\gamma_V \overset{\text{def}}{=} \operatorname*{argmin}_{\gamma \in \Gamma} R_V(h_\gamma),$$

where we use γ_V to designate the best hyperparameter configuration found by the validation set V, on Γ.

To evaluate the performance of the polynomial curve fitting in the previous example, we use the square difference loss function, *i.e.*,

$$\mathcal{L}(y', y) = (y' - y)^2.$$

In Figure 1.4, we compare the training risk, $R_S(h_\gamma)$, against the validation risk, $R_V(h_\gamma)$, for $d \in \{0, 1, 2, \ldots, 20\}$. This clearly expresses the trade-off obtained by carefully selecting d. In the left figure, the training is done with 15 samples, the minimum validation risk is 0.146 and is obtained for $d = 4$. In the right figure, we increased the training size to 100 samples. We observe that, in this case, we can afford a higher degree of the polynomial before observing a significant overfitting. The minimum validation risk is then 0.048 and occurs at $d = 10$.

1.4 Cross-Validation

The validation technique expressed in the previous section is useful to prevent overfitting. However, by partitioning the data set, an important fraction of the data is missing from the training set. This usually degrades the generalization performance of the final prediction. To alleviate this problem, we can use most of the data for the training set and the rest for validation. Sadly, this now causes the validation risk estimator to become noisy and less reliable. Luckily, to overcome this, it suffices to repeat the process with a different partitioning of the data set and average the individual validation risk to reduce its variance. This idea is implemented by k-fold cross-validation, where the original data set is separated in k subsets which are iteratively used as the validation set. Namely, let $\{V_1, V_2, \ldots, V_k\}$ be a partition of S, and let $h_{\gamma,j} \stackrel{\text{def}}{=} \mathcal{A}_\gamma\left(S \setminus V_j\right)$. Then, the cross-validation risk is

$$R_{\mathrm{CV}}(\gamma) \stackrel{\text{def}}{=} \frac{1}{k}\sum_{j=1}^{k} R_{V_j}(h_{\gamma,j}).$$

This gives a metric for selecting γ,

$$\gamma_{\mathrm{CV}} \stackrel{\text{def}}{=} \operatorname*{argmin}_{\gamma\in\Gamma} R_{\mathrm{CV}}(\gamma).$$

The final model is then retrained on the full data set S using γ_{CV}, *i.e.*,

$$h_{\gamma_{\mathrm{CV}}} \stackrel{\text{def}}{=} \mathcal{A}_{\gamma_{\mathrm{CV}}}(S).$$

This then raises a subtle question: Since we retrain on the full data set using the selected γ, why is it important to validate with as much training data as possible? It turns out that the *appropriate* value of γ depends on m, the size of the training set. For example, in the polynomial curve fitting example, if we have more training data, we can afford to use a higher degree d before being plagued by the overfitting problem. Therefore, with a higher k, $|S \setminus V_j| \approx |S|$ and the influence of the hyperparameter configuration γ has less chance to be biased but it comes with a greater computational cost. Common cases are $k = 5$ or $k = 10$ and the special case where $k = m$ is called **leave-one-out cross-validation**.

1.5 Other Formulations of Machine Learning

“Big data is like teenage sex: everyone talks about it, nobody really knows how to do it, everyone thinks everyone else is doing it, so everyone claims they are doing it...”

– Dan Ariely, *January 6, 2013*

One of the most straightforward ways to increase the generalization performance is to get more data. We are in the era of *big data*, where more and more users are working or playing through online services and everything is logged. The amount of sensors surrounding us is increasing everyday and their price is getting lower. Hard drives are filling up with (hopefully) valuable information. Sadly, most of this data does not fit in the supervised learning framework or at least, not directly for the task we need to solve. However, in many cases, this data contains *patterns* that could help solving other tasks with less data. This gave rise to **Inductive Transfer Learning**, which aims at *transferring* information from *source* tasks to a *target* task. Inductive transfer learning does not have a precise framework. Instead it describes the generic idea of reusing information coming from other tasks to simplify the learning of the target task. To address specific problems, the following sub-domains have been developed[8]:

Multitask Learning: When more than one task shares a common structure, it can be exploited by simultaneously solving all of them. For example, spam filtering can be personalized by considering each email account as a different task. While the training set of each of these individual tasks is too small to obtain a reliable spam filter, considering a common representation and solving them simultaneously makes this personalization possible [Weinberger et al., 2009].

Domain Adaptation: This approach assumes that the source distribution D_{source} over $\mathcal{X} \times \mathcal{Y}$ is *similar*[9] to the target distribution D_{target}, but not identical. It also assumes that the information in the source data set is abundant enough to learn a reliable predictor. Then, this predictor is adapted to the target task using the information available in the corresponding data set. For example, Amazon's reviews on books consists in a substantial data set and we would like to use it to enhance the predictions on DVD reviews [Chen et al., 2012].

Other approaches use a single task but work around the scarcity of the label. For example, **semi-supervised** learning uses the idea that unlabeled samples are abundant but labeling them is expensive. Thus, the usual training set S is accompanied by an unlabeled set U sampled from the marginal distribution $p(x) = \sum_{y \in \mathcal{Y}} D(x, y)$. This unlabeled data set can then be used to learn a similarity function based on the density $p(x)$ [Belkin et al., 2006]. **Active learning** offers an extension to this idea where it is possible to iteratively query the label for specific x. By carefully choosing x, it is possible to significantly accelerate the learning [Tong and Koller, 2002]. Finally, **transductive learning** is similar to semi-supervised learning in the sense that it is accompanied by an unlabeled set U. However, the goal is solely to predict the labels on the set U instead of inducing a function. This simplification allows an increase in generalization performance in certain cases [Joachims, 1999].

[8]There is not a common agreement on the exact definition of these terms and some authors describe domain adaptation and inductive transfer learning as two distinct frameworks.

[9]The precise definition of similarity depends on the author.

1.6 Thesis Outline

In the next chapter, we show how the Bayesian modeling of machine learning offers a simple solution to the overfitting problem. We also expose the basic elements of the PAC-Bayes theory and we show that it provides the same solution to the overfitting problem.

Then, the rest of the thesis presents three of my main publications in chronological order. Odd chapters serve as a presentation while even chapters correspond to a formatted version of the publication. Their notation was adapted to avoid important clash with the current notation. An early version of two of these works were also published in workshops and are provided in Appendix A and Appendix B.

The first publication presents the Poisson binomial test, a Bayesian statistical significance test designed to compare the generalization performance of two learning algorithms (Chapter 4). The second publication focuses on enhancing the generalization performance of the validation process. It does so by integrating out the uncertainty arising when selecting a single predictor based on a finite validation set (Chapter 6). Finally, the third work combines this validation enhancement with a new optimization technique used to accelerate the search of the minimal validation risk (Chapter 8).

Chapter 2

Priors and Model Averaging

As seen previously, simply selecting the model that best *fits* the observed data may yield the problem of *overfitting*. This usually gives high accuracy for predictions made on observed data. However, the final predictor is deprived of any guarantee on the predictions made for unseen data. At first sight, this looks like a degenerated problem, where learning from examples would be a hype with no real solution. Fortunately, the machine learning community proposed several learning theories to address this problem. These theories are often based on different approaches and propose different solutions. Interestingly, there are two key elements that are at the core of most of these approaches:

- Prior, also known as "function class complexity",
- Model Averaging, a concept that is often implicit or ignored.

The first element aims at limiting the complexity of the set of candidate models. The second element integrates out the uncertainty about which model is the true underlying model.

In this chapter, we offer an overview of Bayesian model averaging as well as an overview of the PAC-Bayesian theory. While being based on fundamentally different axioms, we show that both approaches agree on the final solution. This outlines the fact that using an appropriate prior and performing model averaging are the two key aspects for a good generalization performance. This will serve as a foundation for the rest of this thesis.

2.1 Bayesian Model Averaging

The general idea behind Bayesian modelization of machine learning is that with a finite amount of observations, we cannot find with certainty the *true* model at the origin of the observations in S. However, it is possible to obtain a probability distribution over all considered models $\theta \in \Theta$ that are *likely* to be the one. Then, when a prediction needs to be made on an unseen test point, all models are queried to provide a probability distribution over possible answers.

> *"Overfitting does not apply to fully Bayesian methods since they don't involve any fitting."*
>
> – Zoubin Ghahramani, *MLSS 2012*

Probability theory offers two simple rules to transform joint probability distributions.

$$p(a) = \sum_b p(a, b) \tag{2.1}$$

$$\begin{aligned} p(a, b) &= p(a|b)p(b) \\ &= p(b|a)p(a). \end{aligned} \tag{2.2}$$

The first rule is called *marginalization* and the second one *factorization.* Note that in these equations, it is implicit that a and b are realizations of the random variables A and B and that the summation goes for all possible values of b. When b is a continuous variable, it should be integrated. In order to keep a light notation, we leave these details implicit whenever there is no ambiguity. We also note that we can always condition on an extra variable c without affecting the above equalities, *i.e.*, $p(a|c) = \sum_b p(a, b|c)$ and $p(a, b|c) = p(a|b, c)p(b|c) = p(b|a, c)p(a|c)$. In the case where a and b are independent, *i.e.*, the knowledge of b does not influence the probability of a, we can write $p(a|b) = p(a|b) = p(a)$, where we use the gray notation when we want to highlight the fact that this information is available but does not influence the probability distribution.

Interestingly, these two simple rules are sufficient to write a complete probabilistic modelization of machine learning. As a first step, we obtain Bayes theorem by simply considering the two different ways of factorizing a joint distribution (Equation 2.2) and rearranging the terms

$$p(a|b) = \frac{p(b|a)p(a)}{p(b)}. \tag{2.3}$$

There is nothing *Bayesian* about this equation until we attribute meanings to the different terms that compose it. To do so, let's consider an example related to machine learning. Let S be the training data set and $\theta \in \Theta$ the collection of parameters that compose the model, where Θ corresponds to the family of possible models. This gives

$$p(\theta|S) = \frac{p(S|\theta)p(\theta)}{p(S)},$$

where we attribute the following meanings to each term:

Prior: $p(\theta)$ represents the probability that θ corresponds to the *true* model before any observation is made. At this point, we usually don't have much information about which one is the true model and this should be reflected in our choice of $p(\theta)$, *i.e.*, it should have a high entropy.

Posterior: $p(\theta|S)$ is still a probability distribution over Θ. However, it is now conditioned on the observation of S. In other words, this corresponds to the probability that model θ is at the origin of the observations in S.

Likelihood: $p(S|\theta)$ corresponds to the probability of observing the data set S *supposing* that it was generated by model θ. Using the i.i.d. assumption, it can be factorized as follows $p(S|\theta) = \prod_{i=1}^{m} p(x_i, y_i|\theta)$.

Marginal Likelihood: $p(S) = \sum_{\theta \in \Theta} p(S|\theta)p(\theta)$ can be seen as a simple normalization factor since it does not depend on θ. Later, we will see that this term can measure the *complexity* of the current prior and can serve as a model selection tool.

Based on the attributed meaning, we first have to choose a set of hypothetic models $\theta \in \Theta$ defining a probability distribution $p(x, y|\theta)$ over $\mathcal{X} \times \mathcal{Y}$. Then, after specifying a prior $p(\theta)$ over Θ and observing S, we obtain a posterior distribution $p(\theta|S)$. Bayes posterior is in general consistent [Ibragimov and Has'minskii, 1981, Chapter 1], that is, it *almost surely converges* to the true model θ_{true} as $m \to \infty$, where $p(x, y|\theta_{\text{true}}) = D(x, y)$. However, this may not always be the case, *e.g.*, $p(\theta_{\text{true}}) = 0 \ \Rightarrow \ p(\theta_{\text{true}}|S) = 0$. This situation is referred to as a *mis-specified prior.*

2.1.1 Discriminative Model

"One should solve the [classification] problem directly and never solve a more general problem as an intermediate step."

– Vladimir Vapnik, *1998*

We have to keep in mind that the goal of supervised learning is not to find the true distribution D, but to perform predictions for a new point $x \in \mathcal{X}$. Namely, we are searching for the distribution $p(y|x, \theta)$, and not $p(y, x|\theta)$. Using the i.i.d. assumption and the factorization rule, the likelihood can be written

$$\begin{aligned} p(S|\theta) &= \prod_{i=1}^{m} p(x_i, y_i|\theta) \\ &= \prod_{i=1}^{m} p(y_i|x_i, \theta) \prod_{i=1}^{m} p(x_i|\theta). \end{aligned}$$

We note that $p(x)$ is not explicitly modeled[1] and it does not depend on θ. This allows us to write the following, using Bayes' theorem

$$\begin{aligned} p(\theta|S) &= \frac{p(S|\theta)p(\theta)}{\sum_{\theta\in\Theta} p(S|\theta)p(\theta)} \\ &= \frac{\prod_{i=1}^m p(x_i)}{\prod_{i=1}^m p(x_i)} \frac{\prod_{i=1}^m p(y_i|x_i,\theta)p(\theta)}{\sum_{\theta\in\Theta}\prod_{i=1}^m p(y_i|x_i,\theta)p(\theta)}. \end{aligned}$$

Since $p(x)$ does not depend on θ, it is possible to move it outside of the summation and cancel it out with its numerator instance. Thus, we never have to explicitly model $p(x)$. This approach is called the **discriminative model** and it corresponds to an important simplification since $p(x)$ usually represents a complex probability distribution, *e.g.*, the probability of observing any images composed of 100×100 pixels. This approach contrasts with the **generative model** where the joint distribution $p(x, y|\theta)$ is modeled. This can be useful if our task requires us to generate samples from $p(x)$. Otherwise, it is often considered *wise* to choose the discriminative approach [Ng and Jordan, 2001] and this will be the focus for the rest of this work.

To simplify equations, we use $S_x \overset{\text{def}}{=} \{x_i\}_{i=1}^m$, $S_y \overset{\text{def}}{=} \{y_i\}_{i=1}^m$ and $S = (S_x, S_y)$. This allows us to write $p(S_y|S_x,\theta) = \prod_{i=1}^m p(y_i|x_i,\theta)$ and the the posterior for the discriminative model becomes

$$p(\theta|S) = \frac{p(S_y|S_x,\theta)p(\theta)}{\sum_{\theta\in\Theta} p(S_y|S_x,\theta)p(\theta)}.$$

Also, since the marginal likelihood $p(S_y|S_x) = \sum_{\theta\in\Theta} p(S_y|S_x,\theta)p(\theta)$ does not depend on θ (it has been integrated out), we can write the posterior as follows

$$p(\theta|S) \propto p(S_y|S_x,\theta)p(\theta).$$

To model $p(y|x,\theta)$, it is common to consider a deterministic function $h \in \mathcal{H}$ with an output noise model η. Hence, we have $\theta = (h, \eta)$ and $p(y|x,\theta) = p(y|x,h,\eta)$. It is also possible to consider an input noise model, but this is less common.

2.1.2 Model Averaging

The Bayes posterior gives us a probability over which model is likely to be the *true* one. However, with a finite sample size, we cannot identify it. How do we choose the right one then? Some authors simply select the most probable one, also known as *maximum a posteriori* (MAP). Unfortunately, this is equivalent to choosing the model that best fits the observed points and, as we have already seen, it can suffer from important overfitting. A solution to this problem is then to consider the answer of each of these models and to weigh their predictions according to the posterior distribution. Namely, from the probabilistic rules 2.2

[1] We could write $p(x|\phi)$ instead, where $\phi \in \Phi$ represents parameters that do not need to be inferred.

and 2.1, we have

$$
\begin{aligned}
p(y|x,S) &= \sum_{\theta\in\Theta} p(y|x,S,\theta)p(\theta|x,S) \\
&= \sum_{\theta\in\Theta} p(y|x,\theta)p(\theta|S), \qquad (2.4)
\end{aligned}
$$

In this operation, we have integrated out the model θ. Thus, the final distribution $p(y|x,S)$ does not depend on θ and we have fully taken into account the uncertainty about which θ was the true model. We refer to this approach as **model averaging** and we claim that this corresponds to the overfitting's nemesis.

2.1.3 Example with Linear Regression

To avoid getting lost in generic equations, we take a pause to address linear regression with a Bayesian approach. This will give us a concrete example for the rest of this section. This learning algorithm is derived in many different textbooks, but here, we present an adapted version from Rasmussen and Williams [2006, Chapter 2.1].

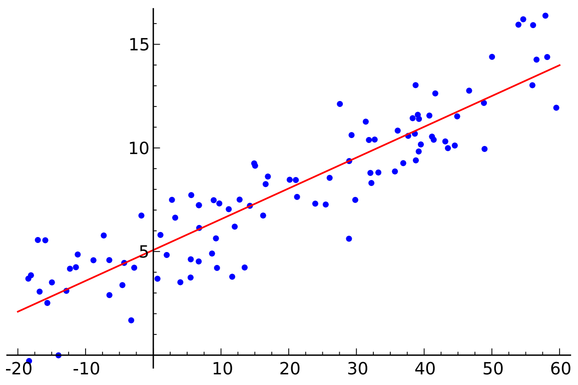


Figure 2.1: One dimensional linear regression example with Gaussian noises. Image taken from Wikipedia.

Linear regression addresses the specific machine learning setup where $\mathcal{X} = \mathbb{R}^d$, $\mathcal{Y} = \mathbb{R}$ and $\mathcal{H}$ is the set of d-dimensional linear functions (See Figure 2.1). To favor the usage of the linear algebra notation, in this section $x \in \mathbb{R}^d$ is a vector, $X \overset{\text{def}}{=} (x_1, x_2, \ldots, x_m) \in \mathbb{R}^{d\times m}$ is a matrix and $\mathbf{y} \overset{\text{def}}{=} (y_1, y_2, \ldots, y_m)^T \in \mathbb{R}^m$ is a vector. Since $\mathcal{H}$ is a set of linear functions, the parameters of the model, $\mathbf{w} \in \mathbb{R}^d$, are also a vector and each hypothesis has the form[2]

$$h(x) = \mathbf{w}^T x.$$

To properly define the likelihood function $p(y|x,\mathbf{w})$, we need a noise model. Any noise model is valid, but it turns out that an additive Gaussian noise yields interesting mathematical

[2]As is commonly done, we omit the extra parameter for the bias as it can be included in $\mathbf{w}$ by adding an extra dimension to x and setting its value to 1.

properties and is appropriate for many real world problems. Thus, we have $y = h(x) + \varepsilon$, where $\varepsilon \sim \mathcal{N}\left(0, \sigma^2\right)$, which gives

$$p(y|x, \mathbf{w}, \sigma) = \mathcal{N}\left(y \;\middle|\; \mathbf{w}^T x, \sigma^2\right).$$

The parameters of the model are $\theta = (\mathbf{w}, \sigma)$, but for simplicity, we will consider σ as a known parameter for now. Hence, we use $\theta = \mathbf{w}$. With this at hand, we can define the likelihood as follows

$$\begin{aligned}
p(\mathbf{y}|X, \mathbf{w}, \sigma) &= \prod_{i=1}^{m} p(y_i|x_i, \mathbf{w}, \sigma) \\
&= \prod_{i=1}^{m} \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{(y_i - x_i^T\mathbf{w})^2}{2\sigma^2}\right) \\
&= \frac{1}{(2\pi\sigma^2)^{m/2}} \exp\left(-\frac{1}{2\sigma^2}(\mathbf{y} - X^T\mathbf{w})^T(\mathbf{y} - X^T\mathbf{w})\right) \\
&= \mathcal{N}\left(\mathbf{y} \;\middle|\; X^T\mathbf{w}, \sigma^2 I\right),
\end{aligned}$$

where I is the $m \times m$ identity matrix. We use the prior $p(\mathbf{w}) = \mathcal{N}\left(\mathbf{w} \mid \mathbf{0}, \Sigma_\mathrm{p}\right)$ (this choice will be explained in Section 2.1.6). Since this prior is parametrized by the covariance matrix Σ_p, the prior $p(\mathbf{w})$ is written $p(\mathbf{w}|\Sigma_\mathrm{p})$. Using $A \overset{\text{def}}{=} \sigma^{-2}XX^T + \Sigma_\mathrm{p}^{-1}$ and $\bar{\mathbf{w}} \overset{\text{def}}{=} \sigma^{-2}A^{-1}X\mathbf{y}$, we can write the posterior as follows

$$\begin{aligned}
p(\mathbf{w}|X, \mathbf{y}, \sigma, \Sigma_\mathrm{p}) &\propto p(\mathbf{y}|X, \mathbf{w}, \sigma)p(\mathbf{w}|\Sigma_\mathrm{p}) \\
&\propto \exp\left(-\frac{1}{2\sigma^2}(\mathbf{y} - X^T\mathbf{w})^T(\mathbf{y} - X^T\mathbf{w})\right) \exp\left(-\frac{1}{2}\mathbf{w}^T\Sigma_\mathrm{p}^{-1}\mathbf{w}\right) \\
&\propto \exp\left(-\frac{1}{2}(\mathbf{w} - \bar{\mathbf{w}})^T A(\mathbf{w} - \bar{\mathbf{w}})\right),
\end{aligned}$$

where the last equation is obtained by "completing the square" and, by its form, we identify that it corresponds to a normal distribution. This saves us from having to explicitly calculate the normalization factor and we obtain

$$p(\mathbf{w}|X, \mathbf{y}, \sigma, \Sigma_\mathrm{p}) = \mathcal{N}\left(\mathbf{w} \;\middle|\; \bar{\mathbf{w}}, A^{-1}\right).$$

Finally, by using the model averaging principle presented in Section 2.1.2, we obtain

$$\begin{aligned}
p(y|x, X, \mathbf{y}) &= \int_{\mathbf{w}\in\mathbb{R}^d} p(y|x, \mathbf{w})p(\mathbf{w}|X, \mathbf{y}, \sigma, \Sigma_\mathrm{p})d\mathbf{w} \\
&= \int_{\mathbf{w}\in\mathbb{R}^d} \mathcal{N}\left(y \;\middle|\; \mathbf{w}^T x, \sigma^2\right)\mathcal{N}\left(\mathbf{w} \;\middle|\; \bar{\mathbf{w}}, A^{-1}\right)d\mathbf{w} \\
&= \mathcal{N}\left(y \;\middle|\; \frac{1}{\sigma^2}x^T A^{-1}X\mathbf{y},\ x^T A^{-1}x\right).
\end{aligned}$$

The simplicity of this result is impressive. This is a consequence of choosing a linear function space with an additive Gaussian noise model and a normally distributed prior. In general, model averaging does not yield a closed-form solution and we need to rely on approximation methods. While linear regression is limited to very specific tasks, by projecting x on a higher dimensional space (possibly infinite) with a mapping function ϕ, we gain access to a function space more flexible than linear functions. This gives rise to the Gaussian Process algorithm [Rasmussen and Williams, 2006, Chapter 2], a very flexible algorithm that is appropriate for a wide range of tasks. The choice of the mapping function ϕ can be seen as being part of the prior, but we will see in Section 2.1.6 that this parameter can also be marginalized. For now, we come back to the Bayesian modelization of machine learning.

2.1.4 Optimal Bayes Decision

From Bayes' theorem and the model averaging principle, we have obtained the distribution $p(y|x, S)$. We are close to the goal; but to fulfill the requirements of supervised learning, we still have to commit to a particular answer for a given x. This can only be achieved when the loss function $\mathcal{L} : \mathcal{Y} \times \mathcal{Y} \mapsto \mathbb{R}$ is specified. With it, we can select the answer with the least consequence according to $p(y|x, S)$, this gives the **optimal Bayes predictor**:

$$\mathrm{B}_{\mathrm{opt}}(x) \overset{\mathrm{def}}{=} \underset{y' \in \mathcal{Y}}{\operatorname{argmin}} \sum_{y \in \mathcal{Y}} p(y|x, S)\mathcal{L}(y', y). \tag{2.5}$$

For a given loss, this equation can be simplified [Robert, 2001, Chapter 2].

zero-one loss: $\mathcal{L}(y', y) = \mathrm{I}[y' \neq y]$, where I corresponds to the indicator function. In this case, the most probable answer is the one satisfying Equation 2.5, *i.e.*,

$$\mathrm{B}_{\mathrm{opt}}(x) = \underset{y \in \mathcal{Y}}{\operatorname{argmax}}\ p(y|x, S).$$

quadratic difference loss: $\mathcal{L}(y', y) = (y' - y)^2$. In this case, the expected value is the appropriate answer, *i.e.*,

$$\mathrm{B}_{\mathrm{opt}}(x) = \sum_{y \in \mathcal{Y}} y p(y|x, S).$$

absolute difference loss: $\mathcal{L}(y', y) = |y' - y|$. In this case, the appropriate answer is the median, *i.e.*, $\mathrm{B}_{\mathrm{opt}}(x) = \tau$ such that

$$\Pr(y \leq \tau | x, S) = \Pr(y \geq \tau | x, S).$$

Since the quadratic difference loss reduces to the expected value of $p(y|x, S)$, this offers an important simplification.

$$\begin{aligned}
\mathrm{B}_{\mathrm{opt}}(x) &= \sum_{y \in \mathcal{Y}} y p(y|x, S) \\
&= \sum_{\theta \in \Theta} p(\theta|S) \left(\sum_{y \in \mathcal{Y}} y p(y|x, \theta) \right),
\end{aligned}$$

which corresponds to the model averaging of deterministic predictors. When using linear models with additive noise such as described in Section 2.1.3, this simplifies further more

$$\begin{aligned}
\mathrm{B}_{\mathrm{opt}}(x) &= \int_{\mathbf{w}\in\mathbb{R}^d} p(\mathbf{w}|X,\mathbf{y},\sigma,\Sigma_{\mathrm{p}}) \int_{y\in\mathbb{R}} y\ p(y|x,\mathbf{w},\sigma)\ dy\ d\mathbf{w} \\
&= \int_{\mathbf{w}\in\mathbb{R}^d} p(\mathbf{w}|X,\mathbf{y},\sigma,\Sigma_{\mathrm{p}})\ x^T\mathbf{w}\ d\mathbf{w} \\
&= x^T \int_{\mathbf{w}\in\mathbb{R}^d} \mathbf{w}\mathcal{N}\left(\mathbf{w}\ \middle|\ \bar{\mathbf{w}}, A^{-1}\right)\ d\mathbf{w} \\
&= x^T\bar{\mathbf{w}},
\end{aligned}$$

where $\bar{\mathbf{w}} \stackrel{\text{def}}{=} \sigma^{-2}A^{-1}X\mathbf{y}$. When limiting Σ_{p} to be of the form $\sigma_{\mathrm{p}}I$, we have[3]

$$\bar{\mathbf{w}} = \left(XX^T + \frac{1}{\sigma_{\mathrm{p}}}I\right)^{-1} X\mathbf{y},$$

which corresponds to the ridge regression algorithm [Rasmussen and Williams, 2006, Chapter 6.2] also known as Tikhonov regularization. We note however that, in this particular case, $\underset{\mathbf{w}\in\mathbb{R}^d}{\operatorname{argmax}}\ p(\mathbf{w}|X,\mathbf{y},\sigma,\Sigma_{\mathrm{p}}) = \bar{\mathbf{w}}$. Hence, the maximum a posteriori model coincides with the behavior of the full model averaging.

2.1.5 Practical Implementation

At this point, we now have a complete probabilistic solution to the supervised learning framework. It directly followed from the two probability rules 2.2 and 2.1 and the i.i.d. assumption[4]. To highlight the simplicity of this approach, we quickly review the 4 steps required to achieve the final prediction:

1. Choose a **prior** $p(\theta)$ over a family of models able to express $p(y|x,\theta)$.

2. Compute the **posterior** distribution

$$p(\theta|S) \propto \prod_{i=1}^{m} p(y_i|x_i,\theta)p(\theta).$$

3. Perform **model averaging**

$$p(y|x,S) = \sum_{\theta\in\Theta} p(y|x,\theta)p(\theta|S).$$

4. Compute the **optimal decision**

$$\mathrm{B}_{\mathrm{opt}}(x) \stackrel{\text{def}}{=} \underset{y'\in\mathcal{Y}}{\operatorname{argmin}} \sum_{y\in\mathcal{Y}} p(y|x,S)\mathcal{L}(y',y).$$

[3]Note that it does not depend on σ.

[4]Note that the i.i.d. assumption is not necessary. However, it greatly simplifies the expression of the likelihood.

There is no approximation in these equations, but we don't have an algorithm yet. The task now resides in choosing an appropriate prior and finding a way to perform efficient model averaging, hence the title of this chapter. The difficulty is that, for most priors of interest, Θ is an infinite uncountable set, causing exact model averaging to be computationally intractable. As we have seen in Section 2.1.3, some priors offer a closed form solution for model averaging, but in general, we have to rely on approximations. To this end, we present a quick overview of a few common approximation methods available in the Bayesian literature.

Maximum A Posteriori: As seen previously, the MAP approach simply chooses the most probable model. This can be seen as a form of approximation and is often used in practice. However, it completely ignores the amount of uncertainty we have about which model is good. Even though it is often used in practice, this approach does not always have a generalization guarantee.

Sampling From The Posterior: In order to approximate the infinite sum (or integral) of Equation 2.4, it is possible to use a Monte Carlo method. This consists in sampling N times $\theta_i \sim p(\theta|S)$ and $y_i \sim p(y|x, \theta_i)$. Then, the collection $\{y_i\}_1^N$ is used to approximate $p(y|x, S)$ or to directly address the optimal Bayes decision (Equation 2.5), *e.g.*, in the case of the quadratic difference loss, the answer is $\mathrm{B}_{\mathrm{opt}}(x) = \frac{1}{N}\sum_{i=1}^{N} y_i$. Sadly, it is generally not possible to directly sample from $p(\theta|S)$. To tackle this, techniques such as Markov Chain Monte Carlo (MCMC) can be used [Andrieu et al., 2003]. This yields a highly flexible algorithm and a good approximation when N is sufficiently large. The drawback is that MCMC algorithms need to converge before generating reliable samples and the convergence time is usually long and unknown.

Variational Bayes: An efficient approach is to search for the distribution $q(\theta)$ amongst a set of distribution $\mathcal{Q}$ that best approximate the posterior $p(\theta|S)$ according to the Kullback-Leibler divergence[5] between $q(\theta)$ and $p(\theta|S)$. The set $\mathcal{Q}$ is chosen such that model averaging becomes computationally tractable when $p(\theta|S)$ is approximated by $q(\theta)$. For more details, Fox and Roberts [2012] offer a great introduction.

2.1.6 Choosing the Prior

Now that we have practical solutions to perform model averaging, we are left with the task of selecting an appropriate prior $p(\theta)$. This probability distribution should reflect the prior knowledge we have about this task. If we have none, then this distribution should be as *vague* as possible. What is paradoxical at this point is that machine learning aims at developing a generic learning algorithm able to tackle a wide variety of tasks, but Bayes' theorem suggests that choosing an appropriate prior for the given task would enhance the performance of the

[5]Other dissimilarity functions can be used

final predictor. For this reason, many learning algorithms are based on the following vague assumption. Let $(x, y) \sim D$ and $(x', y') \sim D$ then if x is *similar* to x', y is also *similar* to y'. By parameterizing the notion of *similarity*, we can get both benefits i.e, a flexible algorithm that can work for a wide range of tasks and the opportunity to specialize the algorithm for a given task. This notion is used in a wide range of popular learning algorithms and the rest of this section describes how to handle these prior parameters.

Approaches such as *Artificial Neural Networks* provide more flexibility on the prior by letting the choice of the architecture an open parameter. By doing so, a skilled engineer can communicate important prior information to the learning algorithm and can lead to an important increase in generalization performances. Also, the usage of *deeper* architectures has proven to be more efficient on many tasks, yielding a new branch of machine learning called *deep learning* [Bengio, 2009].

Marginal Likelihood

An important aspect of the prior is the notion of *complexity*, which is often related to the famous Ockham's razor. When the prior is highly flexible and distributes its probability mass amongst a wide range of models, then the good models will have really small mass and the learning will require more data. However, it is hard to know a priori which level of complexity is appropriate for the given task. For this reason, it is common to parametrize the prior and select the one that offers the best trade-off between *flexibility* and *simplicity.* To represent the parameters of the prior, we use $p(\theta|\gamma)$, where γ is any configuration of prior parameters taking values in a predefined set Γ. We note that these prior parameters are analogous to hyperparameters and could be selected through cross-validation (Section 1.3), but the Bayesian approach offers an interesting tool to achieve a similar result. This tool is nothing more than the marginal likelihood defined at the beginning of this section. Now that the prior is parametrized, the marginal likelihood is written as follows

$$p(S|\gamma) = \sum_{\theta \in \Theta} p(S|\theta) p(\theta|\gamma),$$

or in the case of discriminative modeling

$$p(S_y|S_x, \gamma) = \sum_{\theta \in \Theta} p(S_y|S_x, \theta) p(\theta|\gamma).$$

In other words, this quantity describes the probability of observing S amongst any other data set of the same size, given that we chose the prior related to γ. A toy representation of this idea is presented in Figure 2.2. It shows that when the prior is highly flexible it will be suited for a wide range of data sets, but $p(S|\gamma)$ is likely to be low. On the other extreme, if we choose a really specific prior, it will be highly suited for a restricted set of data sets. Finally, somewhere in between, there exists an appropriate choice that performs a good trade-off and this choice is the one maximizing the marginal likelihood (MML), *i.e.*,

$$\gamma_{\text{MML}} = \underset{\gamma \in \Gamma}{\text{argmax}}\; p(S|\gamma)$$

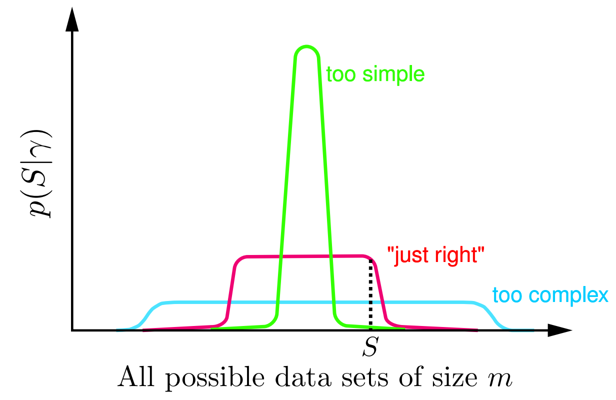


Figure 2.2: Toy representation of the marginal likelihood. Image adapted from Ghahramani [2013].

In the case of linear regression presented in Section 2.1.3, the marginal likelihood can also be expressed in a closed-form solution [Rasmussen and Williams, 2006, Chapter 2.2]

$$\begin{aligned} p(\mathbf{y}|X, \sigma, \Sigma_{\text{p}}) &= \int_{\mathbf{w} \in \mathbb{R}^d} p(\mathbf{y}|X, \mathbf{w}, \sigma) p(\mathbf{w}|\Sigma_{\text{p}}) d\mathbf{w} \\ &= \mathcal{N}\left(\mathbf{y} \;\middle|\; \mathbf{0},\; X^T \Sigma_{\text{p}} X + \sigma^2 I\right). \end{aligned}$$

Then, by differentiating the function $p(\mathbf{y}|X, \sigma, \Sigma_{\text{p}})$ with respect to σ and Σ_{p}, we can use gradient ascent to find the most appropriate parameters, which is by far more computationally efficient than cross-validation.

Hierarchical Bayes

Another way to deal with prior parameters is to marginalize them like other parameters by performing model averaging with a posterior $p(\gamma|S)$. In order to do this, we need to specify a *hyper-prior*[6] $p(\gamma)$ over the set of prior parameters Γ. This gives the following posterior distribution

$$p(\gamma|S) \propto p(S|\gamma)p(\gamma).$$

Then, it can be marginalized

$$p(y|x, S) = \sum_{\gamma \in \Gamma} p(y|x, S, \gamma)p(\gamma|S),$$

[6]It seems like we are just shoveling the problem further by using a hyper-prior, but this hyper-prior has less influence on the final posterior. If the choice of this hyper-prior still has too much influence on the end result, we can still use a *hyper-hyper-prior*.

where $p(y|x, S, \gamma)$ is the previously obtained result of $p(y|x, S)$ that is now conditioned on γ (refer to Section 2.1.2).

This approach also sheds light on the maximum marginal likelihood approach. If we consider an uniform prior over Γ, we have $p(\gamma|S) \propto p(S|\gamma)$ and

$$\underset{\gamma \in \Gamma}{\operatorname{argmax}}\ p(S|\gamma) = \underset{\gamma \in \Gamma}{\operatorname{argmax}}\ p(\gamma|S).$$

Hence, maximum marginal likelihood is equivalent to maximum a posteriori of γ under a uniform prior. This means that marginal likelihood is just an approximation of hierarchical Bayes. On the other hand, hierarchical Bayes requires an extra marginalization step which may be time consuming and the overfitting incurred when choosing the maximum a posteriori of a few prior parameters may have a small impact on the end result.

Mis-specification Caveat

Hierarchical Bayes and marginal likelihood offer very elegant solutions to the model selection problem and they draw an important shadow over cross-validation. However, we still need to be careful with mis-specification issues. This arises when the true model is not considered by the prior, *i.e.*, $p(\theta_{\text{true}}) = 0$. In this case, the posterior cannot converge on the true model and this can be detrimental to the generalization performance of the optimal Bayes predictor. In Grünwald and Langford [2007], using a model of the form $p(y|x, \theta) = p(y|h(x), \eta)$ (refer to Section 2.1.1), they showed that the true risk of the optimal Bayes predictor may diverge significantly from $\min_{h \in \mathcal{H}} R_D(h)$, even when $m \to \infty$. Hence, a final validation or cross-validation step should always be performed on a test set to ensure the good behavior of the learned model.

2.2 The PAC-Bayesian Theory

The PAC-Bayesian theory, despite its name, takes a radically different approach at preventing overfitting. Instead of directly modeling the task to solve, it seeks to bound the true risk of a predictor. Then, by minimizing the bound, we can find the predictor having the best generalization guarantee.

Our goal in this section is to outline some of the main results of the PAC-Bayesian theory to show how it shares some important links with the Bayesian modeling of machine learning. The reader that is interested in this subject should read Lacasse et al. [2006], Germain et al. [2009] and Germain et al. [2011]. Also, Audibert and Bousquet [2007] offers an overview of various learning theory bounds.

While the PAC-Bayesian theory has been generalized to different machine learning setups [Zhang, 2006, Germain et al., 2006], the binary classification paradigm has been widely explored and offers some simplifications. Hence, in this section, we restrict ourselves to $\mathcal{Y} = \{-1, 1\}$ and $\mathcal{L}(y, y') = \mathrm{I}[y \neq y']$, the zero-one loss.

Prior and Posterior

Interestingly, this theory also includes the notion of prior and posterior probabilities over the set $\mathcal{H}$. Since they are not the outcome of Bayes' rule, we use P and Q to distinguish them from their Bayesian analog $p(h)$ and $p(h|S)$ respectively. Note that P is a prior distribution and must obey the same rule as $p(h)$, *i.e.*, it must be defined before observing any sample in S (refer to the proof of Corollary 2.2 of Germain et al. [2009]).

Gibbs Classifier vs Bayes' Classifier

Instead of obtaining bounds on the risk of the deterministic predictors $h \in \mathcal{H}$, the PAC-Bayesian approach produces bounds on a stochastic predictor G_Q called the **Gibbs classifier**, which yields predictions according to the following probability distribution

$$G_Q(y|x) \overset{\text{def}}{=} \sum_{h \in \mathcal{H}} Q(h) \, \mathrm{I}[h(x) = y].$$

In this setup, we have $\mathrm{I}[h(x) = y] = p(y|x, h)$. Hence, we can also write

$$\begin{aligned} G_Q(y|x) &= \sum_{h \in \mathcal{H}} p(h|Q) p(y|x, h) \\ &= p(y|x, Q). \end{aligned}$$

In Section 2.2, we will see that, for a given S, we obtain $Q^\star$, the optimal posterior distribution. Hence, the Gibbs classifier, $p(y|x, Q^\star)$, is the PAC-Bayesian analog of $p(y|x, S)$, the probability of y after model averaging (Section 2.1.2). We note that the Gibbs classifier differs from the **Bayes classifier** B_Q, where

$$B_Q(x) \overset{\text{def}}{=} \underset{y \in \mathcal{Y}}{\operatorname{argmax}} \; G_Q(y|x).$$

This corresponds to the optimal Bayes prediction when using the zero-one loss (Section 2.1.4). While different, these two types of classifiers still behave similarly. For a given x, $B_Q(x)$ will perform a mistake when at least half of the probability mass of Q is erroneous. It follows that the error rate of G_Q is at least half of the error rate of B_Q. Hence, $R_D(B_Q) \leq 2 R_D(G_Q)$. This allows us to directly transform an upper bound on $R_D(G_Q)$ to an upper bound on $R_D(B_Q)$. While this adds a factor of two, the risk of the Bayes classifier is usually smaller [Lacasse et al., 2006] and the factor of two can sometimes be reduced to $1 + \epsilon$, where $0 < \epsilon \ll 1$ [Langford and Shawe-Taylor, 2002].

Catoni's Bound

With these definitions at hand, we can state Catoni's bound [Catoni, 2007], one of the great results of this field. The same theorem is also expressed with great elegance in corollary 2.2 of Germain et al. [2009].

Theorem 2.2.1. *For any distribution D, any set $\mathcal{H}$ of classifiers, any distribution P of support $\mathcal{H}$, any $\delta \in (0,1]$, and any positive real number C, the following inequality holds simultaneously, with probability $1-\delta$, for all distribution Q on $\mathcal{H}$.*

$$R_D(G_Q) \leq F_C\left(CR_S(G_Q) + \frac{1}{m}\left(\mathrm{KL}(Q||P) + \ln\frac{1}{\delta}\right)\right),$$

where $\mathrm{KL}(Q||P) = \mathop{\mathbf{E}}_{h\sim Q} \ln\frac{Q(h)}{P(h)}$ *and* $F_C(x) = \frac{1-e^{-x}}{1-e^{-C}}$.

Before explaining this result, we first perform a few common transformations (which will cost the bound to be looser). First, we use the fact that $R_D(B_Q) \leq 2R_D(G_Q)$ to obtain a bound on Bayes' classifier instead. We also fix δ to 0.05. This means that this probabilistic bound holds with probability 0.95, and $\ln(1/\delta) \lesssim 3$. Also, by noting that $1-e^{-x} \leq x$, we have $F_C(x) \leq x/(1-e^{-C})$. This gives the following probabilistic bound

$$R_D(B_Q) \leq \tfrac{2}{1-e^{-C}}\left(CR_S(G_Q) + \frac{1}{m}\mathrm{KL}(Q||P) + \tfrac{3}{m}\right),$$

where we have grayed out the less important terms.

The term $\mathrm{KL}(Q||P)$ represents the Kullback-Leibler divergence between Q and P. That is, when Q diverges significantly from P, this term becomes more important. In other words, it acts as a regularizer and C is a hyperparameter governing the trade-off between this term and $R_S(G_Q)$, the empirical risk of the Gibbs classifier. The Kullback-Leibler can be seen as a measure of complexity of Q with respect to P. Since it is preceded by $\frac{1}{m}$, as we get more data, we can afford a more complex posterior Q.

One important particularity of this bound is that it holds simultaneously for any posterior Q (which is not the case of the fixed parameter C). This allows us to find $Q^\star$, the posterior that minimizes this bound. Since F_C is a monotonic increasing function, we have

$$Q^\star(h) = \underset{Q}{\mathrm{argmin}}\; CR_S(G_Q) + \frac{1}{m}\mathrm{KL}(Q||P)$$
$$s.t. \sum_{h\in\mathcal{H}} Q(h) = 1$$

By setting the derivative to zero and using Lagrange's multipliers to respect the equality constraint, we find (refer to Lemma 2.1 of Germain et al. [2009])

$$Q^\star(h) \propto P(h)e^{-CmR_S(h)}. \tag{2.6}$$

This probability distribution attributes more mass to the predictors having low empirical risk $R_S(h)$ while taking into account the prior's information. As m or C increase, the distribution becomes sharper around the predictor minimizing the empirical risk and the prior becomes less important. When $m \to \infty$ it becomes a point mass on the one minimizing the true risk (if unique).

Since other Q would be suboptimal, this strongly suggests that model averaging is the best approach to appropriately bound the true risk. To see what happens when we choose a single predictor, we use the linear regression example presented in Section 2.1.3. For simplicity, we use isotropic normal distributions[7] for the prior and posterior, *i.e.*, $P \sim \mathcal{N}(\boldsymbol{\mu}_p, \sigma_p^2 I)$ and $Q \sim \mathcal{N}(\boldsymbol{\mu}_q, \sigma_q^2 I)$. Then, the Kullback-Leibler divergence takes the following form [Rasmussen and Williams, 2006, Appendix A.5]

$$\mathrm{KL}(Q||P) = \frac{1}{2\sigma_p^2}\|\boldsymbol{\mu}_p - \boldsymbol{\mu}_q\|_2^2 + d\log\frac{\sigma_p}{\sigma_q} + \frac{d}{2}\left(\frac{\sigma_q^2}{\sigma_p^2} - 1\right).$$

From this, we see that as $\sigma_q \to 0$, $\mathrm{KL}(Q||P) \to \infty$. In other words, if the posterior distribution is composed of a single predictor, the bound diverges to infinity and gives no guarantee on $R_D(B_Q)$. This again suggests that performing model averaging is the key to avoid overfitting.

To sum up, this bound states that there are three important elements to obtain a good generalization performance:

- Use an appropriate prior.
- Obtain as much data as possible.
- Perform model averaging.

The Missing Noise Model

We saw that Bayesian modeling of machine learning and the PAC-Bayesian theory have several elements in common. Yet, the PAC-Bayesian approach directly works with deterministic predictors and does not explicitly propose a noise model. It instead uses the loss function as a measure of dissimilarity between y_i and $h(x_i)$.

Interestingly, we can show that $\mathcal{L}$ is at the heart of an implicit noise model, where C governs the amount of noise. To achieve this, we use the following noise model

$$p(y|x, h, C) \propto e^{-C\mathcal{L}(y, h(x))}. \tag{2.7}$$

[7]This is also being used in Langford and Shawe-Taylor [2002].

Assuming C is given and using the prior $p(h)$, the posterior becomes

$$\begin{aligned} p(h|S,C) &\propto p(h)\prod_{i=1}^{m} p(y_i|x_i,h,C) \\ &\propto p(h)\prod_{i=1}^{m} e^{-C\mathcal{L}(y_i,h(x_i))} \\ &= p(h)e^{-CmR_S(h)}, \end{aligned}$$

which is identical to the PAC-Bayesian posterior obtained by minimizing Catoni's bound.

For the case of binary classification[8] with the zero-one loss, this corresponds to a Bernoulli noise model, where with probability ζ, the answer $h(x)$ is inverted. Using the shorthand notation $k_y \overset{\text{def}}{=} \mathrm{I}[y \neq h(x)]$, we have

$$\begin{aligned} p(y|x,h,\zeta) &= \zeta^{k_y}(1-\zeta)^{1-k_y} \\ &= e^{\log\left(\zeta^{k_y}(1-\zeta)^{(1-k_y)}\right)} \\ &= e^{k_y\log(\zeta)+(1-k_y)\log(1-\zeta)} \\ &= e^{k_y\log\left(\frac{\zeta}{1-\zeta}\right)}e^{\log(1-\zeta)} \\ &\propto e^{\mathrm{I}[y\neq h(x)]\log\left(\frac{\zeta}{1-\zeta}\right)}. \end{aligned}$$

Hence, by using

$$C = \log\frac{1-\zeta}{\zeta}, \tag{2.8}$$

we recover the noise model of Equation 2.7. In other words, the parameter C introduced in Theorem 2.2.1 corresponds to a reparametrization of the amount of label noise.

[8]This can be generalized to multi-class classification.

Chapter 3

First Paper Presentation

3.1 Details

Bayesian comparison of machine learning algorithms on single and multiple data sets
Alexandre Lacoste, François Laviolette, and Mario Marchand
In *International Conference on Artificial Intelligence and Statistics*, 2012.

3.2 Context

As we have seen, in Section 1.3 and Section 2.1.6, after training a predictor $h = \mathcal{A}(S)$, we should always perform a final evaluation of the generalization performance on a test set $T \sim D^n$. This allows us to get an unbiased estimate of $R_D(h)$ using $R_T(h)$. This measure can then be used to compare the result of different learning algorithms on the same data set. However, this should be used with caution. Given two learning algorithms $\mathcal{A}$ and $\mathcal{B}$ and their resulting predictor $h \overset{\text{def}}{=} \mathcal{A}(S)$ and $g \overset{\text{def}}{=} \mathcal{B}(S)$, it is common to imply:

"If $R_T(h) < R_T(g)$ then h is better than g".

Since T is a random variable, the outcome may differ by using a different sample $T' \sim D^n$. In some cases, $|T|$ may be sufficiently large to support this claim, but it is common malpractice to go further and imply:

"If $R_T(h) < R_T(g)$ then $\mathcal{A}$ is better than $\mathcal{B}$".

Since a learning algorithm is meant to work across different tasks, testing it on a single task is far from being sufficient. Also, many learning algorithms are non-deterministic, *i.e.*, repeating the training phase on the same training set, may yield a different predictor. This problem can even go further. A research scientist will perform several experimentations to explore various techniques. When an experiment provides a negative result, he usually tries another approach until he obtains a good result. In its extreme form, this process is analogous to rolling a die until the desired value is obtained, which has no scientific value and can cause an important bias on the results published in the literature.

3.3 Contributions

The uncertainty of this evaluation process must be dealt with to provide more reproducible experiments and help the machine learning field progress at a reliable pace. To this end, the statistical literature already offers a few statistical significance tests developed for various purposes. Demšar [2006b] performs a survey of these methods and concludes on which one is safe to use for the machine learning community. However, no work has been done on developing a test dedicated to the evaluation of learning algorithms.

In this work, we propose a clear definition of what is meant to have a learning algorithm that is better than another one. The estimation of this answer uses a collection of training set S_i and testing set T_i coming from N different tasks. Based on this, we provide a closed-form solution for computing $\Pr\left(\mathcal{A} \text{ is better than } \mathcal{B} \,\middle|\, \{S_i, T_i\}_{i=1}^{N}\right)$. This model fully integrates out all the uncertainty in the evaluation of learning algorithms by building upon $\Pr\left(R_D(h) < R_D(g)|T\right)$. Using this probabilistic answer, we can choose a threshold τ (e.g., $\tau = 0.9$) to decide which results are significant enough. By doing so, experiments that are based on insufficient data will be rejected as well as experiments comparing algorithms that are almost equivalent. Finally, by using this dedicated test, we are able to outperform the reliability of the tests proposed by Demšar [2006b].

Obviously, we cannot directly address the bias caused by research scientists ignoring their negative results until a positive one occurs. However, the amount of data required to achieve statistical significance helps reducing this problem, *i.e.*, in order to achieve statistical significance on a false result requires many more iterations.

3.4 Comments

This work does not directly discuss the agnostic Bayes approach, the subject of this thesis. However, the term $\Pr\left(R_D(h) < R_D(g)|T\right)$ is at the heart of this idea. In this work, this term is limited to the comparison of only two classifiers and is also limited to the zero-one loss metric. I spent a great amount of effort to generalize this idea to a general loss function with a set $\mathcal{H}$ of size greater than two. Interestingly, the solution came to me while I was working on another problem. In fact, it even took me several months to realize that it was the answer of what I was searching for. This generalization will be presented in the next work, in Chapter 5.

The following work has been adapted from its original version to accommodate the current notation. The main modification has been to use D instead of $\mathcal{D}$ for the task's unknown distribution over $\mathcal{X} \times \mathcal{Y}$ and consequently, we use Dir instead of D to designate the Dirichlet distribution. We also changed $I(a)$ to $\mathrm{I}[a]$ as the indicator function to avoid confusion with the identity matrix I that is often used in this work.

Chapter 4

Bayesian Comparison of Machine Learning Algorithms on Single and Multiple Data sets

4.1 abstract

We propose a new method for comparing learning algorithms on multiple tasks which is based on a novel non-parametric test that we call the Poisson binomial test. The key aspect of this work is that we provide a formal definition for what is meant to have an algorithm that is better than another. Also, we are able to take into account the dependencies induced when evaluating classifiers on the same test set. Finally we make optimal use (in the Bayesian sense) of all the testing data we have. We demonstrate empirically that our approach is more reliable than the sign test and the Wilcoxon signed rank test, the current state of the art for algorithm comparisons.

4.2 Introduction

In this paper, we address the problem of comparing machine learning algorithms using testing data. More precisely, we provide a method that verifies if the amount of testing data is sufficient to support claims such as : "Algorithm $\mathcal{A}$ is better than Algorithm $\mathcal{B}$". This is particularly useful for authors who want to demonstrate that their newly designed learning algorithm is significantly better than some state of the art learning algorithm.

Many published papers simply compare the empirical test risk of the classifiers produced by their learning algorithms. This is insufficient. Since the testing data is randomly sampled from the original task, repeating the experiment might lead to different conclusions. Moreover, when the goal is to compare the generalization performances of learning algorithms, more than one task must be taken into consideration.

In an effort to quantify the differences between learning algorithms, some authors estimate the variance of the risk over the different folds of cross validation. However, since the different training sets are correlated, this violates the independence assumption required by the variance estimator. Moreover, Bengio and Grandvalet [2004, 2005] proved that there is no unbiased variance estimator for k-fold cross validation. To overcome this problem, Dietterich [1998], Alpaydm [1999] developed the 5×2 cross validation which performs an average over a *quasi*-unbiased variance estimator.

Langford [2006] observed that the number of classification errors follows a binomial distribution and proposed a probabilistic testing set bound for the risk of classifiers. A lower and an upper bound on the true risk is then used to determine if the observed difference in empirical testing errors implies that the true risks differ with high confidence. While this non-parametric approach is rigorous and statistically valid, it has low statistical power. Indeed, in practice, the method often claims that there is not enough data to assert any statistical differences.

When the goal is to identify if an algorithm is more suited for general learning than another, both algorithms are analyzed over several data sets. In this situation, some authors propose to average the risk from the different tasks. This is also incorrect. Indeed, consider an algorithm that fails to obtain a good classifier on a task where the risk usually lies around 0.1. This would draw shadow on all the good work this algorithm could perform on tasks having low risk (around 0.01). Thus, we adopt the common viewpoint that the risk is *incommensurable* [Demšar, 2006a] across the different tasks.

To address the incommensurability issue, methods such as the sign test Mendenhall [1983] choose to ignore the magnitude of the difference on each task and simply count how many times the algorithm is better than its competitor on the set of tasks. When more than one learning algorithm is available, the Friedman test Friedman [1937] averages the rank of the classifiers across the different tasks, where each rank is obtained by sorting the empirical test risk of the produced classifiers for a particular task. This metric is able to partially take into account the amount of differences between the risk of classifiers. However, when only two algorithms are compared, it becomes equivalent to the sign test. Alternatively, it is possible to use the Wilcoxon signed rank (WSR) test Wilcoxon [1945]. Instead of simply counting the number of times an algorithm is better than its competitor, like the sign test does, each count is weighted by the rank of the amount of differences in the risk. More precisely, the absolute differences between the empirical test risk of the produced classifiers are sorted across the different tasks. Then the rank of this magnitude is used to weight the counts for both algorithms.

To demystify which methods are appropriate, Demšar Demšar [2006a] performed a benchmark and concludes that non-parametric tests such as the sign test, the WSR test Wilcoxon [1945] and the Friedman test Friedman [1937] are safer to use than methods that assume a normal distribution of the data such as a the t-test or ANOVA. He also concludes that methods assuming some but limited commensurability such as the Friedman test and the WSR test, have more power than the sign test.

Our Contributions

Inspired by the work of Langford Langford [2006], we use the fact that each error performed by a classifier on a test set follows a Bernoulli law. In other words, the *true risk*[1] of a classifier is defined as the probability of performing an error on the given task. Instead of obtaining bounds on the *true risk* like Langford did, we answer a simpler question : "Does classifier h have a smaller *true risk* than classifier g?". It is simpler in the sense that instead of having to estimate two real values, we only have a single binary variable to estimate. Since the test set has only a finite amount of samples, we use the Bayesian methodology to obtain the probability of either outcome. Using this approach, we are also able to take into account the dependencies induced when evaluating classifiers on the same test set.

To be able to compare two learning algorithms on several tasks, we introduce a new concept called a *context*. It represents a distribution over the different tasks a learning algorithm is meant to encounter. Then, each time a task is sampled from the context, whether or not algorithm $\mathcal{A}$ produces a better classifier than algorithm $\mathcal{B}$ follows a Bernoulli law. This means that we do not need to explicitly know the underlying distribution of the context to obtain a probabilistic answer to the following question : "Does algorithm $\mathcal{A}$ have a higher chance of producing a better classifier than algorithm $\mathcal{B}$ in the given context?".

To compare our new methodology to the sign test and the WSR test, we apply the different methods on a wide range of synthetic contexts[2]. Then, an analysis of the false positives and false negatives shows that the newly proposed method consistently outperforms these widely used tests.

The key aspect of this work is that we provide a formal definition for what is meant to have an algorithm that is better than another. Also, we are able to take into account the dependencies induced when evaluating classifiers on the same test set. Finally we make optimal use (in the Bayesian sense) of all the testing data we have. Also, note that all algorithms described in this paper are available as open source software on `http://code.google.com/p/mleval/`.

[1]The *true risk* is the limit value obtained with probability 1 when the size of the test set goes to infinity.

[2]Note that, if one wants to compare these different methodology, the resulting experiments will have to be performed on contexts for which the underlying distribution $\mathcal{W}$ is known, because, otherwise, it is impossible to state which algorithm is "really" better.

4.3 Theoretical Setup

We consider the classification problem. In the single task case, the input space $\mathcal{X}$ is an arbitrary set and the output space $\mathcal{Y} \subset \mathbb{N}$ denotes the set of possible *classes*. An *example* is an input-output pair (x, y) where $x \in \mathcal{X}$ and $y \in \mathcal{Y}$. Throughout the paper we make the usual assumption that each example is drawn according to an unknown distribution D on $\mathcal{X} \times \mathcal{Y}$. A *classifier* is a function $h : \mathcal{X} \longrightarrow \mathcal{Y}$. The *risk* $R_D(h)$ of classifier h on distribution D is defined as $\underset{(x,y)\sim D}{\mathbf{E}} \mathrm{I}[h(x) \neq y]$ where $\mathrm{I}[a] = 1$ if predicate a is true and $\mathrm{I}[a] = 0$ otherwise. We say that classifier h is better than classifier g with respect to D (denoted as $h \overset{D}{\succ} g$) when $R_D(h) < R_D(g)$.

A learning algorithm is a function $\mathcal{A}$ that takes, as input, a set of examples called the *training set* and returns, as output, a classifier. Our goal is to find a metric that determines if an algorithm $\mathcal{A}$ is better than another algorithm $\mathcal{B}$. In order to do so, we assume that the input-output space $\mathcal{X} \times \mathcal{Y}$, the data-generating distribution D on $\mathcal{X} \times \mathcal{Y}$, and the number m of training examples are sampled i.i.d. from an (unknown) distribution $\mathcal{W}$. We refer to $\mathcal{W}$ as being the *context* from which the different data sets are drawn.

Given two learning algorithms, $\mathcal{A}$ and $\mathcal{B}$, and a context $\mathcal{W}$, we define $\tilde{q}_{\mathcal{AB}|\mathcal{W}}$ as the probability that $\mathcal{A}$ is better than $\mathcal{B}$ within the context $\mathcal{W}$, *i.e.*,

$$\tilde{q}_{\mathcal{AB}|\mathcal{W}} \overset{\text{def}}{=} \underset{(D,m)\sim\mathcal{W}}{\mathbf{E}} \; \underset{S\sim D^m}{\mathbf{E}} \; I\left[\mathcal{A}(S) \overset{D}{\succ} \mathcal{B}(S)\right] \tag{4.1}$$

Consequently, we say that $\mathcal{A}$ is better than $\mathcal{B}$ within the context $\mathcal{W}$ iff $\tilde{q}_{\mathcal{AB}|\mathcal{W}} > 1/2$, *i.e.*,

$$\mathcal{A} \overset{\mathcal{W}}{\succ} \mathcal{B} \quad \Leftrightarrow \quad \tilde{q}_{\mathcal{AB}|\mathcal{W}} > \tfrac{1}{2} \tag{4.2}$$

The definition of $\tilde{q}_{\mathcal{AB}|\mathcal{W}}$ is simple but, in practice, we do not observe $\mathcal{W}$ nor any of the sampled D. It is thus not possible to directly determine if $\mathcal{A} \overset{\mathcal{W}}{\succ} \mathcal{B}$. However, we are able to provide a probabilistic answer by using Bayesian statistics.

4.4 Bounding the Risk

We cannot evaluate the true risk of a classifier h with respect to D. However, using a test set $T \sim D^n$, we can evaluate a probability distribution over the possible values of $R_D(h)$.

This is done by first observing that $k_h \overset{\text{def}}{=} nR_T(h)$ follows a binomial law of parameter $p \overset{\text{def}}{=} R_D(h)$. Next, we will use Bayes' theorem to obtain a posterior distribution over p. But first, we need to provide a prior distribution.

Since the beta distribution is the conjugate prior of the binomial distribution, it is wise to use it for the prior over p, with parameters $\alpha' > 0$ and $\beta' > 0$:

$$B(p; \alpha', \beta') \overset{\text{def}}{=} \tfrac{\Gamma(\alpha'+\beta')}{\Gamma(\alpha')\Gamma(\beta')} p^{\alpha'-1}(1-p)^{\beta'-1},$$

where Γ denotes the gamma function. It follows from Bayes' theorem, that the posterior will also be a beta distribution, but this time, with parameters $\alpha \overset{\text{def}}{=} \alpha' + k_h$ and $\beta \overset{\text{def}}{=} \beta' + n - k_h$.

To complete the choice of the prior, we still need to specify the values for α' and β'. This could be achieved by using some information coming from the training set or some knowledge we might have about the learning algorithm. However, to stay as neutral as possible as advocated by Jaynes Jaynes [1957], we chose the least informative prior—which is the uniform prior given by $\alpha' = \beta' = 1$.

Now that we have a probability distribution over $R_D(h)$, we can use the cumulative distribution, to obtain the probability that the risk is smaller than x :

$$B_c(x; \alpha, \beta) \overset{\text{def}}{=} \int_0^x B\left(t; \alpha, \beta\right) dt$$

From the nature of $B(\,\cdot\,; \alpha, \beta)$, $B_c(\,\cdot\,; \alpha, \beta)$ is a one to one relation. Therefore, it has an inverse that we denote by $B_c^{-1}(\,\cdot\,; \alpha, \beta)$.

Using this inverse function, we can now obtain a probabilistic upper bound of the risk

$$\Pr\Big(\; R_D(h) \;\leq\; B_c^{-1}(1-\delta; \alpha, \beta) \;\Big) \geq 1-\delta \tag{4.3}$$

where δ is the probability that our bound fails and is commonly set to small values such as 0.05. Since the bound also depends on δ, smaller values loosen the bound while higher values tighten the bound.

When $\alpha' = 1$ and in the limit where $\beta' \to 0$, we converge to the bound described in Theorem 3.3 of Langford's tutorial Langford [2006]. This can be shown using the following identity Abramowitz and Stegun [1964]

$$\sum_{i=0}^{k} \binom{n}{i} p^i \left(1-p\right)^{n-i} = 1 - B_c\left(p, k+1, n-k\right), \tag{4.4}$$

to rewrite the Langford's test-set bound as follows : $\Pr\left(\; R_D(h) \;\leq\; B_c^{-1}(1-\delta; k_h+1, n-k_h) \;\right) \geq 1–\delta$. Consequently, the probabilistic risk upper bound of Equation (4.3), given by the Bayesian posterior, becomes Langford's test-set bound when the prior has all its weight on $R_D(h) = 1$ (*i.e.*, when we assume the worst). Finally, a numerical evaluation of B_c^{-1} can be computed using a Newton method and is available in most statistical software.

4.5 Probabilistic Discrimination of Learning Algorithms on a Single Task

Let $\mathcal{T}$ be any fixed classification task, or equivalently, let D be any fixed (but unknown) probability distribution on $\mathcal{X} \times \mathcal{Y}$. Let also $\mathcal{A}$ and $\mathcal{B}$ be two learning algorithms. In this section, we want to evaluate, for the task $\mathcal{T}$, the probability that a classifier obtained using $\mathcal{A}$ is better than a classifier obtained using $\mathcal{B}$.

In the previous section, we saw that, using a test set $T \sim D^n$ we can obtain a probability distribution over $R_D(\mathcal{A}(S))$. Similarly, with a second test set $T' \sim D^{n'}$, we can obtain a probability distribution over $R_D(\mathcal{B}(S'))$. When both training sets and both test sets are independent, the joint distribution over $R_D(\mathcal{A}(S))$ and $R_D(\mathcal{B}(S'))$ is simply the product of the individual distributions. From this joint distribution, we can therefore evaluate $\Pr\left(\mathcal{A}(S) \overset{D}{\succ} \mathcal{B}(S')\right)$ by integrating the probability mass in the region where $R_D(\mathcal{A}(S)) < R_D(\mathcal{B}(S'))$.

However, in most experiments, learning algorithms are sharing the same training set and the same testing set. To take these dependencies into account, let us exploit the fact that the joint error of the pair of classifiers comes from a multinomial distribution.

More precisely, let $h \overset{\text{def}}{=} \mathcal{A}(S)$ and $g \overset{\text{def}}{=} \mathcal{B}(S)$ where $S \sim D^m$. For each sample $(x, y) \sim D$, we have the following three outcomes: (1) $h(x) \neq y \wedge g(x) = y$; (2) $h(x) = y \wedge g(x) \neq y$; (3) $h(x) = g(x)$ with probability p_h, p_g and $\overline{p}$ respectively. Moreover, $p_h + p_g + \overline{p} = 1$. Let k_h, k_g, and $\overline{k}$, be the respective counts of these events on a testing set T. Then $(k_h, k_g, \overline{k})$ follows a multinomial law with parameters $(|T|, (p_h, p_g, \overline{p}))$.

Since $R_D(h) < R_D(g)$ whenever $p_h < p_g$, our aim is to obtain a probability distribution over $(p_h, p_g, \overline{p})$, given $(k_h, k_g, \overline{k})$, to be able to integrate the probability mass in the region where $p_h < p_g$. This can be done using Bayes' theorem whenever we have a prior distribution.

Since the Dirichlet distribution is the conjugate prior of the multinomial, it is wise to use it for the prior information we have about p_h, p_g and $\overline{p}$. The Dirichlet distribution of parameters $(\alpha_h, \alpha_g, \overline{\alpha})$ is defined as

$$\mathrm{Dir}\left(p_h, p_g, \overline{p}; \alpha_h, \alpha_g, \overline{\alpha}\right) \overset{\text{def}}{=} \tfrac{\Gamma(\alpha_h+\alpha_g+\overline{\alpha})}{\Gamma(\alpha_h)\Gamma(\alpha_g)\Gamma(\overline{\alpha})} p_h^{\alpha_h-1} p_g^{\alpha_g-1} \left(1-p_g-p_h\right)^{\overline{\alpha}-1}$$

where $\alpha_h > 0$, $\alpha_g > 0$ and $\overline{\alpha} > 0$.

If the prior is a Dirichlet with parameters $\left(\alpha'_h, \alpha'_g, \overline{\alpha}'\right)$, it follows from Bayes' law that, after observing k_h, k_g and $\overline{k}$ on the testing set, the posterior is also a Dirichlet with parameters $(\alpha_h, \alpha_g, \overline{\alpha})$ where $\alpha_h \overset{\text{def}}{=} \alpha'_h + k_h$, $\alpha_g \overset{\text{def}}{=} \alpha'_g + k_g$ and $\overline{\alpha} \overset{\text{def}}{=} \overline{\alpha}' + \overline{k}$. Consequently, the following theorem gives us the desired result for $\Pr\left(h \overset{D}{\succ} g\right)$.

Theorem 4.5.1. *Let $\alpha_h \overset{\text{def}}{=} \alpha'_h + k_h$, $\alpha_g \overset{\text{def}}{=} \alpha'_g + k_g$ and $\overline{\alpha} \overset{\text{def}}{=} \overline{\alpha}' + \overline{k}$, where $\alpha'_g > 0$, $\alpha'_h > 0$, $\overline{\alpha}' > 0$, then*

$$\begin{aligned} &\Pr\left(h \overset{D}{\succ} g\right) \\ &= \int_0^1 \int_0^{\frac{1-\overline{p}}{2}} \mathrm{Dir}\left(p_g, p_h, \overline{p}\ ;\ \alpha_g, \alpha_h, \overline{\alpha}\right)\ dp_h\ d\overline{p} \\ &= B_c\left(\tfrac{1}{2}; \alpha'_h + k_h, \alpha'_g + k_g\right) \end{aligned}$$

Proof. The first equality follows from the explanations above. Now, using $C \overset{\text{def}}{=} \frac{\Gamma(\alpha_h+\alpha_g+\overline{\alpha})}{\Gamma(\alpha_h)\Gamma(\alpha_g)\Gamma(\overline{\alpha})}$, $\gamma \overset{\text{def}}{=} 1 - \overline{p}$ and $z \overset{\text{def}}{=} \frac{p_h}{\gamma}$, we have :

$$\begin{aligned}
&\int_0^1 \int_0^{\frac{1-\overline{p}}{2}} \mathrm{Dir}\left(p_g, p_h, \overline{p}\ ;\ \alpha_g, \alpha_h, \overline{\alpha}\right)\ dp_h\ d\overline{p} \\
&= C \int_0^1 \overline{p}^{\overline{\alpha}-1} \int_0^{\frac{1-\overline{p}}{2}} p_h^{\alpha_h-1} \left(1-\overline{p}-p_h\right)^{\alpha_g-1}\ dp_h\ d\overline{p} \\
&= C \int_0^1 \overline{p}^{\overline{\alpha}-1} \int_0^{\frac{1}{2}} (\gamma z)^{\alpha_h-1} \left(\gamma-\gamma z\right)^{\alpha_g-1} \gamma\ dz\ d\overline{p} \\
&= C \int_0^1 \overline{p}^{\overline{\alpha}-1} \gamma^{\alpha_h+\alpha_g-1}\ d\overline{p} \int_0^{\frac{1}{2}} z^{\alpha_h-1} \left(1-z\right)^{\alpha_g-1}\ dz \\
&= \frac{\Gamma(\alpha_h+\alpha_g)}{\Gamma(\alpha_h)\Gamma(\alpha_g)} \int_0^{\frac{1}{2}} z^{\alpha_h-1} \left(1-z\right)^{\alpha_g-1}\ dz \\
&\overset{\text{def}}{=}\ B_c\left(\tfrac{1}{2}; \alpha'_h + k_h, \alpha'_g + k_g\right)
\end{aligned}$$

$\square$

4.5.1 About the Prior

From Theorem 4.5.1, we see that $\Pr\left(h \overset{D}{\succ} g\right)$ does not depend on $\overline{k}$ nor $\overline{\alpha}'$. However, to complete the choice of the prior distribution, we still need to provide values for α_h and α_g. It might be possible to extract information from the training set using cross-validation. However, this approach is not straightforward and will require further investigation in future work. Also, we should not introduce favoritism by using an imbalanced prior. Hence, one should use $\alpha_h = \alpha_g \overset{\text{def}}{=} \widetilde{\alpha}$. This leaves us with only one parameter, $\widetilde{\alpha}$, for the prior. Using $\widetilde{\alpha} > 1$ is equivalent to supposing, *a priori*, that both classifiers are similar. On the opposite, using $0 < \widetilde{\alpha} < 1$ is equivalent to supposing, *a priori*, that both classifiers are different. Since they are no evidences supporting the choice of one over the other, we follow Jaynes' maximum entropy principle Jaynes [1957] and we use $\widetilde{\alpha} = 1$, i.e., $\alpha_h = \alpha_g = 1$, which represents the uniform distribution.

4.6 The Poisson Binomial Test

In this section we generalize the results of the previous section to contexts. In context $\mathcal{W}$, whether or not algorithm $\mathcal{A}$ outputs a better classifier than algorithm $\mathcal{B}$ is a Bernoulli random variable of parameter $\tilde{q}_{\mathcal{AB}|\mathcal{W}}$ (Equation (4.1)). Therefore, after observing N data sets, the number of wins of $\mathcal{A}$ over $\mathcal{B}$ is a binomial distribution of parameter $\tilde{q}_{\mathcal{AB}|\mathcal{W}}$ and N.

Knowing the number of wins of $\mathcal{A}$ on N trials would allow us to directly integrate the Beta distribution to evaluate the probability that $\mathcal{A} \overset{\mathcal{W}}{\succ} \mathcal{B}$. However, since we only have a probabilistic answer when discriminating learning algorithms on a single task, we need to take the expectation over the different number of wins.

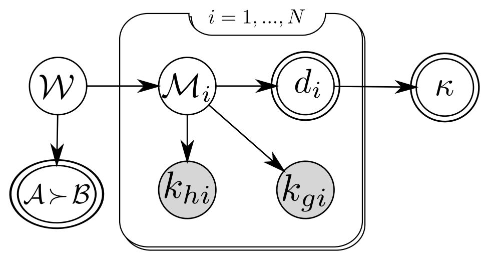


Figure 4.1: A graphical model representing the dependencies between the variables during the process of evaluating algorithms on multiple data sets.

Putting this more formally, comparing $\mathcal{A}$ and $\mathcal{B}$ on data set i yields the multinomial $\mathcal{M}_i$ of parameters p_{hi}, p_{gi} and $|T_i|$. As seen in Section 4.5, this multinomial is hiding the process of comparing h_i and g_i, trained on S_i and tested on T_i. Let k_i be the observations obtained on the i^{th} test set (e.g., $k_i = (k_{hi}, k_{gi})$). To simplify equations, we also define $d_i \overset{\text{def}}{=} I\left(p_{hi} < p_{gi}\right)$, $\kappa \overset{\text{def}}{=} \sum_i d_i$, $\mathbf{k} \overset{\text{def}}{=} (k_1, k_2, ..., k_N)$, $\mathbf{d} \overset{\text{def}}{=} (d_1, d_2, ..., d_N)$, and $r \overset{\text{def}}{=} \tilde{q}_{\mathcal{AB}|\mathcal{W}}$. As represented in Figure 4.1 the only observed variables are the ones in $\mathbf{k}$.

Consequently, we have :

$$
\begin{aligned}
\Pr\left(\mathcal{A} \overset{\mathcal{W}}{\succ} \mathcal{B} \;\middle|\; \mathbf{k}\right) &= \int_{\frac{1}{2}}^{1} p\left(r \mid \mathbf{k}\right) dr \\
&= \int_{\frac{1}{2}}^{1} \sum_{\kappa=0}^{N} p\left(r, \kappa \mid \mathbf{k}\right) dr \\
&= \sum_{\kappa=0}^{N} \int_{\frac{1}{2}}^{1} p\left(r \mid \kappa\right) dr \Pr\left(\kappa \mid \mathbf{k}\right) \\
&= \sum_{\kappa=0}^{N} \Pr\left(\kappa \mid \mathbf{k}\right) B_c\left(\tfrac{1}{2}, N-\kappa+\widehat{\beta}_{\mathcal{B}}, \kappa+\widehat{\beta}_{\mathcal{A}}\right)
\end{aligned}
$$

where we have used the beta distribution with parameter $\widehat{\beta}_{\mathcal{A}}$ and $\widehat{\beta}_{\mathcal{B}}$ for the prior distribution over r and, using Bayes' theorem, have obtained a posterior also given by a beta distribution. To avoid favoritism over $\mathcal{A}$ or $\mathcal{B}$, it is crucial to use $\widehat{\beta}_{\mathcal{A}} = \widehat{\beta}_{\mathcal{B}} \overset{\text{def}}{=} \widehat{\beta}$. Also, to maximize entropy, we use $\widehat{\beta} = 1$.

We are now left with one term to evaluate:

$$\begin{aligned}
&\Pr\left(\kappa \mid \mathbf{k}\right)\\
&= \sum_{\mathbf{d}\in\{0,1\}^N} \Pr\left(\kappa \mid \mathbf{d}\right) \Pr\left(\mathbf{d} \mid \mathbf{k}\right)\\
&= \sum_{\mathbf{d}\in\{0,1\}^N} \Pr\left(\kappa \mid \mathbf{d}\right) \prod_{i=1}^{N} \Pr\left(d_i \mid k_i\right)\\
&= \sum_{\mathbf{d}\in\{0,1\}^N} I\left(\sum_i d_i = \kappa\right) \prod_{i=1}^{N} p_i^{d_i}(1-p_i)^{1-di} \qquad (4.5)
\end{aligned}$$

where $p_i \overset{\text{def}}{=} \Pr\left(h_i \overset{D}{\succ} g_i\right)$ is given by Theorem 4.5.1. In this form, Equation (4.5) is computationally hard to track. However, this represents a Poisson-binomial distribution Wang [1993], which is the generalization of the binomial distribution when the Bernoulli trials have different probabilities. There exist several ways to compute such probability distribution Fernandez and Williams [2010], Chen et al. [1994], Chen and Liu [1997]. In our case, we use the following dynamic programming algorithm. We have $\Pr(\kappa) = q_N(\kappa)$ where $q_i(\kappa)$ is defined recursively as:

$$q_i(\kappa) \overset{\text{def}}{=} \begin{cases} 1 & \text{if } i = 0 \wedge \kappa = 0 \\ 0 & \text{if } \kappa < 0 \vee \kappa > i \\ p_i q_{i-1}(\kappa - 1) \\ \quad +(1-p_i)q_{i-1}(\kappa) & \text{otherwise} \end{cases}$$

This algorithm has $O(N^2)$ complexity. It is possible to build a $O\left(N\log^2(N)\right)$ algorithm, using spectral domain convolution to combine the solution of a divide and conquer strategy. But the $O\left(N^2\right)$ algorithm is simpler and fast enough for our needs.

4.6.1 Transitivity and Ordering

The operator $\overset{D}{\succ}$ is transitive for any D since it is based on the risk of the classifiers. We then have $h \overset{D}{\succ} g \wedge g \overset{D}{\succ} f \implies h \overset{D}{\succ} f$. However, this transitive property doesn't hold for $\overset{\mathcal{W}}{\succ}$. This can be shown with a small counterexample presented on Figure 4.2. In this example, we are comparing algorithms $\mathcal{A}$, $\mathcal{B}$ and $\mathcal{C}$ on data sets a, b and c. We work in the theoretical setup. Therefore we know $\mathcal{W}$ and assume that it is the uniform distribution on the 3 data sets. Also, the exact risk for each classifier is expressed in Figure 4.2. Hence, from Definition (4.2), we have $\mathcal{A} \overset{\mathcal{W}}{\succ} \mathcal{B}$, $\mathcal{B} \overset{\mathcal{W}}{\succ} \mathcal{C}$ and $\mathcal{C} \overset{\mathcal{W}}{\succ} \mathcal{A}$. This implies that $\overset{\mathcal{W}}{\succ}$ cannot be used for ordering algorithms.

The non-transitivity of $\overset{\mathcal{W}}{\succ}$ is a consequence of ignoring the amount of differences between the two risks. This also affects other methods such as the sign test (defined in Section 4.7.1). In most practical situations, the finite amount of samples have the side effect of weighting the differences between classifiers. Thus, alleviating the chances of observing a high probability cycle.

	$\mathcal{A}$	$\mathcal{B}$	$\mathcal{C}$
a	0.1	0.2	0.3
b	0.3	0.1	0.2
c	0.2	0.3	0.1

Figure 4.2: A minimalist counterexample on the transitivity of comparison on multiple data sets.

4.7 Simulations With Synthetic Contexts

In this section we compare the performances of the Poisson binomial test with the sign test and the WSR test. Ideally, we would run learning algorithms with the different evaluation methods on contexts for which we know the real underlying distributions and we would repeat this process several times. However, we do not have such data and this process would take forever. Fortunately, we saw that the outcome of testing two learning algorithms on a task is equivalent to sampling from a multinomial of parameters p_g, p_h, $\overline{p}$, and $n \overset{\text{def}}{=} |T|$. Therefore, by defining a probability distribution over the four parameters of this multinomial, we build what we call a *synthetic context*.

With those contexts, it is now easy to sample the data we need to compare the answer of the different methods with the real answer. However, the behavior of the three methods depends on a confidence threshold which offer a trade-off between the success rate and the error rate. It is thus hard to compare them with a fixed threshold. To overcome this difficulty, we use the area under the ROC curve (AUC) as a metric of performances.

The following subsections provide the details on how to obtain the p-values for the sign test and the WSR test and on how to compute the AUC. Finally, in Section 4.7.4, we present the results obtained on different synthetic contexts.

4.7.1 Sign Test

The sign test Mendenhall [1983] simply counts the number of times that $\mathcal{A}$ has a better empirical metric than $\mathcal{B}$ and assumes that this comes from a binomial distribution of parameters $\tilde{q}_{\mathcal{AB}|\mathcal{W}}$ and N. In our case, we use $\operatorname{sgn}(k_{hi} - k_{gi})$ and any sample i where $k_{hi} = k_{gi}$ is ignored, for $i \in \{1, 2, ..., N\}$.

To assert that the observed difference is significant enough, the sign test is based on hypothesis testing. In our case, this corresponds to whether $\mathcal{A} \overset{\mathcal{W}}{=} \mathcal{B}$ or not. More formally, $H_0 : \tilde{q}_{\mathcal{AB}|\mathcal{W}} = 0.5$ and $H_1 : \tilde{q}_{\mathcal{AB}|\mathcal{W}} \neq 0.5$. Let κ_+ be the number of times we observe a positive value of $\operatorname{sgn}(k_{gi} - k_{hi})$ and κ_- be the number of times we observe a negative value. Then, the two tailed p-value of H_0 is given by

$$\rho_s(\kappa_+, \kappa_-) = 2 \sum_{i=0}^{\min(\kappa_+,\kappa_-)} \binom{\kappa_+ + \kappa_-}{i} \frac{1}{2^{\kappa_+ + \kappa_-}}$$

It is common to reject H_0 when $\rho_s(\kappa_+, \kappa_-)$ goes below 0.05 (or any reasonable threshold). In that case, we proceed as follows : if $\kappa_+ > \kappa_-$, we conclude that $\mathcal{A} \overset{\mathcal{W}}{\succ} \mathcal{B}$, and we conclude the converse when $\kappa_+ < \kappa_-$.

4.7.2 Wilcoxon Signed Rank (WSR) Test

Instead of simply counting the number of wins like the sign test does, the WSR test Wilcoxon [1945] weight each count by the rank of the absolute difference of the empirical risk. More precisely, the empirical risk difference is

$$d_i \overset{\text{def}}{=} \frac{k_{gi} - k_{hi}}{m_i}.$$

The samples where $d_i = 0$ are rejected and we use J as the set of indices where $d_i \neq 0$. To take into account the fact that there might be samples of equal rank, we use the following formula to compute the rank :

$$r_i \overset{\text{def}}{=} \frac{1}{2}\left(1 + \sum_{j \in J} I\left(|d_j| < |d_i|\right) + \sum_{j \in J} I\left(|d_j| \leq |d_i|\right)\right)$$

The sum of positive rank is $c_+ \overset{\text{def}}{=} \sum_{j \in J} r_j I\left(d_j > 0\right)$ and the sum of negative rank is $c_- \overset{\text{def}}{=} \sum_{j \in J} r_j I\left(d_j < 0\right)$.

The WSR test assumes that the d_i are sampled i.i.d. from a symmetric distribution around a common median[3]. Then, under H_0, the values of $I\left(d_i < 0\right)$ are i.i.d. and have equal probabilities for either outcome. This allows us to recursively compute the probability distribution for the values of c_+ under H_0 as follows :

$$w_n(c) \overset{\text{def}}{=} \begin{cases} 0 & \text{if } c \notin [0, \frac{n(n+1)}{2}] \\ 1 & \text{if } n = 1 \text{ and } c \in [0, \frac{n(n+1)}{2}] \\ \frac{w_{n-1}(c) + w_{n-1}(c-n)}{2} & \text{otherwise} \end{cases}$$

Finally, the two tailed p-value is given by :

$$\rho_w(c_+, c_-) \overset{\text{def}}{=} 2 \sum_{c=0}^{\min(c_+, c_-)} w_{|J|}(c)$$

and whenever $\rho_w(c_+, c_-) \leq \delta$, we reject H_0. In that case, we proceed as follows : if $c_+ > c_-$, we conclude that $\mathcal{A} \overset{\mathcal{W}}{\succ} \mathcal{B}$, and conclude the converse when $c_+ < c_-$.

[3]We will see that the symmetric assumption is inappropriate for machine learning algorithm evaluation.

4.7.3 Area Under the ROC Curve

With a synthetic context, we can directly sample a set of k_{hi} and k_{gi} for $i \in \{1, 2, \ldots, N\}$. Providing this information to one of the methods, we obtain an answer $\hat{a}$ and a confidence level γ. In the case of the sign test, the confidence level is $1-\rho_s(\kappa_+, \kappa_-)$. For the WSR test, we use $1-\rho_w(c_+, c_-)$. Finally, for the Poisson binomial test, we use $\max\left(\Pr\left(\mathcal{A} \overset{\mathcal{W}}{\succ} \mathcal{B}\right), \Pr\left(\mathcal{B} \overset{\mathcal{W}}{\succ} \mathcal{A}\right)\right)$.

Next, repeating this experiment M times for a given threshold τ and comparing the answer to the real answer a, we obtain a success count s_τ and an error count e_τ. More formally,

$$\begin{aligned} s_\tau &\overset{\text{def}}{=} \textstyle\sum_{j=1}^{M} \mathrm{I}[\gamma_j > \tau]\, \mathrm{I}[\hat{a}_j = a] \\ e_\tau &\overset{\text{def}}{=} \textstyle\sum_{j=1}^{M} \mathrm{I}[\gamma_j > \tau]\, \mathrm{I}[\hat{a}_j \neq a]. \end{aligned}$$

To obtain the ROC curve, we have computed all pairs (s_τ, e_τ) by selecting τ from the set of the M obtained confidence levels γ_j. Next, to obtain the AUC, we use a trapezoidal approximation of the integral over the values $\left(\frac{s_\tau}{s_0}, \frac{e_\tau}{e_0}\right)$ where s_0 and e_0 correspond to the success count and error count when the threshold is at its minimum value.

To obtain a good resolution for the ROC curve, we fix M to 10^5. Also, to make sure that there is no bias in the methods, we randomly swap the k_{hi} and k_{gi} and, we adjust the value of a accordingly.

4.7.4 Experimental results

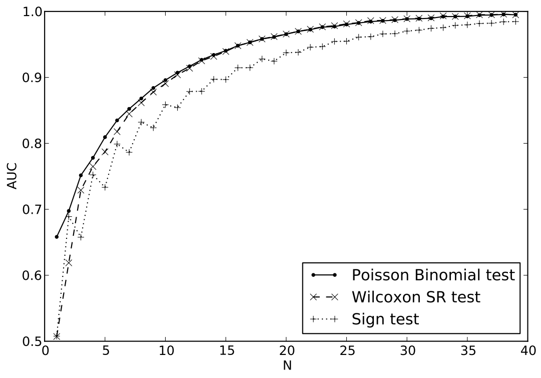


Figure 4.3: Comparison of the 3 methods, using AUC on a single Dirichlet synthetic context, for various values of N where n is fixed to 1001.

As explained before, a synthetic context is completely determined by a probability distribution over the parameters of the three outcome multinomial. For simplicity reasons, we fix n, and use Dirichlet distributions to represent synthetic contexts.

For our first experiment, we fix n to 1001 and use the Dirichlet distribution of parameters $(100, 110, 790)$. The expected values for p_h, p_g and $\overline{p}$ are then respectively 0.1, 0.11 and

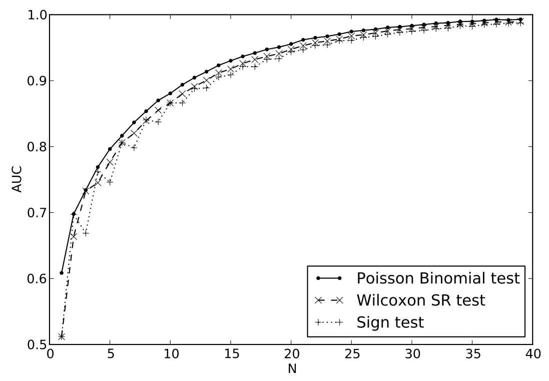


Figure 4.4: Comparison of the 3 methods, using AUC on a multiple Dirichlet synthetic context, for various values of N where n is fixed to 1001.

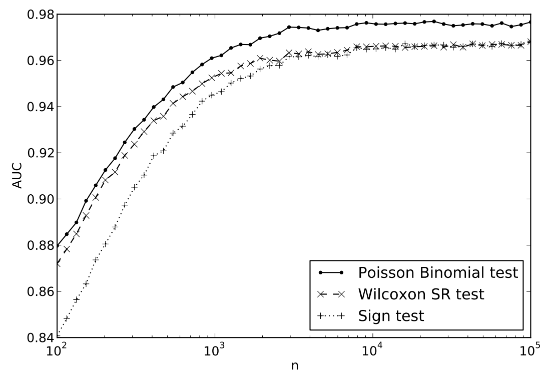


Figure 4.5: Comparison of the 3 methods, using AUC on a multiple Dirichlet synthetic context, for various values of n where N is fixed to 21.

0.79. This means that in this context $\mathcal{A} \overset{\mathcal{W}}{\succ} \mathcal{B}$. Figure 4.3 expresses the AUC of the three different methods, for various values of N. From those results, we first observe that the Poisson binomial test constantly outperform the sign test. Also, we observe that the WSR test have performances similar to our approach. However, it is important to understand that this synthetic context is favorable to the WSR test. Indeed, the d_i (see Section 4.7.2) are sampled from a symmetric distribution, which is one of the assumptions required by the WSR test.

To explore how the WSR test behaves with a non-symmetric distribution of the d_i, we use the following bimodal context : with probability $\frac{2}{3}$, we sample from the Dirichlet distribution of parameters $(100, 140, 9760)$, otherwise we sample from the Dirichlet distribution of parameters $(1400, 1000, 7600)$. Now, fixing n to 100001 and N to 14, we have an AUC over 0.8 for the sign test and the Poisson binomial test, while the AUC of the WSR test is 0.334, i.e., worse than random. This means that the WSR test may drastically fail in some situations. Such events can occur in practice when a learning algorithm is better than its competitor on tasks having many classes, but the two algorithms are exposed to a context where tasks having fewer classes are more frequent. Since the average risk typically raises with the number of classes, this would create an asymmetrical distribution of the empirical risk differences.

Finally, to build a more plausible context, we use the error counts obtained when comparing svm with parzen window on 22 data sets coming from UCI and MNIST (Section 4.10.1). The synthetic context is thus a uniform distribution over 22 Dirichlet distributions where the parameters are provided in supplementary materials. The AUC for various values of N and various values of n are expressed in Figure 4.4 and Figure 4.5. From those results, we conclude that the Poisson binomial test is a significantly more appropriate test for machine learning algorithm comparison.

4.8 Comparing Popular Algorithms

Table 4.1: Comparison of the test risk for 3 of the 4 algorithms on 5 of the 22 data sets. Individual significance is shown, using Theorem 4.5.1, for selected pair of classifiers.

	svm	$\overset{D_i}{\succ}$	**ann**	$\overset{D_i}{\succ}$	**parzen**
Adult	0.157	0.52	0.157	1.00	0.172
Glass	0.140	0.61	0.150	0.40	0.140
MNIST:08	0.003	1.00	0.012	0.04	0.006
Mushrooms	0.000	0.50	0.000	0.99	0.001
Sonar	0.154	0.84	0.202	0.42	0.192

Table 4.2: The pairwise Poisson binomial test showing $\Pr(\text{row} \succ \text{col})$. Gray values represent redundant information.

	svm	**ann**	**parzen**	**adaBoost**
svm	0.50	0.72	0.99	0.95
ann	0.28	0.50	0.88	0.87
parzen	0.01	0.12	0.50	0.52
adaBoost	0.05	0.13	0.48	0.50

Table 4.3: The pairwise sign test for 1 - $\rho_s(\kappa_+, \kappa_-)$. Cases where $\kappa_+ \not> \kappa_-$ are omitted.

	svm	**ann**	**parzen**	**adaBoost**
svm	-	0.88	1.00	0.92
ann	-	-	0.71	0.83
parzen	-	-	-	-
adaBoost	-	-	0.18	-

Table 4.1 presents the empirical test risk $R_{T_i}(\mathcal{A}_j(S_i))$ for three popular learning algorithms on five UCI data sets. A more complete table is provided in the supplementary material. Table 4.2 expresses the pairwise comparison of algorithms when using the Poisson binomial

Table 4.4: The pairwise WSR test for 1 - $\rho_w(c_+, c_-)$. Cases where $c_+ \not\succ c_-$ are omitted.

	svm	**ann**	**parzen**	**adaBoost**
svm	-	0.30	0.97	0.92
ann	-	-	0.95	0.41
parzen	-	-	-	-
adaBoost	-	-	-	-

test. For comparison, we have also reported the p-value of the sign test in Table 4.3 and for the WSR test in Table 4.4. Here, the three methods yield comparable results but, in contrast with the experiment in Section 4.7, it is not possible to conclude if one method yields better conclusions than the others since the distribution $\mathcal{W}$ is unknown here.

When performing experiments on synthetic contexts, we observed that using a threshold of 0.85 for the Poisson binomial test yield a lower type I error rate than the sign test and the WSR test with a threshold of 0.90 on the confidence level. Using this remark, the sign test perform 2 assertions, the WSR test perform 3 assertions and, the Poisson binomial test perform 4 assertions. While it seems favorable to our approach, another test might have yield different results. However, the results provided in Section 4.7 are reliable and support the same conclusion.

4.9 Future Work

In this work we did not addressed the problem of multiple comparisons. This case occurs when more than one probabilistic comparison has to be made. Then, the probability of having a false conclusion inrcreases with the number of comparisons. To address this problem, it is common to use methods to control the familywise error rate Hochberg and Tamhane [1987] or the false discovery rate Benjamini and Hochberg [1995]. However, the current approaches are made to work with frequentist comparisons and are not directly applicable to our Bayesian tool. To this end, we are currently working on a Bayesian method to control the false discovery rate under dependencies.

We are also investigating on how to extend the Poisson binomial test to work outside of the classification paradigm. This would allow us to compare algorithms meant to work on regression or structured output tasks.

Together, these new methods should provide the machine learning community with a wide range of tools to shed light on the discovery of new learning algorithms.

Acknowledgement

Thanks to Pascal Germain, Alexandre Lacasse, Jean-Francis Roy, and Francis Turgeon-Boutin for their help in the software implementation. Thanks to CLUMEQ for providing support and access to Colosse's high performance computer grid. Work supported by NSERC Discovery Grants 122405 (M. M.) and 262067 (F. L.).

4.10 APPENDIX—SUPPLEMENTARY MATERIAL

4.10.1 Synthetic Contexts

Table 4.5: List of Dirichlet parameters used in the multimodal synthetic context.

kh+1	**kg+1**	**n-kh-kg+1**
11	14	1223
12	20	1232
10	16	4078
13	15	1228
31	17	320
22	22	148
27	35	154
14	18	112
17	25	286
24	17	151
149	240	5555
16	30	762
19	14	791
11	16	237
79	127	3834
15	20	1213
10	11	803
15	15	122
11	29	176
14	18	352
46	64	410
31	1019	2694

4.10.2 Experimentation details

In this section, we compare the following commonly-used learning algorithms:

- **svm:** Support Vector Machine Cortes and Vapnik [1995] with the RBF kernel.
- **parzen:** Parzen Window Parzen [1962], Rosenblatt [1956] with the RBF kernel.
- **adaBoost:** AdaBoost Freund and Schapire [1995] using stumps as weak classifiers.
- **ann:** Artificial Neural Networks Minsky and Papert [1969] with two hidden layers, sigmoid activation function and L^2 regularization of the weights.

These learning algorithms are compared on 18 binary classification data sets coming from UCI and 4 other data sets coming from MNIST. Our goal is to explore the behavior of the Poisson binomial test on commonly-used learning algorithms applied on real data sets rather than discriminating the quality of these learning algorithms. The details on the tuning of these algorithms are provided in the appendix.

Table 4.6: Comparison of $R_{T_i}(\mathcal{A}_j(S_i))$ for $i \in \{1, 2, ..., 22\}$ and $j \in \{1, 2, 3, 4\}$. Individual significance is shown, using Theorem 4.1, for selected pair of classifiers.

	svm	$\succ$	**ann**	$\succ$	**parzen**	$\succ$	**adaBoost**
Adult	0.157	0.52	0.157	1.00	0.172	0.00	0.151
Breast	0.041	0.50	0.041	0.87	0.053	0.50	0.053
Credit	0.187	0.00	0.129	0.84	0.144	0.43	0.141
Glass	0.140	0.61	0.150	0.40	0.140	0.95	0.206
Haberman	0.279	0.57	0.286	0.05	0.231	0.50	0.231
Hearts	0.196	0.03	0.135	0.98	0.196	0.97	0.257
Ionosphere	0.057	0.77	0.074	1.00	0.160	0.05	0.109
Letter:AB	0.000	0.94	0.004	0.12	0.001	0.99	0.009
Letter:DO	0.014	0.73	0.017	0.04	0.008	1.00	0.040
Letter:OQ	0.009	0.76	0.013	0.98	0.027	0.89	0.038
Liver	0.349	0.71	0.366	0.75	0.395	0.07	0.331
MNIST:08	0.003	1.00	0.012	0.04	0.006	1.00	0.016
MNIST:17	0.007	0.69	0.007	0.64	0.008	0.36	0.007
MNIST:18	0.011	1.00	0.037	0.00	0.017	0.42	0.016
MNIST:23	0.017	1.00	0.035	0.00	0.022	1.00	0.041
Mushrooms	0.000	0.50	0.000	0.99	0.001	0.01	0.000
Ringnorm	0.015	1.00	0.054	1.00	0.282	0.00	0.027
Sonar	0.154	0.84	0.202	0.42	0.192	0.59	0.202
Tic-Tac-Toe	0.161	0.00	0.052	1.00	0.198	1.00	0.357
USVotes	0.069	0.25	0.065	0.98	0.092	0.01	0.051
WDBC	0.049	0.02	0.021	1.00	0.077	0.03	0.042
Waveform	0.068	0.26	0.067	1.00	0.080	0.41	0.079

Experimental Setup

Each data set is split into two equal-size data set called S_i and T_i. Then, each learning algorithm is trained on S_i and tested on T_i using the zero-one loss. Finally, all pairs of learning algorithms are compared using both the Poisson binomial test and the sign test.

Since each learning algorithm comes with adjustable hyperparameters, we use the 10-fold cross validation estimate of the error rate on S_i to select the most appropriate set of values.

Details on Learning Algorithms

We present here the details about the tuning of the learning algorithms used in the experimental setup.

To concisely define the list of explored hyperparameters, we use $\text{Lin}_k(a, b) \overset{\text{def}}{=} \{a, a+s, a+2s, ..., b\}$ where $s \overset{\text{def}}{=} \frac{b-a}{k-1}$. This represents k values uniformly selected in the interval $[a, b]$.

Similarly, we define $\text{Log}_k(a, b) \stackrel{\text{def}}{=} \{10^x \mid x \in \text{Lin}_k(a, b)\}$.

When searching the appropriate parameter for the RBF kernel, we use the following approach. First recall that the RBF kernel values are given by

$$k_{\text{RBF}}(x, x') \stackrel{\text{def}}{=} \exp\left(\frac{-r}{2\hat{\sigma}^2}\|x - x'\|^2\right) , \tag{4.6}$$

where r is found using cross-validation and $\hat{\sigma}$ is obtained using the train set. To obtain an appropriate value for $\hat{\sigma}$, we used the 10th percentile over the distribution of distances between all pairs of examples in S_i.

To build weak learners for boosting algorithms, we use c stumps per attribute where c is constant across all attributes and is found using cross validation. For each attribute, each of the c threshold values are uniformly distributed across the interval of values realized on the training data.

svm: Support Vector Machine Cortes and Vapnik [1995] with RBF kernel. The width r of the kernel (Equation (4.6)) is selected from $\text{Log}_{20}(-5, 5)$ and the soft-margin parameter C is selected from $\text{Log}_{20}(-2, 5)$.

parzen: Parzen Window Parzen [1962], Rosenblatt [1956] with RBF kernel. The width r of the kernel is selected from $\text{Log}_{30}(-4, 4)$.

adaBoost: AdaBoost Freund and Schapire [1995], using stumps as weak classifiers. The number of iterations is selected using cross validation for values in $\{2^n \mid n \in \{1, 2, ..., 10\}\}$ and the number of stumps is selected from { 1, 2, 3, 4, 6, 10, 14, 21, 31 }.

ann: Artificial Neural Networks Minsky and Papert [1969] with two hidden layers, sigmoid activation function, and L^2 regularization of the weights. The input space is normalized on the training set, such that each attribute has zero mean and unit variance. The number of neurons on the second layer is $\lceil \sqrt{N_{l1}} \rceil$ where N_{l1} is the number of neurons on the first layer and is selected from { 3, 4, 5, 6, 7, 9, 11, 13, 16, 19, 23, 28, 33, 40, 48, 57, 69, 83, 100 }. Finally, the weight of the L^2 regularizer is selected from $\text{Log}_{20}(-2, 2)$. The network is trained using conjugate gradient descent with no early stopping.

In all cases, when there are two hyperparameters, all pair of proposed values are explored.

Chapter 5

Second Paper Presentation

5.1 Details

Agnostic Bayesian Learning of Ensembles
Alexandre Lacoste, François Laviolette,Mario Marchand, and Hugo Larochelle
In *Proceedings of The 31st International Conference on Machine Learning*, 2014.

5.2 Early Version

An early version of this work was presented at a NIPS workshop in December 2012, and can be found in Appendix A.

Model Averaging With Holdout Estimation of the Posterior Distribution
Alexandre Lacoste, François Laviolette, and Mario Marchand
In *NIPS workshop on Perturbations, Optimization, and Statistics*, December 2012.

5.3 Context

In Section 1.3, we saw how to select the optimal hyperparameter configuration on a validation set $V \sim D^n$ using

$$\gamma_V \stackrel{\text{def}}{=} \operatorname*{argmin}_{\gamma \in \Gamma} R_V(h_\gamma).$$

While this works well in practice, by selecting the single best model, we may suffer from overfitting. This overfitting is usually mild since $|\Gamma|$ traditionally ranges from a few dozen to a few hundred hyperparameters configurations. However, with the increasing computer speed and new optimization techniques [Snoek et al., 2012], machine learning scientists are now able to explore more exotic models composed of several hyperparameter variables. In such cases, overfitting can become an important problem and treating it can provide significant generalization benefits.

In Section 2.1.6, we saw that when the model is Bayesian, the prior parameters can be marginalized out through model averaging. Sadly, this technique is not usable in many contexts. For one, most practical learning algorithms are not fully Bayesian. Also, the required marginal likelihood term usually comes with an important computational cost. Finally, as seen in Section 2.1.6, blindly performing a full hierarchical Bayesian inference without any validation can suffer from mis-specifications problems.

For this reason, conventional cross-validation is still widely used. Yet, it is frequent to observe a generalization gain by combining the different models obtained through cross-validation [Zhang et al., 2006, Bell et al., 2007]. However, by choosing the single best combination of models, we just postpone the problem of overfitting to a more complex set of predictors. Some Bayesian approaches to model combination have been proposed [Zhang et al., 2006], but it is limited to the zero-one loss function and the computational cost is impractical.

In other words, there is no learning algorithm that directly addresses model averaging during the validation step to integrate out the uncertainty of selecting the single best predictor.

5.4 Contributions

In this work we explore 3 different models for estimating the following probability distribution

$$p(h^\star = h|S),$$

where $h^\star \overset{\text{def}}{=} \underset{h \in \mathcal{H}}{\operatorname{argmin}}\ R_D(h)$ is a random variable corresponding to which predictor is the best according to D. This distribution is equivalent to $p\left(R_D(h) \leq R_D(g)\ \forall g \in \mathcal{H}|S\right)$ and generalizes[1] the previously obtained result, $p\left(R_D(h) \leq R_D(g)|S\right)$, to $|\mathcal{H}|$ greater than two and to any loss function.

As a practical application, we apply this approach on the set of predictors obtained during the validation of hyperparameters. Specifically, after training all predictors $h_\gamma \overset{\text{def}}{=} \mathcal{A}_\gamma(S)$ for each hyperparameter configuration $\gamma \in \Gamma$, we obtain a set of predictors $\mathcal{H}_\Gamma \overset{\text{def}}{=} \{h_\gamma\}_{\gamma \in \Gamma}$. Then, instead of using conventional validation and searching for the one minimizing $R_V(h_\gamma)$, we use $p(h^\star = h_\gamma|V)$ to carry the uncertainty about which one is the best. By doing so, we can perform model averaging and get the generalization benefits that come with it. Extensive experiments on several data sets confirm this anticipated generalization gain. We also show how this method can be adapted to cross-validation.

[1]We now use S instead of T to estimate this distribution. This is equivalent since S, T, and V are all i.i.d samples coming from the same distribution.

5.5 Comments

In this work, we apply the posterior distribution $p(h^\star = h_\gamma | V)$ on the set $\mathcal{H}_\Gamma$. This provides a really simple and fast algorithm for a state of the art ensemble method. However, we would like to highlight that this tool is more flexible than what this work implies. For example, if one really aims at combining predictors instead of just performing model averaging, it suffices to propose a set, $\mathcal{H}_{\text{combiners}}$, of combining functions $h_{\text{combiner}} : \mathcal{Y}^{|\Gamma|} \to \mathcal{Y}$ and obtain the corresponding distribution $p\left(h^\star = h_{\text{combiner}} | V\right)$ over $\mathcal{H}_{\text{combiners}}$.

We also note that this tool is highly related to bagging [Breiman, 1996] when used with the bootstrap approach. In some sense, it provides a theoretical backbone to bagging while being more specific in its way of performing the ensemble.

Chapter 6

Agnostic Bayesian Learning of Ensembles

6.1 abstract

We propose a method for producing ensembles of predictors based on holdout estimations of their generalization performances. This approach uses a prior directly on the performance of predictors taken from a finite set of candidates and attempts to infer which one is best. Using Bayesian inference, we can thus obtain a posterior that represents our uncertainty about that choice and construct a weighted ensemble of predictors accordingly. This approach has the advantage of not requiring that the predictors be probabilistic themselves, can deal with arbitrary measures of performance and does not assume that the data was actually generated from any of the predictors in the ensemble. Since the problem of finding the *best* (as opposed to the true) predictor among a class is known as agnostic PAC-learning, we refer to our method as *agnostic Bayesian learning*. We also propose a method to address the case where the performance estimate is obtained from k-fold cross validation. While being efficient and easily adjustable to any loss function, our experiments confirm that the agnostic Bayes approach is state of the art compared to common baselines such as model selection based on k-fold cross-validation or a learned linear combination of predictor outputs.

6.2 Introduction

When designing a machine learning system that relies on a trained predictor, one is usually faced with the problem of choosing this predictor from a finite class of models. In practice, the class of models might correspond to different learning algorithms or to different choices of hyperparameters for a specific learning algorithm. A common approach to this problem is to estimate the generalization performance of each predictor on a holdout data set (through a training/validation set split or using k-fold cross-validation) and use the predictor with the best performance. However, this approach is invariably noisy and overfitting can become a problem. A more successful procedure is to construct an ensemble of many different learned predictors. Many machine learning contests are won this way [Guyon et al., 2010]. For instance, the winning team of the Netflix's contest relied on a final predictor trained on the output of the learned models [Bell et al., 2007]. Great care must be taken however to avoid overfitting, e.g. by carefully tuning the predictor's own regularization hyperparameters. The choice of the final predictor is likely to influence the end result as well.

At the heart of this selection problem is our inability to know for sure which predictor is the best among our model class. One natural way to reason about such uncertainty would be to formulate it in probabilistic terms. In this paper, we propose to follow this paradigm by formulating priors about the expected performance of each predictor in our chosen class of models. We then use the observed loss measurements on each held-out example as evidence for updating our posterior over the identity of the best predictor in the model class. At test time, we can use this posterior to weight the contribution of each predictor in the ensemble that performs the final prediction.

We explore different ways of expressing priors over predictor performances and discuss how to perform Bayesian inference. As we will see, this simple paradigm naturally takes into account the correlation between the predictor's output so as to leverage diversity among the ensemble, which is another desiderata for ensemble learning and model averaging methods.

Unlike Bayesian model averaging [Hoeting et al., 1999], our approach does not require that the predictors be themselves probabilistic. It can also deal with arbitrary performance measures. More crucially, this approach does not assume that the observed data has been generated by a predictor from the model class. In other words, we are not looking for the predictor that best explains the observed data, assuming it was generated by a predictor coming from our model class. Instead, at the centre of our approach, we want to find the *best* predictor in terms of a task's performance measure and among all available predictors, while reasoning about our uncertainty around this problem in a Bayesian way.

The non-reliance on the assumption that the true underlying data generating function belongs to our model class is also at the center of agnostic PAC-learning. For this reason, we refer to the proposed framework as *agnostic Bayesian learning.*

Section 6.3 formally describes the agnostic Bayes approach. We then propose a few methods for obtaining a posterior distribution over a set of predictors. Section 6.5 presents an adaptation to k-fold cross-validation estimation of the losses. Finally, several experimental results are presented in Section 6.7.

6.3 Theoretical Setup

Throughout this paper, we use the inductive learning paradigm and make the usual assumptions of PAC learning theory [Kearns et al., 1994, Valiant, 1984]. Thus, a task D corresponds to a probability distribution over the input-output space $\mathcal{X} \times \mathcal{Y}$. Given a training set $S \sim D^m$, the objective is to find, among a set $\mathcal{H}$, the *best* function $h^\star : \mathcal{X} \to \mathcal{Y}$. In general, $\mathcal{H}$ could be any set. However, this work will focus on the case where $\mathcal{H}$ is a finite set of predictors obtained from one or many learning algorithms, with various hyperparameters. We will refer to a member of $\mathcal{H}$ as an hypothesis.

To assess the quality of an hypothesis, we use a loss function $\mathcal{L} : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}$ that quantifies the penalty incurred when h predicts $h(x)$ while the true answer is y. Then, we can define the risk $R_D(h)$ as being the expected loss of h on task D, i.e. $R_D(h) \overset{\text{def}}{=} \underset{x,y\sim D}{\mathbf{E}} \mathcal{L}\left(h(x), y\right)$. Finally, the *best*[1] function is simply the one minimizing the risk, i.e. $h^\star \overset{\text{def}}{=} \underset{h\in\mathcal{H}}{\operatorname{argmin}}\, R_D(h)$.

Since we do not observe D, it is not generally possible to find $h^\star$ with certainty. For this reason, we are interested in inferring $h^\star$ while modeling our uncertainty about it, using a

[1]The best solution may not be unique.

posterior probability distribution $\Pr(h^\star = h|S)$. Then, after marginalizing $h^\star$, we obtain a probabilistic prediction

$$\Pr(y^\star{=}y|x,S) = \sum_{h\in\mathcal{H}} \Pr(h^\star{=}h|S)\Pr(y^\star{=}y|x,h),$$

where $y^\star$ stands for the prediction made by $h^\star$ for a given x. We note that the uncertainty in this prediction solely comes from our lack of knowledge about $h^\star$.

In order to perform a final prediction $\hat{y}$ for a given x it is tempting to use the optimal Bayes decision theory

$$\hat{y} = \operatorname*{argmin}_{y'\in\mathcal{Y}} \sum_{y\in\mathcal{Y}} \Pr(y^o{=}y|x,S)\mathcal{L}\left(y',y\right),$$

where y^o is the random variable corresponding to the observed values of y. However, the contrast between $\Pr(y^o{=}y|x,S)$ and $\Pr(y^\star{=}y|x,S)$ prevents us from using this approach. To this end, we use:

$$\hat{y} = \operatorname*{argmax}_{y\in\mathcal{Y}} \Pr(y^\star{=}y|x,S),$$

the most probable answer. This yields the following ensemble method:

$$E^\star(x) \overset{\text{def}}{=} \operatorname*{argmax}_{y\in\mathcal{Y}} \sum_{h\in\mathcal{H}} \Pr(h^\star = h|S)\,\mathrm{I}[h(x) = y] \tag{6.1}$$

Before going further, we first review the usual Bayesian model averaging approach to highlight the fact that it does not exactly use $\Pr(h^\star{=}h|S)$.

6.3.1 Standard Bayesian Model Averaging

To address the inductive learning paradigm, a variant of Bayesian model averaging can be used, where we suppose that a deterministic function $h^\rightarrow$, belonging to $\mathcal{H}$, is at the origin of the observed relationship between x and y. To perform inference on $h^\rightarrow$, we treat it as a random variable and assume that the observations in S have been altered by a noise model[2] $p(y^o = y|x,h)$. Using the i.i.d. assumption, $p(S|h) = \prod_{i=1}^m p(y_i|x_i,h)p(x_i)$. Next, by defining a prior distribution over $\mathcal{H}$, we can perform Bayesian inference to compute $p(h^\rightarrow = h|S) \propto p(S|h)p(h)$. Finally, after marginalization of h, we obtain

$$\Pr(y^o{=}y|x,S) = \sum_{h\in\mathcal{H}} p(h^\rightarrow{=}h|S)\Pr(y^o{=}y|x,h),$$

which can be used with the optimal Bayes decision theory, to give the following ensemble decision rule

$$E^\rightarrow(x) \overset{\text{def}}{=} \operatorname*{argmin}_{y'\in\mathcal{Y}} \sum_{y\in\mathcal{Y}} \Pr(y^o{=}y|x,S)\mathcal{L}\left(y',y\right). \tag{6.2}$$

This formulation has proven to be very useful. However, if the *true* data-generating hypothesis does not belong to $\mathcal{H}$, the posterior $\Pr(h^\rightarrow = h|S)$ may not converge to a posterior peaked

[2]The noise model could also be inferred. In this work, we use a fixed noise model.

at the best hypothesis $h^\star$, as $m \to \infty$. This misbehavior has been studied by Grünwald and Langford [2007] for the zero-one loss scenario. It was shown that under some reasonable restrictions on the prior, there exists a distribution D where the risk of the Bayes predictor is significantly higher than $R_D(h^\star)$.

One way to overcome this inconsistency is to commit to a noise model that leverages the loss function, such as $\Pr(y^o = y|h, x) \propto e^{-\beta\mathcal{L}(h(x),y)}$ for some fixed $\beta > 0$. Then, we have that $\Pr(h^{\rightarrow}{=}h|S) \propto \Pr(h)e^{-m\beta R_S(h)}$, where $R_S(h)$ is the empirical risk measured on S. As $m \to \infty$, the exponential part of the posterior ensures that any hypothesis not having a risk as low as $R_D(h^\star)$ will have a negligible weight. We will examine this ensemble method to show that it is outperformed by the methods we propose in this paper.

6.3.2 Agnostic Bayes

Our main contribution is to propose a method for obtaining $p(h^\star = h|S)$, to be used in our ensemble decision $E^\star(x)$. The core idea of our approach is to directly reason about $h^\star$ instead of assuming the existence in $\mathcal{H}$ of a data generating $h^{\rightarrow}$ and trying to infer it. Since the observed losses in S suffice to distinguish $h^\star$ from other hypotheses in $\mathcal{H}$, we do not have to commit to a particular model for the relationship between x and y, and can limit ourselves to modeling the losses under each hypothesis.

Specifically, we propose to treat the risk $r_h \overset{\text{def}}{=} R_D(h)$ of each hypothesis h as a random variable, over which we will be defining a prior distribution. Let $l_{h,i} \overset{\text{def}}{=} \mathcal{L}\left(h\left(x_i\right), y_i\right)$ be the observed loss of hypothesis h for a sample $(x_i, y_i) \in S$. We also treat $l_{h,i}$ as random variables, governed by a conditional distribution $p(l_{h,i}|r_h)$. For example, in the zero-one loss $\mathcal{L}(y, y') = \mathrm{I}[y \neq y']$ case, a natural choice would be to treat the observed losses $l_{h,i}$ as Bernoulli trials of parameter r_h. Assuming a beta prior over r_h, we could then perform Bayesian inference in order to reason about the uncertainty over r_h given the losses observed from S.

In the case of ensemble learning where we have multiple competing hypotheses, the losses $l_{h,i}$ are dependent across the different hypotheses h for the same example (x_i, y_i). Hence, we need to model the losses $\boldsymbol{l}_i \overset{\text{def}}{=} \left(l_{1,i}, l_{2,i}, \ldots, l_{|\mathcal{H}|,i}\right)$ for a given example jointly, given the joint risk for all hypotheses $\mathbf{r} \overset{\text{def}}{=} \left(r_1, r_2, \ldots, r_{|\mathcal{H}|}\right)$. Section 6.4 will discuss different joint priors $p(\mathbf{r})$ and observation models $p(\boldsymbol{l}_i|\mathbf{r})$. For now, we just note that from $p(\boldsymbol{l}_i|\mathbf{r})$, we can derive the likelihood of the set of losses $L \overset{\text{def}}{=} \{\boldsymbol{l}_i\}_{i=1}^m$ as $p(L|\mathbf{r}) = \prod_{i=1}^m p(\boldsymbol{l}_i|\mathbf{r})$ and, combined with our prior $p(\mathbf{r})$, perform Bayesian inference to obtain $p(\mathbf{r}|L) \propto p(L|\mathbf{r})p(\mathbf{r})$.

After obtaining $\Pr(\mathbf{r}|L)$, we can now compute the posterior probability that a given hypothesis

h is the best hypothesis $h^\star$ with the lowest risk among $\mathcal{H}$

$$\begin{aligned} &Pr\left(\forall g \in \mathcal{H} : r_h \leq r_g \;\mid L\right) \\ &= \underset{\mathbf{r}\sim\Pr(\cdot\,\mid\,L)}{\mathbf{E}} p\left(r_h \leq r_g,\; \forall g \neq h|\mathbf{r}\right) \\ &= \underset{\mathbf{r}\sim\Pr(\cdot\,\mid\,L)}{\mathbf{E}} \mathrm{I}\left(r_h \leq r_g,\; \forall g \neq h\right). \end{aligned}$$

We propose to use this posterior as our ensemble posterior in Equation (6.1). Under this model, L is a sufficient statistic for $\mathbf{r}$ and thus for h, i.e. $p\left(h|S\right) = p\left(h|L\right)$. Hence, to sample from $p(h|S)$, it suffices to sample a joint risk $\mathbf{r}$ from $\Pr\left(\mathbf{r} \mid L\right)$ and to search for the hypothesis with the smallest risk. With repeated sampling, we can then approximately compute our ensemble decision rule. When $\mathcal{Y}$ is continuous, this approximation can affect $\underset{y\in\mathcal{Y}}{\operatorname{argmax}}\, p(y^\star = y|S,x)$. To address this issue, we consider a simple Gaussian model to smooth $p(y^\star = y|S,x)$. This yields a weighted average of the predictions: $E^\star(x) = \sum_{h\in\mathcal{H}} p(h^\star = h|S)h(x)$.

6.4 Priors Over the Joint Risk

In this section, we propose a few choices for the prior $p(\mathbf{r})$ and observation model $p(\boldsymbol{l}_i|\mathbf{r})$. We also discuss how to perform inference for $\Pr(\mathbf{r}|L)$ under different assumptions of the loss function.

6.4.1 Dirichlet Distribution

We start with a proposal for the specific case of the zero-one loss. As described in Section 6.3, the observations $l_{h,i} \in \{0,1\}$ are correlated and put together in a vector $\boldsymbol{l}_i \in \{0,1\}^d$, where $d \overset{\text{def}}{=} |\mathcal{H}|$. We propose to consider the collection of observations $\{\boldsymbol{l}_i\}_{i=1}^m$ as coming from a categorical distribution of $N \overset{\text{def}}{=} 2^d$ possible states (i.e. outcomes). Therefore, the counts of observations $\mathbf{k} \overset{\text{def}}{=} (k_1, k_2, \ldots, k_N) \in \mathbb{N}^N$ come from a multinomial distribution of parameters $\mathbf{q}$ and m, where $\mathbf{q}$ is the probability of observing each event and sums to 1. With these assumptions, it is natural to use the Dirichlet distribution of parameter $\boldsymbol{\alpha}$ as the model for the prior over $\mathbf{q}$. The posterior distribution $\Pr\left(\mathbf{q} \mid \mathbf{k}\right)$ is then a Dirichlet distribution of parameter $\boldsymbol{\alpha} + \mathbf{k}$. To convert the sample from $\Pr(\mathbf{q}|L)$ to a sample from $\Pr(\mathbf{r}|L)$, we define the state matrix $G \in \{0,1\}^{d\times N}$ where the j^{th} column corresponds to the binary representation of j. Then, to obtain a sample from $\Pr(\mathbf{r}|L)$, we sample $\mathbf{q}$ from $\mathrm{Dir}\left(\boldsymbol{\alpha} + \mathbf{k}\right)$ and use $\mathbf{r} = G\mathbf{q}$. Equivalently, we have $\Pr\left(\mathbf{r}|L\right) = \mathbf{E}_{\,\mathbf{q}\sim\mathrm{Dir}(\boldsymbol{\alpha}+\mathbf{k})} \mathrm{I}\left[G\mathbf{q} = \mathbf{r}\right]$.

Naively sampling from this posterior yields an algorithm with computational complexity of $O(d2^d)$. However, using a neutral prior of the form $\boldsymbol{\alpha} = \tilde{\alpha}\mathbf{1}_N$ and the stick breaking representation of the Dirichlet (see Lemma 3.1 of Sethuraman [1991]), we have the following identity

$$\theta X_{\boldsymbol{\alpha}} + (1-\theta)X_{\mathbf{k}} = X_{\boldsymbol{\alpha}+\mathbf{k}},$$

where $\theta \sim \text{Beta}(\widetilde{\alpha}N, m)$, $X_{\boldsymbol{\alpha}} \sim \text{Dir}(\boldsymbol{\alpha})$, $X_{\mathbf{k}} \sim \text{Dir}(\mathbf{k})$, $X_{\boldsymbol{\alpha}+\mathbf{k}} \sim \text{Dir}(\boldsymbol{\alpha}+\mathbf{k})$. Since most values in $\mathbf{k}$ are zeros, samples from $\text{Dir}(\mathbf{k})$ can be obtained in $O(m)$. Thus, we are left with the task of sampling from $\text{Dir}(\boldsymbol{\alpha})$, which can be approximated efficiently using $10 \cdot \widetilde{\alpha}N$ samples from the stick breaking process Sethuraman [1991]. Since $\widetilde{\alpha}N > m$ yields too much importance to the prior, one can safely assume that $\widetilde{\alpha}N \leq m$ and obtain a sample from the prior with computational complexity O(m) .

6.4.2 Bootstrap Inference

We point out that the Dirichlet posterior presented in Section 6.4.1 is a generalization of Rubin's Bayesian bootstrap [Rubin, 1981] and is equivalent in the limit $\widetilde{\alpha} \to 0$. Also, Rubin showed that the Bayesian bootstrap is statistically tightly related to Efron's bootstrap [Efron, 1979]. For these reasons, we also consider the bootstrap as a candidate for a simple and generic method to sample from $\Pr(\mathbf{r}|L)$. This is done by sampling with replacement a set $\{\boldsymbol{l}'_i\}_{i=1}^m$ from $\{\boldsymbol{l}_i\}_{i=1}^m$. To obtain $\mathbf{r}$, we use $r_h \leftarrow \sum_{i=1}^m \frac{1}{m} l'_{h,i}; \forall h \in \mathcal{H}$.

6.4.3 t Distribution

In this section, we make the assumption that the variables $\boldsymbol{l}_i$ are observations coming from a multivariate normal distribution of dimensionality $|\mathcal{H}| \overset{\text{def}}{=} d$, whose mean parameter corresponds to the true risk $\mathbf{r}$. While the normal assumption is generally not true, it can be justified from the central limit theorem. As we will see, experiments in Section 6.7 show that this assumption works well in practice even with the zero-one loss function, which is one of the most extreme cases of non Gaussian samples.

Specifically, assuming that $p(\boldsymbol{l}_i|\mathbf{r},\Lambda)$ is normal, the likelihood of $L \overset{\text{def}}{=} \{\boldsymbol{l}\}_{i=1}^m$ is

$$\Pr\left(L \mid \mathbf{r}, \Lambda\right) \;\propto\; |\Lambda|^{\frac{m}{2}} e^{\left(-\frac{1}{2}\sum_{j=1}^m (\boldsymbol{l}_j - \mathbf{r})^T \Lambda (\boldsymbol{l}_j - \mathbf{r})\right)}. \tag{6.3}$$

We want to favor the use of priors over $\mathbf{r}$ and covariance matrix Λ^{-1} such that the posterior $p(\mathbf{r},\Lambda|L)$ is tractable. This can be achieved using the normal-Wishart distribution [DeGroot, 2005, p. 178]

$$\Pr\left(\mathbf{r}, \Lambda\right) \;=\; \mathcal{N}\left(\mathbf{r} \;\middle|\; \mathbf{r}_0, (\kappa_0\Lambda)^{-1}\right) \mathcal{W}\left(\Lambda \mid T_0, \nu_0\right),$$

where $\mathcal{N}$ and $\mathcal{W}$ are the normal and Wishart distributions respectively, $\mathbf{r}_0$ and T_0 are the mean and covariance prior, while κ_0 and ν_0 are parameters related to the confidence we have in $\mathbf{r}_0$ and T_0 respectively (with restrictions $\kappa_0 > 0$ and $\nu_0 > d-1$). Thanks to conjugacy, after observing L, we have that the posterior $p\left(\mathbf{r}, \Lambda \mid L\right)$ is also a normal-Wishart distribution of parameters κ_m, ν_m, $\mathbf{r}_m$ and T_m as follows:

$$\begin{aligned}
\kappa_m &= \kappa_0 + m \\
\nu_m &= \nu_0 + m \\
\mathbf{r}_m &= \frac{\kappa_0 \mathbf{r}_0 + m\bar{\boldsymbol{l}}}{\kappa_m} \\
T_m &= T_0 + mS + m\frac{\kappa_0}{\kappa_m}\left(\mathbf{r}_0 - \bar{\boldsymbol{l}}\right)\left(\mathbf{r}_0 - \bar{\boldsymbol{l}}\right)^T
\end{aligned} \tag{6.4}$$

where $\bar{\boldsymbol{l}} \overset{\text{def}}{=} \frac{1}{m}\sum_{i=1}^{m} \boldsymbol{l}_i$ and $S \overset{\text{def}}{=} \frac{1}{m}\sum_{i=1}^{m}(\boldsymbol{l}_i - \bar{\boldsymbol{l}})(\boldsymbol{l}_i - \bar{\boldsymbol{l}})^T$. Since our goal is to obtain a posterior distribution over $\mathbf{r}$ only, we have to marginalize out Λ from $\Pr(\mathbf{r}, \Lambda \mid L)$. By doing so, we obtain the multivariate Student's t distribution with $\widetilde{\nu} \overset{\text{def}}{=} \nu_m - d + 1$ degrees of freedom [DeGroot, 2005, p. 179]

$$\Pr(\mathbf{r} \mid L) \quad = \quad t\left(\mathbf{r} \,\middle|\, \widetilde{\nu}, \mathbf{r}_m, \tfrac{T_m}{\kappa_m \widetilde{\nu}}\right). \tag{6.5}$$

Samples from this multivariate t-distribution are done by sampling from the normal distribution $\mathbf{z} \sim \mathcal{N}\left(0, \frac{T_m}{\kappa_m \widetilde{\nu}}\right)$, sampling from the chi-squared distribution $\xi \sim \chi^2(\widetilde{\nu})$ and computing $\mathbf{r}_m + \mathbf{z}\sqrt{\frac{\widetilde{\nu}}{\xi}}$. This gives an overall computational complexity of $O\left(d^2(m + k + d)\right)$ to obtain k samples.

For setting the parameters $\mathbf{r}_0$, T_0, κ_0 and ν_0 of the prior, we chose values that were as neutral as possible and numerically stable: $\mathbf{r}_0 = 0.5 \times \mathbf{1}_d$, $T_0 = 0.25 \times I$, $\kappa_0 = 1$ and $\nu_0 = d$.

6.4.4 Posterior Behavior with Correlated Hypotheses

One advantage of the agnostic Bayes posterior for constructing an ensemble is that it naturally encourages diversity among the predictors, even in the presence of correlation between the predictors in $\mathcal{H}$. We illustrate this with a simple example, shown in Table 6.1, comparing an agnostic Bayes ensemble with bootstrap inference ($E_b^\star$) and a Bayesian model averaging ensemble with a loss-based noise model and flat prior over the hypotheses ($E^{\rightarrow}$). Table 6.1(top) illustrates the case of three equally good but different hypotheses, based on three observed losses for each predictor. We see that both $E_b^\star$ and $E^{\rightarrow}$ equally weight the three hypotheses, as expected.

Now, in Table 6.1(bottom), we include into $\mathcal{H}$ an additional hypothesis h_4, which is identical to h_3. We then observe that $E_b^\star$ naturally maintains diversity within the ensemble, by reducing the mass of the identical hypotheses h_3 and h_4, compared to $E^{\rightarrow}$ which still weights all hypotheses equally. Diversity is usually considered to be beneficial when constructing an ensemble of predictors Roy et al. [2011], motivating the use of agnostic Bayes for this task.

Table 6.1: Illustration of the posteriors in an agnostic Bayes ensemble ($E_b^\star$) and in Bayesian model averaging ($E^\rightarrow$). **top:** Uncorrelated predictors. **bottom:** Addition of a correlated predictor.

	$\boldsymbol{l}_1$	$\boldsymbol{l}_2$	$\boldsymbol{l}_3$	$\Pr(h^\star \mid S)$	$\Pr(h^\rightarrow \mid S)$
h_1	1	0	0	0.33	0.33
h_2	0	1	0	0.33	0.33
h_3	0	0	1	0.33	0.33

$$\downarrow$$

	$\boldsymbol{l}_1$	$\boldsymbol{l}_2$	$\boldsymbol{l}_3$	$\Pr(h^\star \mid S)$	$\Pr(h^\rightarrow \mid S)$
h_1	1	0	0	0.31	0.25
h_2	0	1	0	0.31	0.25
h_3	0	0	1	0.19	0.25
h_4	0	0	1	0.19	0.25

6.5 Model Averaging for Trained Predictors

As mentioned in Section 6.3, one natural application for the inference of the best hypothesis is model averaging of trained predictors. Namely, let $\mathcal{A}_\gamma$ be a learning algorithm with a hyperparameter configuration $\gamma \in \Gamma$ and let $h_\gamma = \mathcal{A}_\gamma(T)$ represent the classifier obtained using a training set $T \sim D^n$, disjoint from S. The set $\mathcal{H}$ contains all classifiers obtained from each $\gamma \in \Gamma$, when $\mathcal{A}_\gamma$ is trained on T, i.e. $\mathcal{H} \stackrel{\text{def}}{=} \{h_\gamma \mid \gamma \in \Gamma\}$. Finally, to obtain the posterior $\Pr(h_\gamma^\star = h_\gamma | S)$, we rely on the set S. Experiments in Section 6.7 will show that this approach significantly outperforms the usual method of selecting the hypothesis minimizing $R_S(h_\gamma)$.

Unfortunately, this scenario requires that the hypotheses h_γ be trained on a set of data T separate from S, in a training/validation split fashion, wasting an opportunity to measure the hypotheses performance on T as well. Our next step is thus to adapt our agnostic Bayes approach to the k-fold cross-validation scenario, which more fully uses the available data.

6.5.1 Adapting to k-fold Cross-Validation

Let $\{V_1, V_2, \ldots, V_k\}$ be a partition of S, and let $h_{\gamma,j} \stackrel{\text{def}}{=} \mathcal{A}_\gamma(S \setminus V_j)$. Now, denote the loss of model γ on the example (x_i, y_i) as $\widetilde{l}_{\gamma,i} \stackrel{\text{def}}{=} \mathcal{L}(h_{\gamma,j_i}(x_i), y_i)$, where j_i is the unique index j such that $(x_i, y_i) \in V_j$. Finally, let $\widetilde{\boldsymbol{l}}_i \stackrel{\text{def}}{=} \left(\widetilde{l}_{1,i}, \widetilde{l}_{2,i}, \ldots, \widetilde{l}_{|\Gamma|,i}\right)$. Unlike $\{\boldsymbol{l}\}_{i=1}^m$, it is well known that the set of k-fold generated losses $\{\widetilde{\boldsymbol{l}}\}_{i=1}^m$ contains dependencies across the different examples that are induced by the k-fold procedure Bengio and Grandvalet [2004]. Since the posteriors described in Section 6.4 relied on independence across examples, we cannot simply ignore the dependencies induced within this process and must adapt our approach.

Specifically, we make the simplifying assumption that these dependencies only affect the effective number of samples. Intuitively, since samples are correlated, there may not be as many as it seems and the estimation of $\Pr(\mathbf{r}|L)$ may be overly confident. We thus propose to add an extra parameter ρ, *the effective sample size ratio*, to compensate for these dependencies. While this parameter requires calibration, we describe in Section 6.5.2 an efficient method for automatically adjusting its value.

To include the effective sample size ratio in the methods described in Section 6.4, we will effectively act as if the collection $\{\boldsymbol{l}\}_{i=1}^{m}$ had been generated by artificially replicating a set of m original samples b times each, to give a new set of $bm' \overset{\text{def}}{=} m$ samples. Thus, the effective number of samples would be $m' = m/b$. Now, supposing that we know $\rho = m'/m$, we want to adapt the posterior's parameters in such a way that the posterior's distribution remains the same, on average, as before the "corruption".

Bootstrap: This is probably the simplest method to adapt. Out of the m observed events, we sample with replacement m' events instead, where $m' = \lceil \rho m \rceil$.

Dirichlet: In this case, each observed event is made to count for ρ instead of 1. After observing m events, the vector of counts $\mathbf{k}' \overset{\text{def}}{=} (k'_1, k'_2, \ldots, k'_N)$ will now sum to m' instead of m.

t-Distribution: In this case, we adapt the quantities described in Equation (6.4) as follows: $\nu_{m'} = \nu_0 + m'$, $\nu_{m'} = \nu_0 + m'$, $\mathbf{r}_{m'} = \frac{\kappa_0 \mathbf{r}_0 + m' \bar{\boldsymbol{l}}}{\kappa_{m'}}$ and $T_{m'} = T_0 + m'S + m' \frac{\kappa_0}{\kappa_{m'}} \left(\mathbf{r}_0 - \bar{\boldsymbol{l}}\right) \left(\mathbf{r}_0 - \bar{\boldsymbol{l}}\right)^T$.

6.5.2 Tuning Parameters

To adjust ρ, we treat it as a parameter and fit it by optimizing the resulting ensemble's performance on S, thereby measuring how well the ensemble's weighting posterior can predict each label y_i in S from the hypotheses $(h_{1,j_i}(x_i), h_{2,j_i}(x_i), .., h_{|\Gamma|,j_i}(x_i))$. We've found this to work well in practice. This procedure is also akin to methods that learn a parameterized linear combination of predictors by training on generated examples

$$\tilde{S} \overset{\text{def}}{=} \left\{ \left((h_{1,j_i}(x_i), h_{2,j_i}(x_i), .., h_{|\Gamma|,j_i}(x_i)), y_i \right) \right\}_{i=1}^{m}.$$

The best ρ from a set of 20 values equally spaced from 0.1 to 0.8 is used. We use a similar procedure to tune the prior parameter $\widetilde{\alpha}$ of the ensemble based on a Dirichlet prior.

6.6 Related Work

To overcome some mentioned weaknesses of Bayesian model averaging (such as the reliance on the existence of a single data-generating hypothesis belonging to $\mathcal{H}$), Kim and Ghahramani [2012] proposed an alternative method for Bayesian combination of classifiers. They suppose that, for a given x, the true label is at the origin of the behavior of each individual classifier. Therefore, by modeling the dependencies between each classifier on a validation set, they can perform inference of the original label. Unfortunately, it relies on a combination of MCMC and rejection sampling methods and the computational complexity of certain dependency models grows exponentially with $|\mathcal{H}|$. Thus, this approach is viable only for combining a small set of classifiers. It also only tackles classification tasks and doesn't take into account the loss related to the task at hand, as we do here.

Alternatively, ensemble pruning is an important approach to ensemble methods. Zhang et al. [2006] used semidefinite programming for solving a heuristic based on the covariance of the predictors. Interestingly, the core of their idea is highly related to the covariance matrix used in our t-distribution approach. Unfortunately, they can only address an approximation of their heuristic and it is limited to the zero-one loss.

6.7 Experiments

We performed experiments to assess the performance of the agnostic Bayes ensemble approach and compared with a few commonly used methods:

ArgMin (AMin): This method represents the common approach of selecting the model h_γ with the best estimated holdout risk $r_\gamma \overset{\text{def}}{=} \frac{1}{m}\sum_{i=1}^{m} l_{\gamma,i}$. When the minimum is not unique, we select one at random.

SoftMin (SMin): We use the Gibbs distribution with parameter β to produce a posterior distribution over the collection of h_γ from r_γ. *i.e.*, $\Pr(h_\gamma|S) \propto e^{-\beta r_\gamma}$ and β is selected with the method described in Section 6.5.2. This represents the alternative Bayesian model averaging approach described in Section 6.3.1.

$\boldsymbol{E_b^\star, E_D^\star, E_B^\star, E_t^\star}$**:** The different agnostic Bayes ensemble decision methods based on Equation (6.1) and using posterior inference based on the bootstrap, the Dirichlet distribution, the Bayesian bootstrap and the t-distribution respectively. Effective sample size ratio ρ and Dirichlet prior parameter $\widetilde{\alpha}$ are adjusted according to Section 6.5.2, while the t-distribution prior parameters are fixed to the values specified in Section 6.4.3. We use 1000 samples from $\Pr(\mathbf{r}|L)$ to estimate $\Pr(h|S)$.

MetaSVM (MSVM): We use MetaSVM to represent the state of the art approach *i.e.*, methods that learn a linear model over the set of models as a final predictor. This is done by using the collection $\tilde{S}$ described in Section 6.5.2 as a training set for the linear SVM. Traditional cross validation is used to select the best soft margin parameter over 20 candidates values ranging from 10^{-3} to 10^{0} on a logarithmic scale.

Meta Ridge Regression (MRR): When performing experiments on regression tasks, we use ridge regression as a substitution for MetaSVM. The regularization parameter is selected by the leave one out method over 30 candidates ranging from 10^{-4} to 10^{4} on a logarithmic scale.

6.7.1 Comparing Learning Algorithms On Multiple Data sets

The different model selection methods presented in the previous section are generic and are meant to work across different tasks. It is thus crucial that we test them on several data sets. For that, we have to rely on methods that do not assume commensurability across tasks, such as the sign test, the Wilcoxon signed rank test (WSR) [Demšar, 2006b] and the Poisson binomial test (PB test) [Lacoste et al., 2012]. The PB test is a Bayesian analogue of the sign test meant for comparing learning algorithms on a collection of tasks, called a context. More precisely, it provides a probabilistic answer to the question "*Does algorithm $\mathcal{A}$ have a higher probability of producing a better predictor than algorithm $\mathcal{B}$ in the given context?*", denoted by $\Pr(\mathcal{A} \succ \mathcal{B} \mid \mathcal{W})$, where $\mathcal{W}$ represents the context.

To build a substantial collection of data sets, we used the AYSU collection [Ulaş et al., 2009] coming from the UCI and the Delve repositories and we added the MNIST data set. We also converted the multiclass data sets to binary classification by either merging classes or selecting pairs of classes. The resulting context contains 38 data sets. We have also collected 22 regression data sets from the Louis Torgo collection.[3] to perform experiments using different loss functions.

The set Γ of models used in this experiment is a combination of SVMs, Artificial Neural Networks (ANN), random forests, extra randomized trees Geurts et al. [2006] and gradient tree boosting Friedman [2001] with several variants of hyperparameters. Considering the algorithm name as a hyperparameter and a grid search for each algorithm, this yields a set of 692 hyperparameter configurations, all of which are evaluated using 10 folds cross validation. For the experiments on regression data sets, we used a combination of Kernel Ridge Regression (KRR), Support Vector Regression (SVR), random forests, extra randomized trees and gradient boosted regression, yielding a total of 480 hyperparameter configurations. Except for a custom implementation of ANN and KRR, we used scikit-learn [Pedregosa et al., 2011] for all other implementations. For more details on the choice of hyperparameters, we refer the reader to the supplementary material.

6.7.2 Result Table Notation

Each conducted experiment compares the generalization performances of a set of M algorithms on a set of N data sets. In order to evaluate if the observed differences are statistically significant, we use the pairwise PB test where each cell of the table represents $\Pr(\text{row} \succ \text{column})$. Since the table has a form of symmetry, we have grayed out redundant information and removed the first column. In addition, we also highlight in blue the results having p-values lower than 0.1 according to the one tail sign test. In general, we have observed a strong correlation between the p-values of the sign test and the probabilities obtained from

[3]These data sets were obtained from the following source : `http://www.dcc.fc.up.pt/~ltorgo/Regression/DataSets.html`

the PB test. Note however that their values may differ and a highlighted cell does not imply a strong PB probability, nor the converse. Finally, we added a column to each table which reports the expected rank of each algorithm across the collection of data sets. The rank of predictor $h_i = \mathcal{A}_i(S_j)$ on test set T_j is defined as

$$\text{Rank}_{h_i,T_j} \overset{\text{def}}{=} \sum_{l=1}^{M} \text{I}\left[R_{T_j}(h_l) \leq R_{T_j}(h_i)\right].$$

Then, the expected rank is obtained from the empirical average $\mathbf{E}\left[\text{Rank}\right]_{h_i} \overset{\text{def}}{=} \frac{1}{N}\sum_{j=1}^{N} \text{Rank}_{h_i,T_j}$.

6.7.3 Comparison of Ensemble Decision Methods on Classification Tasks

Our first experiment compares the different methods and baselines in the setting where the hypotheses have been trained and validated on a single split of the data set. In this scenario, the training data generates the set of hypotheses while the validation data provides observations for building an ensemble. Finally, a testing set is used to report the performances. The effective sample size ratio is fixed to 1 in this scenario.

Table 6.2: Comparison of the four proposed agnostic model averaging methods, in the single training/validation split experiment (refer to Section 6.7.2 for notation).

	$\boldsymbol{E_D^\star}$	$\boldsymbol{E_t^\star}$	$\boldsymbol{E_b^\star}$	$\boldsymbol{E_B^\star}$	$\mathbf{E}$[rank]
$\boldsymbol{E_D^\star}$	0.500	0.509	0.524	0.652	2.43 /4
$\boldsymbol{E_t^\star}$	0.491	0.500	0.541	0.662	2.43 /4
$\boldsymbol{E_b^\star}$	0.476	0.459	0.500	0.640	2.46 /4
$\boldsymbol{E_B^\star}$	0.348	0.338	0.360	0.500	2.67 /4

From Table 6.2, there are no significant differences between our methods except for a slight reduction in generalization performances for $E_B^\star$, which corresponds to $E_D^\star$ with $\widetilde{\alpha}$ fixed to 0. In this experiment, the only adjusted parameter is $\widetilde{\alpha}$ in the method $E_D^\star$. This may explain why it is ranked first according to the expected rank metric. To simplify the result tables, further evaluations only includes $E_b^\star$ and $E_t^\star$.

Table 6.3 exhibits a clear conclusion : *The agnostic Bayes ensemble generalizes better than AMin.* Next, when comparing against MSVM and Softmin, while the results are note statistically significant, the expected rank is in favor of both agnostic Bayes ensembles. Also, we note that MSVM is not significantly better than AMin.

It is well known that k-fold cross-validation provides a better estimate of the generalization performance of a learning algorithm than a single training/validation fold experiment. We thus performed another comparison for this setting. In this scenario, the agnostic Bayes method must now take into account the effective sample size ratio, as described in Section 6.5.1. Selected values ranges from 0.1 to 1 and were mainly concentrated between 0.3

Table 6.3: Comparison with the baseline models in the single training/validation split experiment (refer to Section 6.7.2 for notation).

	$E^\star_b$	**MSvm**	**SMin**	**AMin**	**E**[rank]
$E^\star_t$	0.541	0.613	**0.787**	**0.911**	2.63 /5
$E^\star_b$	0.500	0.592	0.763	**0.905**	2.66 /5
MSvm	0.408	0.500	0.623	0.789	2.92 /5
SMin	0.237	0.377	0.500	0.759	3.19 /5
AMin	0.095	0.211	0.241	0.500	3.57 /5

and 0.6. The results are expressed in Table 6.4 and are similar to that of Table 6.3. Again, agnostic Bayes is significantly better than Argmin while MSVM is not.

Table 6.4: Comparison with the baseline models in the cross-validation experiment (refer to Section 6.7.2 for notation).

	$E^\star_t$	**MSvm**	**SMin**	**AMin**	**E**[rank]
$E^\star_b$	0.507	0.575	0.707	**0.840**	2.70 /5
$E^\star_t$	0.500	0.578	0.720	**0.840**	2.75 /5
MSvm	0.422	0.500	0.577	0.725	2.95 /5
SMin	0.280	0.423	0.500	0.682	3.12 /5
AMin	0.160	0.275	0.318	0.500	3.46 /5

6.7.4 Changing the Loss Function

The results from the last section clearly demonstrate the advantage of mixing models over selecting a single one. While the agnostic Bayes methods outperform the baselines, we saw that simply using a linear learning algorithm also exhibits good performances. But what happens when the loss function changes? For example, we cannot use MetaSVM for combining models on a regression task. We can adapt and use ridge regression but, since it minimizes the quadratic loss, it may not perform well if our task is to minimize the expected absolute difference loss (i.e., $\mathcal{L}(y, y') = |y - y'|$). In other words, to perform a linear combination of models, we have to redesign the learning algorithm for every loss functions. Moreover, some loss functions yield a non-convex optimization problem which requires some form of approximation, e.g., SVM uses the hinge loss in place of the zero-one loss. In contrast, the proposed agnostic Bayes approach is designed to work with any loss function.

To outline the independence to the loss function of the agnostic Bayes methods, we performed experiments on regression tasks using both the quadratic loss and the absolute difference loss. We compared against the same baseline methods except for MetaSVM which was replaced by meta ridge regression (MRR) and its regularization parameter was selected by minimizing the appropriate loss function during cross validation. Table 6.5 presents the results obtained when using the quadratic loss function. While we worked with a totally different collection

Table 6.5: Comparison with the baseline models on regression tasks for the quadratic loss function (refer to Section 6.7.2 for notation).

	$E_t^\star$	MRR	SMin	AMin	E[rank]
$E_b^\star$	0.839	0.547	**0.929**	**0.992**	2.22 /5
$E_t^\star$	0.500	0.468	0.793	**0.986**	2.64 /5
MRR	0.532	0.500	0.554	0.809	2.88 /5
SMin	0.207	0.446	0.500	**0.992**	3.02 /5
AMin	0.014	0.191	0.008	0.500	4.23 /5

Table 6.6: Comparison with the baseline models on regression tasks for the absolute loss function (refer to Section 6.7.2 for notation).

	$E_t^\star$	SMin	MRR	AMin	E[rank]
$E_b^\star$	0.735	**0.953**	**0.859**	**0.995**	2.10 /5
$E_t^\star$	0.500	**0.932**	**0.821**	**0.995**	2.37 /5
SMin	0.068	0.500	0.769	**0.982**	3.06 /5
MRR	0.179	0.231	0.500	0.485	3.39 /5
AMin	0.005	0.018	0.515	0.500	4.08 /5

of data sets, the conclusions that follow from this experiment are surprisingly similar to the previous one. In this case, AMin is far down in ranking and the statistical significance of the observed differences are even stronger. Also, MRR is still performing relatively well.

Now, let us see what happens when we change the loss function to the absolute difference loss. Table 6.6 clearly shows an important degradation of MRR while the relative performances of the other methods are almost unchanged. In addition, the agnostic Bayes approach is now significantly better than the linear model. This clearly shows the importance of optimizing the appropriate loss function. Thus, justifying the usage of the agnostic Bayes ensemble.

6.8 Conclusion

We proposed the *agnostic Bayes* framework, which can be used to tackle the ubiquitous problem of model selection. This framework's central idea is to model the relationship between the hypotheses risks and observed empirical losses, without relying on assumptions about the true data-generating model. For one, this idea provides a new way of reasoning about machine learning problems. Also, the application to model selection has several desirable characteristics.

Generalization: The generalization performance of the agnostic Bayes ensemble is significantly better than just selecting the model minimizing the empirical expected loss. Also, our expected rank is systematically higher than any other evaluated methods on all experiments.

Flexibility: While most existing model selection algorithms are limited to a particular loss function, the agnostic Bayes ensemble can be used with any loss function. Also, our experiments showed how optimizing with the wrong loss function can be detrimental.

Speed: The bootstrap algorithm is simple to implement and has a linear computational complexity in the size of the data set. When measuring the learning speed, we observed that the bootstrap algorithm can be several thousand times faster than MetaSVM.

Acknowledgement

Thanks to Calcul Québec for providing support and access to Colosse's high performance computer grid. This work was supported by NSERC Discovery Grants 122405 (M. M.) and 262067 (F. L.).

Chapter 7

Third Paper Presentation

7.1 Details

Sequential Model-Based Ensemble Optimization
Alexandre Lacoste, Hugo Larochelle, François Laviolette, and Mario Marchand
In *Proceedings of the 30th Conference on Uncertainty in Artificial Intelligence*, AUAI Press, 2014.

7.2 Early Version

An early version of this work was presented at an ICML workshop in June 2014, and can be found in Appendix B.

Sequential Model-Based Ensemble Optimization
Alexandre Lacoste, Hugo Larochelle, François Laviolette, and Mario Marchand
In *ICML AutoML Workshop*, June 2014.

7.3 Context

A learning algorithm having d_{hp} continuous hyperparameter variables will have an infinite uncountable set Γ. To be computationally tractable, it is common to use a grid search approach, where each variable is discretized by assigning a finite set of values to each hyperparameter variable. This yields finite set Γ, which can now be explored through exhaustive search, but the size of Γ still increases at an exponential rate with d_{hp} making this approach impractical for more than 2 hyperparameters. Bergstra and Bengio [2012] improved this grid search approach by simply using a random search over the hyperparameter space. While this corresponds to significant improvement, the desire to explore more exotic learning algorithms requires an even more efficient approach.

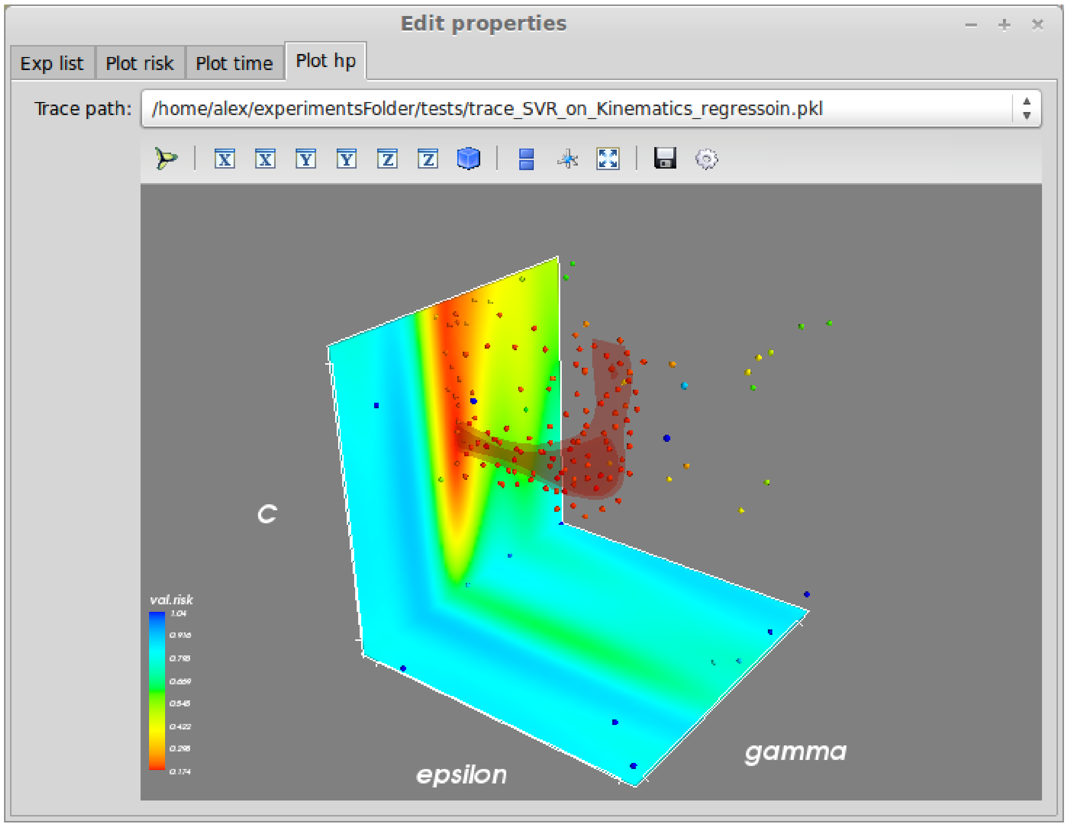
Recently, new optimization techniques were successfully applied to hyperparameter optimization [Bergstra et al., 2011, Snoek et al., 2012]. This unleashed the potential to explore complex hyperparameter spaces. In Bergstra et al. [2013], they perform optimization over hundreds of hyperparameter variables. While this helps finding a more appropriate model, by selecting a single one in such hyperparameter space, we definitely are exposed to significant overfitting.

7.4 Contributions

Model averaging is a good way to approach this overfitting challenge. However, most ensemble methods available in the literature are designed for a finite and fixed hyperparameter space. Luckily, we found that the agnostic Bayes approach can be adapted for this sequential optimization procedure. In this work, we propose an efficient online algorithm for computing the ensemble while iteratively searching the hyperparameter space. Since the training and evaluation time for a given hyperparameter configuration is usually significantly longer than the time required for building the ensemble, the computational cost is negligible. Experiments on a testing data set shows that the risk of the ensemble decreases faster over iterations than simply selecting the best predictor observed so far. Overall, this gives a fast and simple algorithm that can be used with any loss function $\mathcal{L}$.

7.5 Comments

Open source code implementing this agnostic Bayes algorithm is available at `https://github.com/recursix/spearmint-salad`. A graphical interface showing the algorithm in action is expressed in Figure 7.1.



Trace path: /home/alex/experimentsFolder/tests/trace_SVR_on_Kinematics_regressoin.pkl

Figure 7.1: Screenshot of spearmint-salad, a python tool implementing the agnostic Bayes ensemble algorithm. This 3D graph shows the validation performance for the three hyperparameters of support vector regression [Drucker et al., 1997] with a RBF kernel on the Kinematics data set from the UCI repository [Bache and Lichman, 2013]. Each point represents an explored hyperparameter configuration, where the colors represent its validation risk (red is low and blue is high). We clearly see that the optimization algorithm explored significantly more in the region of interest.

Chapter 8

Sequential Model-Based Ensemble Optimization

8.1 Abstract

One of the most tedious tasks in the application of machine learning is model selection, i.e. hyperparameter selection. Fortunately, recent progress has been made in the automation of this process, through the use of sequential model-based optimization (SMBO) methods. This can be used to optimize a cross-validation performance of a learning algorithm over the value of its hyperparameters. However, it is well known that ensembles of learned models almost consistently outperform a single model, even if properly selected. In this paper, we thus propose an extension of SMBO methods that automatically constructs such ensembles. This method builds on a recently proposed ensemble construction paradigm known as Agnostic Bayesian learning. In experiments on 22 regression and 39 classification data sets, we confirm the success of this proposed approach, which is able to outperform model selection with SMBO.

8.2 Introduction

The automation of hyperparameter selection is an important step towards making the practice of machine learning more approachable to the non-expert and increases its impact on data reliant sciences. Significant progress has been made recently, with many methods reporting success in tuning a large variety of algorithms Bergstra et al. [2011], Hutter et al. [2011], Snoek et al. [2012], Thornton et al. [2013]. One successful general paradigm is known as Sequential Model-Based Optimization (SMBO). It is based on a process that alternates between the proposal of a new hyperparameter configuration to test and the update of an adaptive model of the relationship between hyperparameter configurations and their holdout set performances. Thus, as the model learns about this relationship, it increases its ability to suggest improved hyperparameter configurations and gradually converges to the best solution.

While finding the single best model configuration is useful, better performance is often obtained by, instead, combining several (good) models into an ensemble. This was best illustrated by the winning entry of the Netflix competition, which combined a variety of models [Bell et al., 2007]. Even if one concentrates on a single learning algorithm, combining models produced by using different hyperparameters is also helpful. Intuitively, models with comparable performances are still likely to generalize differently across the input space and produce different patterns of errors. By averaging their predictions, we can hope that the majority of models actually perform well on any given input and will move the ensemble towards better predictions globally, by dominating the average. In other words, the averaging of several comparable models reduces the variance of our predictor compared to each individual in the ensemble, while not sacrificing too much in terms of bias.

However, constructing such ensembles is just as tedious as performing model selection and at least as important in the successful deployment of machine-learning-based systems. Moreover, unlike the model selection case for which SMBO can be used, no comparable automatic ensemble construction methods have been developed thus far. The current methods of choice remain trial and error or exhaustive grid search for exploring the space of models to combine, followed by a selection or weighting strategy which is often an heuristic. One exception is the work of Thornton et al. [2013], which can support the construction of ensembles, but only of up to 5 models.

In this paper, we propose a method for leveraging the recent research on SMBO in order to generate an ensemble of models, as opposed to the single best model. The proposed approach builds on the Agnostic Bayes framework [Lacoste et al., 2014b], which provides a successful strategy for weighting a predetermined and finite set of models (already trained) into an ensemble. Using a successful SMBO method, we show how we can effectively generalize this framework to the case of an infinite space of models (indexed by its hyperparameter space). The resulting method is simple and highly efficient. Our experiments on 22 regression and 39 classification data sets confirm that it outperforms the regular SMBO model selection method.

The paper develops as follows. First, we describe SMBO and its use for hyperparameter selection (Section 8.3). We follow with a description of the Agnostic Bayes framework and present a bootstrap-based implementation of it (Section 8.4). Then, we describe the proposed algorithm for automatically constructing an ensemble using SMBO (Section 8.5). Finally, related work is discussed (Section 8.6) and the experimental comparisons are presented (Section 8.7).

8.3 Hyperparameter Selection with SMBO

Let us first lay down the notation we will be using to describe the task of model selection for a machine learning algorithm. In this setup, a task D corresponds to a probability distribution over the input-output space $\mathcal{X} \times \mathcal{Y}$. Given a set of examples $S \sim D^m$ (which will be our holdout validation set), the objective is to find, among a set $\mathcal{H}$, the *best* function $h^\star : \mathcal{X} \to \mathcal{Y}$. In general, $\mathcal{H}$ can be any set and we refer to a member as a predictor. In the context of hyperparameter selection, $\mathcal{H}$ corresponds to the set of models trained on a training set $T \sim D^n$ (disjoint from S), for different configurations of the learning algorithm's hyperparameters γ. Namely, let $\mathcal{A}_\gamma$ be the learning algorithm with a hyperparameter configuration $\gamma \in \Gamma$, we will note $h_\gamma = \mathcal{A}_\gamma(T)$ the predictor obtained after training on T . The set $\mathcal{H}$ contains all predictors obtained from each $\gamma \in \Gamma$ when $\mathcal{A}_\gamma$ is trained on T, i.e. $\mathcal{H} \overset{\text{def}}{=} \{h_\gamma \mid \gamma \in \Gamma\}$.

To assess the quality of a predictor, we use a loss function

$$\mathcal{L} : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R},$$

that quantifies the penalty incurred when h_γ predicts $h_\gamma(x)$ while the true target is y. Then, we can define the risk $R_D(h_\gamma)$ as being the expected loss of h_γ on task D, i.e. $R_D(h_\gamma) \overset{\text{def}}{=} \underset{x,y\sim D}{\mathbf{E}} \left[\mathcal{L}\left(h_\gamma(x), y\right)\right]$. Finally, the *best*[1] function is simply the one minimizing the risk, i.e.

$$h^\star \overset{\text{def}}{=} \underset{h_\gamma \in \mathcal{H}}{\operatorname{argmin}}\ R_D(h_\gamma).$$

Here, estimating $h^\star$ thus corresponds to hyperparameter selection.

For most of machine learning history, the state of the art in hyperparameter selection has been testing a list of predefined configurations and selecting the best according to the loss function $\mathcal{L}$ on some holdout set of examples S. When a learning algorithm has more than one hyperparameter, a grid search is required, forcing $|\Gamma|$ to grow exponentially with the number of hyperparameters. In addition, the search may yield a suboptimal result when the minimum lies outside of the grid or when there is not enough computational power for an appropriate grid resolution. Recently, randomized search has been advocated as a better replacement to grid search [Bergstra and Bengio, 2012]. While it tends to be superior to grid search, it remains inefficient since its search is not informed by results of the sequence of hyperparameters that are tested.

To address these limitations, there has been an increasing amount of work on automatic hyperparameter optimization [Bergstra et al., 2011, Hutter et al., 2011, Snoek et al., 2012, Thornton et al., 2013]. Most rely on an approach called sequential model based optimization (SMBO). The idea consists in treating $R_S(h_\gamma) \overset{\text{def}}{=} f(\gamma)$ as a learnable function of γ, which we can learn from the observations $\{(\gamma_i, R_S(h_{\gamma_i}))\}$ collected during the hyperparameter selection process.

[1]The best solution may not be unique but any of them are equally good.

We must thus choose a model family for f. A common choice is a Gaussian process (GP) representation, which allows us to represent our uncertainty about f, i.e. our uncertainty about the value of $f(\gamma^*)$ at any unobserved hyperparameter configuration γ^*. This uncertainty can then be leveraged to determine an *acquisition function* that suggests the most promising hyperparameter configuration to test next.

Namely, let functions $\mu : \Gamma \to \mathbb{R}$ and $K : \Gamma \times \Gamma \to \mathbb{R}$ be the mean and covariance kernel functions of our GP over f. Let us also denote the set of the M previous evaluations as

$$\mathcal{R} \stackrel{\text{def}}{=} \{(\gamma_i, R_S(h_{\gamma_i}))\}_{i=1}^{M} \tag{8.1}$$

where $R_S(h_{\gamma_i})$ is the empirical risk of h_{γ_i} on set S, i.e. the holdout set error for hyperparameter γ.

The GP assumption on f implies that the conditional distribution $p(f(\gamma^*)|\mathcal{R})$ is Gaussian, that is

$$\begin{aligned} &p(f(\gamma^*)|\mathcal{R}) = \mathcal{N}(f(\gamma^*); \mu(\gamma^*;\mathcal{R}), \sigma^2(\gamma^*;\mathcal{R}), \\ &\mu(\gamma^*;\mathcal{R}) \stackrel{\text{def}}{=} \mu(\gamma^*) + \mathbf{k}^\top \mathbf{K}^{-1}(\mathbf{r} - \boldsymbol{\mu}), \\ &\sigma^2(\gamma^*;\mathcal{R}) \stackrel{\text{def}}{=} K(\gamma^*, \gamma^*) - \mathbf{k}^\top \mathbf{K}^{-1}\mathbf{k} \end{aligned}$$

where $\mathcal{N}(f(\gamma^*); \mu(\gamma^*;\mathcal{R}), \sigma^2(\gamma^*;\mathcal{R})$ is the Gaussian density function with mean $\mu(\gamma^*;\mathcal{R})$ and variance $\sigma^2(\gamma^*;\mathcal{R})$. We also have vectors

$$\begin{aligned} \boldsymbol{\mu} &\stackrel{\text{def}}{=} [\mu(\gamma_1), \ldots, \mu(\gamma_M)]^\top, \\ \mathbf{k} &\stackrel{\text{def}}{=} [K(\gamma^*, \gamma_1), \ldots, K(\gamma^*, \gamma_M)]^\top, \\ \mathbf{r} &\stackrel{\text{def}}{=} [R_S(h_{\gamma_1}), \ldots, R_S(h_{\gamma_M})]^\top, \end{aligned}$$

and matrix $\mathbf{K}$ is such that $\mathbf{K}_{ij} = K(\gamma_i, \gamma_j)$.

There are several choices for the acquisition function. One that has been used with success is the one maximizing the *expected improvement*:

$$\text{EI}(\gamma^*;\mathcal{R}) \stackrel{\text{def}}{=} \text{E}\left[\max\{r_{\text{best}} - f(\gamma^*), 0\}|\mathcal{R}\right] \tag{8.2}$$

which can be shown to be equal to

$$\sigma^2(\gamma^*;\mathcal{R})\left(d(\gamma^*;\mathcal{R})\Phi(d(\gamma^*;\mathcal{R})) + \mathcal{N}(d(\gamma^*;\mathcal{R}), 0, 1)\right) \tag{8.3}$$

where Φ is the cumulative distribution function of the standard normal and

$$\begin{aligned} r_{\text{best}} &\stackrel{\text{def}}{=} \min_i R_S(h_{\gamma_i}), \\ d(\gamma^*;\mathcal{R}) &\stackrel{\text{def}}{=} \frac{r_{\text{best}} - \mu(\gamma^*;\mathcal{R})}{\sigma(\gamma^*;\mathcal{R})}. \end{aligned}$$

The acquisition function thus maximizes Equation 8.3 and returns its solution. This optimization can be performed by gradient ascent initialized at points distributed across the hyperparameter space according to a Sobol sequence, in order to maximize the chance of finding a global optima. One advantage of expected improvement is that it directly offers a solution to the exploration-exploitation trade-off that hyperparameter selection faces.

An iteration of SMBO requires fitting the GP to the current set of tested hyperparameters $\mathcal{R}$ (initially empty), invoking the acquisition function, running the learning algorithm with the suggested hyperparameters and adding the result to $\mathcal{R}$. This procedure is expressed in Algorithm 1. Fitting the GP corresponds to learning the mean and covariance functions hyperparameters to the collected data. This can be performed either by maximizing the data's marginal likelihood or defining priors over the hyperparameters and sampling from the posterior using sampling (see Snoek et al. [2012] for more details).

Algorithm 1 SMBO Hyperparameter Optimization with GPs

```
$\mathcal{R} \leftarrow \{\}$
for $k \in \{1, 2, \ldots, M\}$ do
    $\gamma \leftarrow \text{SMBO}(\mathcal{R})$ {Fit GP and maximize EI}
    $h_\gamma \leftarrow \mathcal{A}_\gamma(T)$ {Train with suggested $\gamma$}
    $\mathcal{R} \leftarrow \mathcal{R} \cup \{(\gamma, R_S(h_\gamma))\}$ {Add to collected data}
end for
$\gamma^* \leftarrow \operatorname*{argmin}_{(\gamma, R_S(h_\gamma)) \in \mathcal{R}} R_S(h_\gamma)$
return $h_{\gamma^*}$
```

While SMBO hyperparameter optimization can produce very good predictors, it can also suffer from overfitting on the validation set, especially for high-dimensional hyperparameter spaces. This is in part why an ensemble of predictors are often preferable in practice. Properly extending SMBO to the construction of ensembles is, however, not obvious. Here, we propose one such successful extension, building on the framework of Agnostic Bayes learning, described in the next section.

8.4 Agnostic Bayes

In this section, we offer a brief overview of the Agnostic Bayes learning paradigm presented in Lacoste et al. [2014b] and serving as a basis for the algorithm we present in this paper. Agnostic Bayes learning was used in Lacoste et al. [2014b] as a framework for successfully constructing ensembles when the number of predictors in $\mathcal{H}$ (i.e. the potential hyperparameter configurations Γ) was constrained to be finite (e.g. by restricting the space to a grid). In our context, we can thus enumerate the possible hyperparameter configurations from γ_1 to $\gamma_{|\Gamma|}$. This paper will generalize this approach to the infinite case later.

Agnostic Bayes learning attempts to directly address the problem of inferring what is the *best* function $h^\star$ in $\mathcal{H}$, according to the loss function $\mathcal{L}$. It infers a posterior $p_{h^\star}(h_\gamma|S)$, i.e. a distribution over how likely each member of $\mathcal{H}$ is the best predictor. This is in contrast with standard Bayesian learning, which implicitly assumes that $\mathcal{H}$ contains the true data-generating model and infers a distribution for how likely each member of $\mathcal{H}$ has generated the data (irrespective of what the loss $\mathcal{L}$ is). From $p_{h^\star}(h_\gamma|S)$, by marginalizing $h^\star$, we obtain a probabilistic estimate for the best prediction $y^\star \overset{\text{def}}{=} h^\star(x)$

$$p_{y^\star}(y|x,S) = \sum_{\gamma\in\Gamma} p_{h^\star}(h_\gamma|S)\,\mathrm{I}[h_\gamma(x) = y].$$

Finally, to commit to a final prediction, for a given x, we use the most probable answer[2]. This yields the following ensemble decision rule

$$E^\star(x) \overset{\text{def}}{=} \operatorname*{argmax}_{y\in\mathcal{Y}} p_{y^\star}(y|x,S). \tag{8.4}$$

To estimate $p_{h^\star}(h_\gamma|S)$, Agnostic Bayes learning uses the set of losses $l_{\gamma,i} \overset{\text{def}}{=} \mathcal{L}(h_\gamma(x_i), y_i)$ of each example $(x_i, y_i) \in S$ as evidence for inference. In Lacoste et al. [2014b], a few different approaches are proposed and analyzed. A general strategy is to assume a joint prior $p(\mathbf{r})$ over the risks $r_\gamma \overset{\text{def}}{=} R_D(h_\gamma)$ of all possible hyperparameter configurations and choose a joint observation $p(l_{\gamma,i}\ \forall\gamma \in \Gamma|\mathbf{r})$ for the losses. From Bayes rule, we obtain the posterior $p(\mathbf{r}|S)$ from which we can compute

$$p_{h^\star}(h_\gamma|S) \quad = \quad \mathrm{E}_{\mathbf{r}}\left[\mathrm{I}[r_\gamma < r_{\gamma'}, \forall\gamma' \neq \gamma]|S\right] \tag{8.5}$$

with a Monte Carlo estimate. This would result in repeatedly sampling from $p(\mathbf{r}|S)$ and counting the number of times each γ has the smallest sampled risk r_γ to estimate $p_{h^\star}(h_\gamma|S)$. Similarly, samples from $p_{h^\star}(h_\gamma|S)$ could be obtained by sampling a risk vector $\mathbf{r}$ from $p(\mathbf{r}|S)$ and returning the predictor h_γ with the lowest sampled risk. The ensemble decision rule of Equation 8.4 could then be implemented by repeatedly sampling from $p_{h^\star}(h_\gamma|S)$ to construct the ensemble of predictors and using their average as the ensemble's prediction.

Among the methods explored in Lacoste et al. [2014b] to obtain samples from $p(\mathbf{r}|S)$, the bootstrap approach stands out for its efficiency and simplicity. Namely, to obtain a sample from $p(\mathbf{r}|S)$, we sample with replacement from S to obtain S' and return the vector of empirical risks

$$[R_{S'}(h_{\gamma_1}), \ldots, R_{S'}(h_{\gamma_{|\Gamma|}})]^\top$$

as a sample. While bootstrap only serves as a "poor man's" posterior, it can be shown to be statistically related to a proper model with Dirichlet priors and its empirical performance was shown to be equivalent [Lacoste et al., 2014b].

[2]As noted in Lacoste et al. [2014b], $p_{y^\star}(y|x,S)$ does not correspond to the probability of observing y given x and cannot be used with the optimal Bayes theory, thus justifying the usage of the most probable answer

When the bootstrap method is used to obtain samples from $p_{h^\star}(h_\gamma|S)$, the complete procedure for generating each ensemble member can be summarized by

$$\widetilde{h^\star} = \operatorname*{argmin}_{\gamma \in \Gamma} R_{S'}(h_\gamma), \tag{8.6}$$

where $\widetilde{h^\star}$ corresponds to a sample from $p_{h^\star}(h_\gamma|S)$. In this work, we use SMBO to address the optimization part. Thus, we can now extend to an uncountable set Γ.

This method can be seen as applying bagging on the validation set instead of the training set. However, we stand by the Agnostic Bayes theory since it offers a strong theoretical backbone to bagging as well as few refinements. Most importantly, the normal assumption of Section 3.3 in Lacoste et al. [2014b] suggests that methods based on the covariance of the predictions such as ensemble pruning [Zhang et al., 2006] and MinCq [Roy et al., 2011] are simply different approximations of this idea. This connection allows us to be confident that the fast and simple algorithm we propose in this paper is at least equivalent in generalization performance to other state of the art ensemble methods. Finally, this claim is supported by the strong experimental section of Lacoste et al. [2014b].

8.5 Agnostic Bayes Ensemble with SMBO

We now present our proposed method for automatically constructing an ensemble, without having to restrict Γ (or, equivalently $\mathcal{H}$) to a finite subset of hyperparameters.

As described in Section 8.4, to sample a predictor from the Agnostic Bayes bootstrap method, it suffices to obtain a bootstrap S' from S and solve the optimization problem of Equation 8.6. In our context where $\mathcal{H}$ is possibly an infinite set of models trained on the training set T for any hyperparameter configuration γ, Equation 8.6 corresponds in fact to hyperparameter optimization where the holdout set is S' instead of S.

This suggests a simple procedure for building an ensemble of N predictors according to Agnostic Bayes i.e., that reflects our uncertainty about the true best model $h^\star$. We could repeat the full SMBO hyperparameter optimization process N times, with different bootstrap S'_j, for $j \in \{1, 2, \ldots, N\}$. However, for large ensembles, performing N runs of SMBO can be computationally expensive, since each run would need to train its own sequence of models.

We can notice however that predictors are always trained on the same training set T, no matter in which run of SMBO they were trained on. We propose a handy trick that exploits this observation to greatly accelerate the construction of the ensemble by almost a factor of N. Specifically, we propose to simultaneously optimize all N problems in a round-robin fashion. Thus, we maintain N different histories of evaluation $\mathcal{R}_j$, for $j \in \{1, 2, \ldots, N\}$ and when a new predictor $h_\gamma = \mathcal{A}_\gamma(T)$ is obtained, we update all $\mathcal{R}_j$ with $(\gamma, R_{S'_j}(h_\gamma))$. Notice that the different histories $\mathcal{R}_j$ contain the empirical risks on different bootstrap holdout sets, but they are all updated at the cost of training only a single predictor. Also, to avoid recalculating multiple times $\mathcal{L}(h_\gamma(x_i), y_i)$, these values can be cached and shared in the computation of each $\mathcal{R}_j$. This leaves the task of updating all $\mathcal{R}_j$ insignificant compared to the computational time usually required for training a predictor. This procedure is detailed in Algorithm 2.

Algorithm 2 Agnostic Bayes Ensemble with SMBO

```
for $j \in \{1, 2, \ldots, N\}$ do
    $\mathcal{R}_j \leftarrow \{\}$
    $S'_j \leftarrow \text{bootstrap}(S)$
end for

$\mathcal{E} \leftarrow \{\}$ {Will contain all trained predictors}
for $k \in \{1, 2, \ldots, M\}$ do
    $v \leftarrow (k-1) \,\text{modulo}\, N + 1$
    $\gamma \leftarrow \text{NEXT}(\mathcal{R}_v)$ {Selects the next $\gamma$ to explore}
    $h_\gamma \leftarrow \mathcal{A}_\gamma(T)$
    $\mathcal{E} \leftarrow \mathcal{E} \cup \{h_\gamma\}$
    for $j \in \{1, 2, \ldots, N\}$ do
        $\mathcal{R}_j \leftarrow \mathcal{R}_j \cup \left\{\left(\gamma, R_{S'_j}(h_\gamma)\right)\right\}$
    end for
end for

$\mathcal{H}' \leftarrow \{\}$ {Will contain $N$ selected predictors}
for $j \in \{1, 2, \ldots, N\}$ do
    $h_j \leftarrow \underset{h_\gamma \in \mathcal{E}}{\text{argmin}}\; R_{S'_j}(h_\gamma)$
    $\mathcal{H}' \leftarrow \mathcal{H}' \cup \{h_j\}$
end for

$p_{h^\star}(h_\gamma|S) = \text{Uniform}(\mathcal{H}')$
return $p_{h^\star}(h_\gamma|S)$
```

By updating all $\mathcal{R}_j$ at the same time, we *trick* each SMBO run by updating its history with points it did not suggest. This implies that the GP model behind each SMBO run will be able to condition on more observations then it would if the runs had been performed in isolation. This can only benefit the GPs and improve the quality of their suggestions.

While Algorithm 2 is sequential, it can be easily adapted to the parallelized version of SMBO presented in Snoek et al. [2012]. Also, it can be extended to use cross validation, based on the method developed in [Lacoste et al., 2014b].

In our experiments, we fix $N = \lfloor \frac{M}{2} \rfloor$. This maximizes the number of samples used to estimate $p_{y^\star}(y|x, S)$ while ensuring at least one SMBO step with a reasonably large history for each bootstrap. When the prediction time on the test set is a concern, we suggest to choose $N \approx 10$. We observed that it was usually enough to obtain most of the generalization gain.

Finally, since $p_{y^\star}(y|x, S)$ is only estimated, finding the maximum, as requested in Equation 8.4, requires some form of density estimation. In the case of classification, we simply use the most probable class. However, in the regression case, we fit a normal distribution[3]. Thus, the maximum coincide with the average prediction. For a more complex $\mathcal{Y}$, such as in structured output tasks, we recommend to use an appropriate density estimation and increase the number of samples N.

8.6 Related Work

In the Bayesian learning literature, a common way of dealing with hyperparameters in probabilistic predictors is to define hyperpriors and perform posterior inference to integrate them out. This process often results in also constructing an ensemble of predictors with different hyperparameters, sampled from the posterior. Powerful MCMC methods have been developed in order to accommodate for different types of hyperparameter spaces, including infinite spaces.

However, this approach requires that the family of predictors in question be probabilistic in order to apply Bayes rule. Moreover, even if the predictor family is probabilistic, the construction of the ensemble will entirely ignore the nature of the loss function that determines the measure of performance. The comparative advantage of the proposed Agnostic Bayes SMBO approach is thus that it can be used for any predictor family (probabilistic or not) and is loss-sensitive.

On the other hand, traditional ensemble methods such as Laviolette et al. [2011], Kim and Ghahramani [2012], and Zhang et al. [2006] require a predefined set of models and are not straightforward to adapt to an infinite set of models.

8.7 Experiments

We now compare the SMBO ensemble approach (ESMBO) to three alternative methods for building a predictor from a machine learning algorithm with hyperparameters:

- A single model, whose hyperparameters were selected by hyperparameter optimization with SMBO (SMBO).
- A single model, whose hyperparameters were selected by a randomized search (RS), which in practice is often superior to grid search [Bergstra and Bengio, 2012].
- An Agnostic Bayes ensemble constructed over a randomly selected set of hyperparameters (ERS).

[3]It is also possible to use a more elaborated method, such as kernel density estimation.

Both ESMBO and SMBO used GP models of the holdout risk, with hyperparameters trained to maximize the marginal likelihood. A constant was used for the mean function, while the Matérn 5/2 kernel was used for the covariance function, with length scale parameters. The GP's parameters were obtained by maximizing the marginal likelihood and a different length scale was used for each dimension[4].

Each method is allowed to evaluate 150 hyperparameter configurations. To compare their performances, we perform statistical tests on several different hyperparameter spaces over two different collections of data sets.

8.7.1 Hyperparameter Spaces

Here, we describe the hyperparameter spaces of all learning algorithms we employ in our experiments. Except for a custom implementation of the multilayer perceptron, we used scikit-learn[5] for the implementation of all other learning algorithms.

Support Vector Machine We explore the soft margin parameter C for values ranging from 10^{-2} to 10^{3} on a logarithmic scale. We use the RBF kernel $K(x, x') = e^{\gamma ||x-x'||_2^2}$ and explore values of γ ranging from 10^{-5} to 10^{3} on a logarithmic scale.

Support Vector Regressor We also use the RBF kernel and we explore the same values as for the Support Vector Machine. In addition, we explore the ϵ-tube parameter [Drucker et al., 1997] for values ranging between 10^{-2} and 1 on a logarithmic scale.

Random Forest We fix the number of trees to 100 and we explore two different ways of producing them: either the original Breiman [2001] method or the extremely randomized trees method of Geurts et al. [2006]. We also explore the choice of bootstrapping or not the training set before generating a tree. Finally, the ratio of randomly considered features at each split for the construction of the trees is varied between 10^{-4} and 1 on a linear scale.

Gradient Boosted Classifier This is a tree-based algorithm using boosting [Friedman, 2001]. We fix the set of weak learners to 100 trees and take the maximum depth of each tree to be in $\{1, 2, \ldots, 15\}$. The learning rate ranges between 10^{-2} and 1 on a logarithmic scale. Finally, the ratio of randomly considered features at each split for the construction of the trees varies between 10^{-3} and 1 on a linear scale.

Gradient Boosted Regressor We use the same parameters as for Gradient Boosted Classifier except that we explore a convex combination of the least square loss function and the

[4]We used the implementation provided by spearmint: `https://github.com/JasperSnoek/spearmint`
[5]`http://scikit-learn.org/`

least absolute deviation loss function. We also fix the ratio of considered features at each split to 1.

Multilayer Perceptron We use a 2 hidden layers perceptron with tanh activation function and a softmax function on the last layer. We minimize the negative log likelihood using the L-BFGS algorithm. Thus there is no learning rate parameter. However, we used a different L2 regularizer weight for each of the 3 layers with values ranging from 10^{-5} to 100 on a logarithmic scale. Also, the number of neurons on each layer can take values in $\{1, 2, \ldots, 100\}$. In total, this yields a 5 dimensional hyperparameter space.

8.7.2 Comparing Methods on Multiple Data Sets

To assess the generalization performances, we use a separate test set S^{test}, which is obtained by randomly partitioning the original data set. More precisely, we use the ratios 0.4, 0.3, and 0.3 for T, S and S^{test} respectively[6]. However, testing on a single data set is insufficient to testify the quality of a method that is meant to work across different tasks. Hence, we evaluate our methods on several data sets using metrics that do not assume commensurability across tasks [Demšar, 2006b]. The metrics of choice are thus the expected rank and the pairwise winning frequency. Let $\mathcal{A}_i(T_j, S_j)$ be either one of our $K = 4$ model selection/ensemble construction algorithms run on the j^{th} data set, with training set T_j and validation set S_j. When comparing K algorithms, the rank of (best or ensemble) predictor $h_i = \mathcal{A}_i(T_j, S_j)$ on test set S_j^{test} is defined as

$$\text{Rank}_{h_i, S^{\text{test}}} \stackrel{\text{def}}{=} \sum_{l=1}^{K} I\left[R_{S_j^{\text{test}}}(h_l) \leq R_{S_j^{\text{test}}}(h_i)\right].$$

Then, the expected rank of the i^{th} method is obtained from the empirical average over the L data sets i.e., $\mathbf{E}[\text{R}]_i \stackrel{\text{def}}{=} \frac{1}{L}\sum_{j=1}^{L} \text{Rank}_{h_i, S_j^{\text{test}}}$. When comparing algorithm $\mathcal{A}_i$ against algorithm $\mathcal{A}_l$, the winning frequency[7] of $\mathcal{A}_i$ is

$$\rho_{i,l} \stackrel{\text{def}}{=} \frac{1}{L}\sum_{i=1}^{L} \text{I}[R_{S_j^{\text{test}}}(h_i) < R_{S_j^{\text{test}}}(h_l)]$$

In the case of the expected rank, lower is better and for the winning frequency, it is the converse. Also, when $K = 2$, $\mathbf{E}[\text{R}]_i = 1 + (1 - \rho_{i,l})$.

When the winning frequency $\rho_{i,l} > 0.5$, we say that method $\mathcal{A}_i$ is better than method $\mathcal{A}_l$. However, to make sure that this is not the outcome of chance, we use statistical tests such as the sign test and the Poisson Binomial test (PB test) [Lacoste et al., 2012]. The PB test derives a posterior distribution over $\rho_{i,l}$ and integrates the probability mass above 0.5,

[6]Is is also possible to perform cross-validation as mentioned in Lacoste et al. [2014b].

[7]We deal with ties by attributing 0.5 to each method except for the sign test where the sample is simply discarded.

Table 8.1: Significance Notation Used in Result Tables.

Meaning	**Symbol**	$\Pr(\mathbf{A} \succ \mathbf{B})$	**p-value**
Lightly significant	○	> 0.8	< 0.1
Significant	●	> 0.9	< 0.05
Highly significant	●	> 0.95	< 0.01

denoted as $\Pr(A \succ B)$. When $\Pr(A \succ B) > 0.9$, we say that it is significant. Similarly for the sign test, when the p-value is lower than 0.05, it corresponds to a significant result. To report more information, we also use other thresholds for lightly significant and highly significant as described in Table 8.1.

To build a substantial collection of data sets, we used the AYSU collection [Ulaş et al., 2009] coming from the UCI and the Delve repositories and we added the MNIST data set. We also converted the multiclass data sets to binary classification by either merging classes or selecting pairs of classes. The resulting benchmark contains 39 data sets. We have also collected 22 regression data sets from the Louis Torgo collection[8].

8.7.3 Table Notation

The result tables present the winning frequency for each pair of methods, where grayed out values represent redundant information. As a complement, we also add the expected rank of each method in the rightmost column and sort the table according to this metric. To report the conclusion of the sign test and the PB test, we use different symbols to reflect different level of significance. The exact notation is presented in Table 8.1. The first symbol reports the result of the PB test and the second one, the sign test. For more stable results, we average the values obtained during the last 15 iterations.

8.7.4 Analysis

Looking at the overall results over 7 different hyperparameter spaces in Table 8.2 and Table 8.3, we observe that ESMBO is never significantly outperformed by any other method and often outperforms the others. More precisely, it is either ranked first or tightly following ERS. Looking more closely, we see that the cases where ESMBO does not significantly outperform ERS concerns hyperparameter spaces of low complexity. For example, most hyperparameter configurations of Random Forest yield good generalization performances. Thus, these cases do not require an elaborate hyperparameter search method. On the other hand, when looking at more challenging hyperparameter spaces such as Support Vector Regression and Multilayer Perceptrons, we clearly see the benefits of combining SMBO with Agnostic Bayes.

[8]These data sets were obtained from the following source : `http://www.dcc.fc.up.pt/~ltorgo/Regression/DataSets.html`

Table 8.2: Pairwise Win Frequency For the 3 Different Regression Hyperparameter Spaces (Refer to Section 8.7.3 for the notation). Let us recall the acronyms. ESMBO: Ensemble of SMBO, ERS: ensemble of random search, RS: Random Search.

Support Vector Regressor

	ESMBO	ERS	SMBO	RS	E[rank]
ESMBO	0.50	0.66 •	0.82 ••	0.86 ••	1.66
ERS	0.34	0.50	0.50	0.77 ••	2.38
SMBO	0.18	0.50	0.50	0.64 °	2.68
RS	0.14	0.23	0.36	0.50	3.27

Gradient Boosting Regressor

	ERS	ESMBO	RS	SMBO	E[rank]
ERS	0.50	0.52	0.77 ••	0.86 ••	1.84
ESMBO	0.48	0.50	0.77 ••	0.91 ••	1.85
RS	0.23	0.23	0.50	0.42	3.12
SMBO	0.14	0.09	0.58	0.50	3.19

Random Forest

	ESMBO	ERS	SMBO	RS	E[rank]
ESMBO	0.50	0.53	0.76 ••	0.91 ••	1.80
ERS	0.47	0.50	0.72 ••	1.00 ••	1.81
SMBO	0.24	0.28	0.50	0.66	2.82
RS	0.09	0.00	0.34	0.50	3.57

As described in Section 8.5, ESMBO is alternating between N different SMBO optimizations and deviates from the natural sequence of SMBO. To see if this aspect of ESMBO can influence its convergence rate, we present a temporal analysis of the methods in Figure 8.1 and Figure 8.2. The left columns depict $\Pr(A \succ B)$ for selected pairs of methods and the right columns present the expected rank of each method over time.

A general analysis clearly shows that there is no significant degradation in terms of convergence speed. In fact, we generally observe the opposite. More precisely, looking at Pr(ESMBO $\succ$ SMBO), the green curve of the left columns, it usually reaches a significantly better state right at the beginning or within the first few iterations. A notable exception to that trend occurs with the Multiplayer Perceptrons, where SMBO is significantly better than ESMBO for a few iterations at the beginning. Then, it gets quickly outperformed by ESMBO.

8.8 Conclusion

We described a successful method for automatically constructing ensembles without requiring hand-selection of models or a grid search. The method can adapt the SMBO hyperparameter optimization algorithm so that it can produce an ensemble instead of a single model. Theoretically, the method is motivated by an Agnostic Bayesian paradigm which attempts to construct ensembles that reflect the uncertainty over which a model actually has the smallest

Table 8.3: Pairwise Win Frequency for the 4 Different Classification Hyperparameter Spaces (Refer to Section 8.7.3 for the notation). Let us recall the acronyms. ESMBO: Ensemble of SMBO, ERS: ensemble of random search, RS: Random Search.

Support Vector Machine

	ESMBO	RS	SMBO	ERS	E[rank]
ESMBO	0.50	0.54	0.55	0.56	2.35
RS	0.46	0.50	0.51	0.51	2.52
SMBO	0.45	0.49	0.50	0.53	2.54
ERS	0.44	0.49	0.47	0.50	2.59

Gradient Boosting Classifier

	ESMBO	ERS	RS	SMBO	E[rank]
ESMBO	0.50	0.51	0.59	0.65 ○●	2.25
ERS	0.49	0.50	0.59	0.64 ○○	2.28
RS	0.41	0.41	0.50	0.55	2.64
SMBO	0.35	0.36	0.45	0.50	2.83

Random Forest

	ERS	ESMBO	RS	SMBO	E[rank]
ERS	0.50	0.52	0.60 ○	0.64 ○●	2.24
ESMBO	0.48	0.50	0.60	0.67 ○●	2.25
RS	0.40	0.40	0.50	0.57	2.63
SMBO	0.36	0.33	0.43	0.50	2.89

Multilayer Perceptron

	ESMBO	SMBO	ERS	RS	E[rank]
ESMBO	0.50	0.57 ○	0.76 ●●	0.75 ●●	1.92
SMBO	0.43	0.50	0.68 ○●	0.68 ○●	2.21
ERS	0.24	0.32	0.50	0.54	2.91
RS	0.25	0.32	0.46	0.50	2.96

true risk. The resulting method is easy to implement and comes with no extra computational cost at learning time. Its generalization performance and convergence speed are also dominant according to experiments on 22 regression and 39 classification data sets.

Acknowledgement

Thanks to Calcul Québec for providing support and access to Colosse's high performance computer grid. This work was supported by NSERC Discovery Grants 122405 (M. M.), 262067 (F. L.) and 418327 (H. L.).

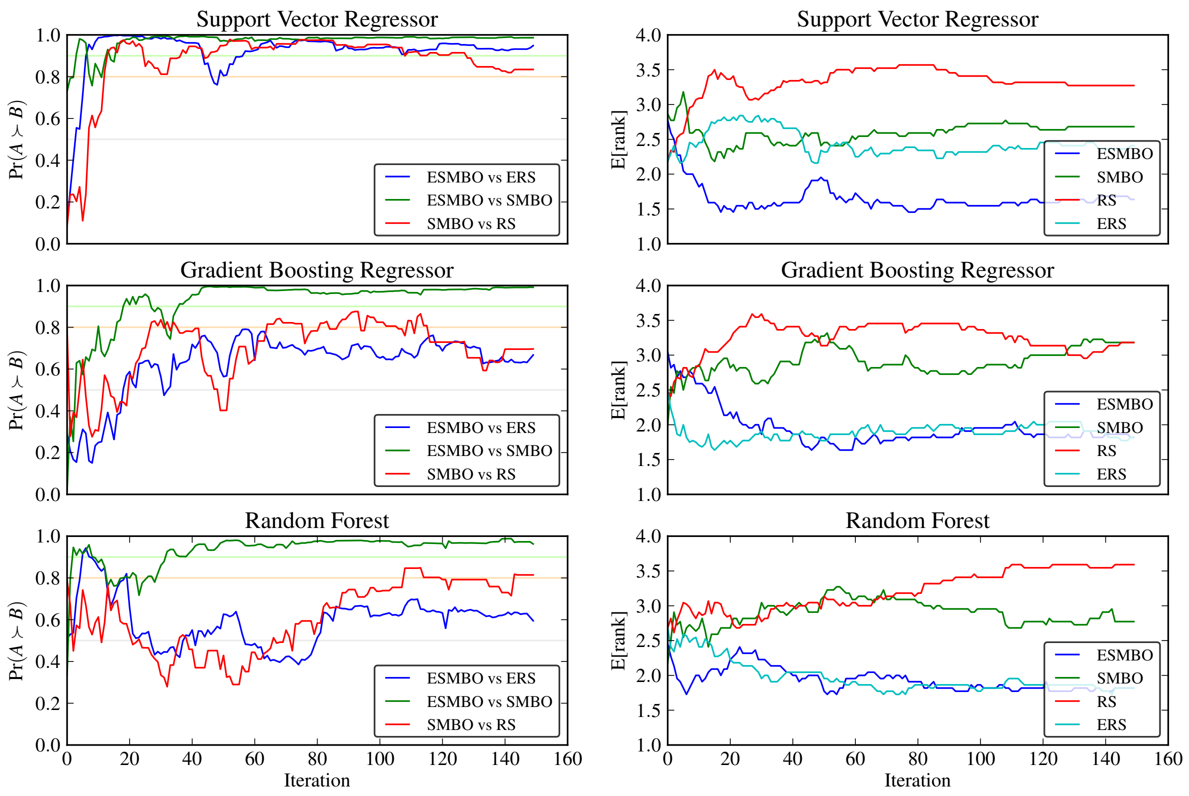


Figure 8.1: PB Probability and Expected Rank over Time for the 3 Regression Hyperparameter Spaces.

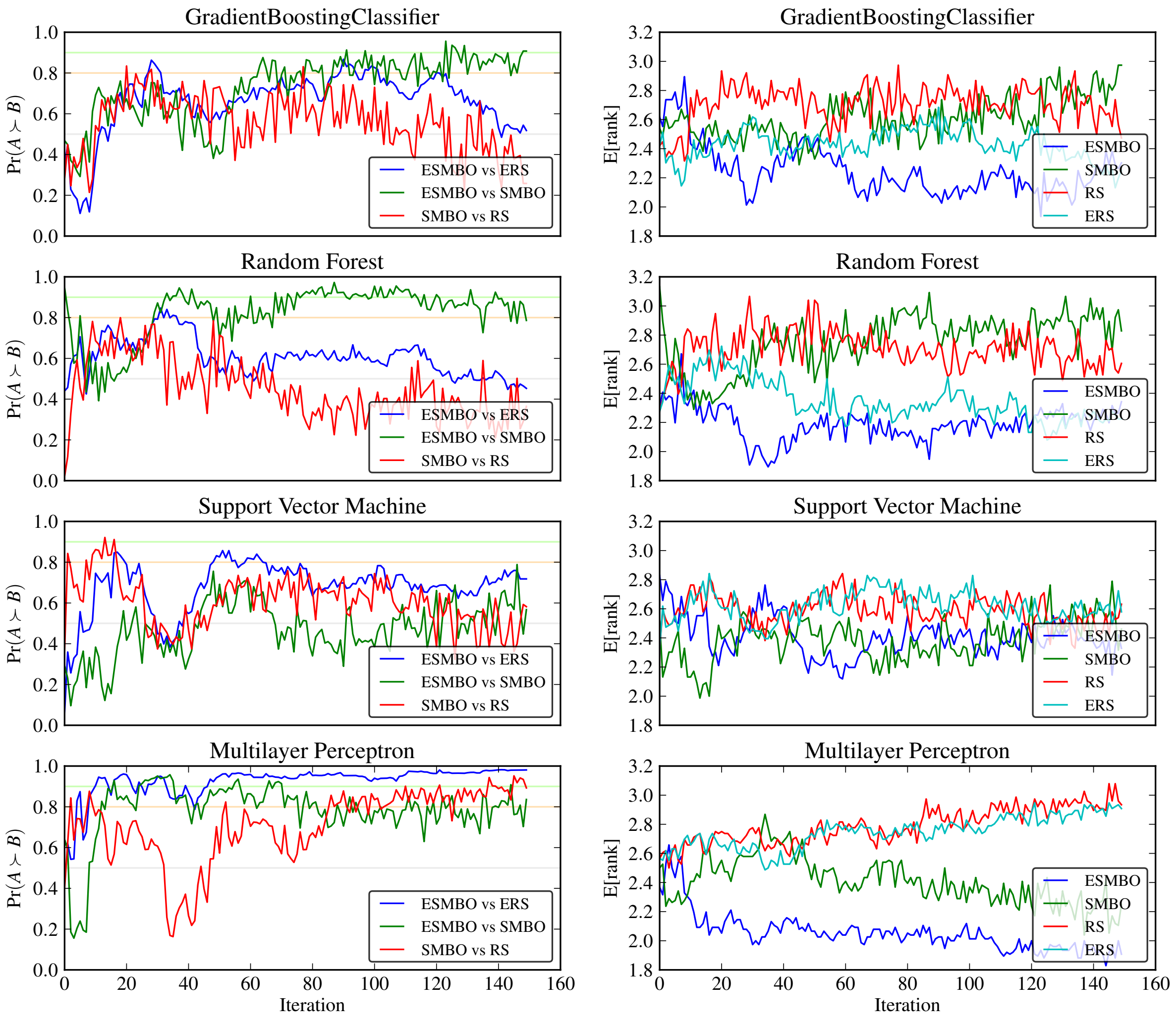


Figure 8.2: PB Probability and Expected Rank over Time for the 4 Classification Hyperparameter Spaces.

Chapter 9

Conclusion

In this thesis, we first exposed the basics of Bayesian modeling of machine learning and the basics of the PAC-Bayesian theory. While both of these approaches are based on fundamentally different assumptions, both agree that choosing a suitable prior and performing model averaging are key ingredients to a better generalization performance. On the other hand, the formulation of machine learning suggests that a learning algorithm must be as generic as possible. Hence, the notion of prior must be as vague as possible. For this reason, this thesis focuses on new model averaging approaches.

The Poisson Binomial Test

In this first work, we addressed the simplest case of the main element of the agnostic Bayes approach

$$p(h^\star = h|S).$$

Since it is limited to the case where $|\mathcal{H}| = 2$ and is used on the test set T, it can be written $p(R_D(h) < R_D(g)|T)$, where h and g are elements of $\mathcal{H}$. Also, by limiting it to the zero-one loss function, we were able to obtain a closed form solution.

This posterior distribution is then used in a greater model to address the probability that a learning algorithm is better than another one when evaluated on a collection of data sets sampled from a *context*. This gave rise to a Bayesian statistical significance test called the Poisson binomial test, and it is used to assess the reliability of all experiments in this thesis.

Agnostic Bayes

By generalizing $p(R_D(h) < R_D(g)|S)$ to any loss function and a set $\mathcal{H}$ of size greater than two, we were able to provide an estimation of the probability distribution $p(h^\star = h|S)$ for any supervised learning setup. Using this during the validation procedure allows us to integrate out the uncertainty about which model is truly the best one. Extensive experiments confirm the generalization gain of this approach.

Sequential Model Based Ensemble Optimization

As opposed to most ensemble methods, the agnostic Bayes ensemble can be adapted to sequential optimization of hyperparameters. By doing so, we obtained a simple online algorithm for building the ensemble at any iterations with a negligible computational cost.

9.1 Future Work

9.1.1 Agnostic Bayes Forest

In this work, the agnostic Bayes approach was applied only at the validation phase to perform model averaging over the set of hyperparameters. This was mainly to demonstrate the effectiveness of this approach. While we worked with a finite set $\mathcal{H}$, some of the models developed in Section 6.4 can be extended to infinite $\mathcal{H}$. For example, to sample a predictor $\tilde{h}$ from the posterior $p(h^\star = h|S)$ with the bootstrap approach, it suffices to take a bootstrap S' of the training set S and minimize the empirical risk, *i.e.*, $\tilde{h} = \underset{h \in \mathcal{H}}{\operatorname{argmin}} R_{S'}(h)$. We also believe that the Dirichlet approach can be extended to the infinite set. This would give different avenues for exploring an agnostic Bayes variant of the random forest algorithm Breiman [2001].

9.1.2 Extending the Poisson Binomial Test

Statistical significance tests are made to quantify the amount of uncertainty in an experiment. Then, based on a threshold one can reject or accept a conclusion. Conventionally, these tests are based on p-values and the threshold is 0.05 (yielding a success chance of 19/20). This kind of approach is said to be *frequentist* and is much more common than a Bayesian test such as the one presented in Section 4. The main culprit for the lack of popularity of Bayesian tests is the notion of prior. It forces us to perform assumptions that would not be necessary otherwise. In our case, we chose an impartial prior that maximizes the entropy to avoid any bias. Despite this effort, the existence of the prior still bothers some scientists. Fortunately, in the light of a recent publication [Benavoli et al., 2014], I found that the notion of *imprecise prior* was developed to address this kind of issues. This notion simply states that one should define a set of possible priors while never committing to any of them. Then, when comes the time to compute an upper bound or a lower bound on the probability of some event, we simply choose the prior that would select the *safest* bound.

Appendix A

Model Averaging With Holdout Estimation of the Posterior Distribution

A.1 Abstract

The holdout estimation of the expected loss of one model is biased and noisy. Yet, practicians often rely on it to select the model to be used for further predictions. Repeating the learning phase with small variations of the training set reveals a variation on the selected model which then induces an important variation of the final test performances. Thus, we propose a small modification to the k-fold cross-validation that greatly enhances the generalization performances of the final predictor. Instead of using the empirical average of the validation losses to select a single model, we propose to use bootstrap to resample the validation losses (without retraining). The variations in the selected models induce a posterior distribution that is then used for model averaging. Comparing this novel approach to the classical cross-validation on 38 data sets with a significance test shows that it has higher generalization performance with probability over 0.9.

A.2 Introduction

In this paper, we work in the inductive learning paradigm where a task D corresponds to a probability distribution over $\mathcal{X} \times \mathcal{Y}$, the input-output space. Given a training set $S \sim D^m$, our objective is to find, among a set of hypothesis $\mathcal{H}$, a function $h^\star : \mathcal{X} \to \mathcal{Y}$ that minimizes the expected loss $R_D(h)$, where $R_D(h) \overset{\text{def}}{=} \underset{x,y \sim D}{\mathbf{E}} l\left(h(x), y\right)$ and where $l : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}$ is the loss incurred when h predicts $h(x)$ while the true output for x is y.

Since we only observe $S \sim D^m$ and not D, it is generally not possible to obtain $h^\star$ with finite values of m. For a given x, Bayes' decision theory suggests to use $y^\star = \underset{\widehat{y}\in\mathcal{Y}}{\operatorname{argmin}} \underset{y\sim\Pr(\cdot\mid x,S)}{\mathbf{E}} l\left(\widehat{y}, y\right)$, where $\Pr\left(y \mid x, S\right) = \underset{h\sim\Pr(\cdot\mid S)}{\mathbf{E}} \Pr\left(y \mid h, x\right)$ and where $\Pr\left(h \mid S\right)$ is obtained using Bayes' update rule with a prior $P(h)$ over $\mathcal{H}$. Since $|\mathcal{H}|$ is either infinite or too large to be computationally tractable for most practical applications, the community of machine learning has been concerned in finding efficient algorithms for some subset of $\mathcal{H}$ that exhibit certain mathematical properties. Let $\gamma \in \Gamma$ be a tuple of hyperparameters describing a particular model. Then, using model averaging, we can factorize the problem and combine all learning algorithms of interest: $\Pr\left(y \mid x, S\right) = \underset{\gamma\sim\Pr(\cdot|S)}{\mathbf{E}} \underset{h\sim\Pr(\cdot\mid S,\gamma)}{\mathbf{E}} \Pr\left(y \mid h, x\right) = \underset{\gamma\sim\Pr(\cdot|S)}{\mathbf{E}} \Pr\left(y \mid x, S, \gamma\right)$, where $\Pr\left(\gamma|S\right) \propto \Pr\left(S|\gamma\right)\Pr\left(\gamma\right)$. Note here that $\Pr\left(S|\gamma\right) = \underset{h\sim\Pr(\cdot|\gamma)}{\mathbf{E}} P(S|h)$ and is known as the marginal likelihood. Thus, we only need to specify a prior $\Pr(\gamma)$ over the different learning algorithms we want to use and average over the different models produced. While this approach is appealing and has been useful in several practical applications, it has a few major drawbacks. For instance, many important learning algorithms do not have a Bayesian flavor and do not provide a means for computing the marginal likelihood. When it does, it often carries an unappealing computational burden. Also, as argued in Wahba [1990] Sec. 4.8, this approach is often a victim of model mis-specification. Meaning that when $P(\gamma)$ or $P(h|\gamma)$ doesn't reflect the task's structure, the final estimator is likely to be suboptimal. For these reasons, it is common to use some form of cross-validation to be able to estimate the expected loss for each individual model and select the most promising one. However, most cross-validation approaches yield a biased and noisy estimation of the expected loss on which we rely to select a single model. To overcome these limitations, in this work, we combine the advantages of both approaches by using holdout methods for estimating a model-free posterior distribution $\Pr(\gamma|S)$ over the model space Γ.

A.3 Uncertainty in the Holdout Estimation of the Expected Loss

A learning algorithm, parametrized by hyperparameters γ, can be seen as a function $\mathcal{A}_\gamma$ that, given a training set S, returns a hypothesis $h : \mathcal{X} \to \mathcal{Y}$. The goal of model selection is to be able to find the model γ that will produce the classifier with the smallest expected loss, $\mathcal{L}(\gamma)$, on task D using m training samples, where $\mathcal{L}(\gamma) \overset{\text{def}}{=} \underset{S\sim D^m}{\mathbf{E}} R_D(\mathcal{A}_\gamma(S))$. The k-fold cross-validation will estimate this average expected loss using $\widehat{\mathcal{L}}_k(\gamma) \overset{\text{def}}{=} \frac{1}{k}\sum_{i=1}^k R_{V_i}\left(\mathcal{A}_\gamma\left(\widetilde{S}_i\right)\right)$, where $\{V_1, V_2, \dots, V_k\}$ is a partition of S, $|V_i| \approx \frac{m}{k}$ and $\widetilde{S}_i \overset{\text{def}}{=} S \setminus V_i$. If A_γ is a stable learning algorithm Devroye and Wagner [1979] and $\frac{k-1}{k} \approx 1$, $\widehat{\mathcal{L}}_k(\gamma)$ is a acceptable sapproximation of $\mathcal{L}(\gamma)$. However, repeating the experiment while applying some small perturbations on S yields an important variation on the choice of $\widehat{\gamma} \overset{\text{def}}{=} \underset{\gamma\in\Gamma}{\operatorname{argmin}} \widehat{\mathcal{L}}_k(\gamma)$ and an even more important variation on the expected loss of the final classifier on the test set $R_T\left(A_{\widehat{\gamma}}(S)\right)$, where $T \sim D^n$.

This indicates that only selecting one $\widehat{\gamma}$ is suboptimal and that model averaging is likely to improve generalization performances.

Let (x_j, y_j) be the j^{th} sample of S and let $h_{\gamma,\backslash i}$ be the classifier that wasn't trained using the block V_i containing the j^{th} sample. Then, we can define the collection of losses by $l_{\gamma,j} \stackrel{\text{def}}{=} l(h_{\gamma,\backslash i}(x_j), y_j)$. With the Bayesian point of view, the collection of expected losses $\boldsymbol{\mathcal{L}} \stackrel{\text{def}}{=} (\mathcal{L}(\gamma_1), \mathcal{L}(\gamma_2), \dots)$ is a multivariate random variable and the observations are the $l_{\gamma,j}$. Thus, with a prior over $l_{\gamma,j}$, we can obtain a posterior distribution Q for $\boldsymbol{\mathcal{L}}$ over $\mathbb{R}^{|\Gamma|}$. With this in mind, we can now define the posterior distribution $\Pr(\gamma|S)$ as being the probability that γ is the best model among all candidates according to the posterior Q.

For example, let's suppose that we only have two different models, $\Gamma = \{\gamma_1, \gamma_2\}$ and let's suppose that Q is simply the product of two independent normal distributions where μ_1 is smaller than μ_2. This means that the classical cross-validation approach would select γ_1 since the average expected loss is smaller. However, when sampling $(\mathcal{L}(\gamma_1), \mathcal{L}(\gamma_2)) \sim Q$, a certain fraction of the samples will correspond to the event $\mathcal{L}(\gamma_1) < \mathcal{L}(\gamma_2)$ while the rest will correspond to the converse. Thus, the probability of these two events defines $\Pr(\gamma_1|S)$ and $\Pr(\gamma_2|S)$ respectively. When more training data becomes available, the variance of both distribution decreases. This allows to narrow down which one is actually the true best. However, with a fixed amount of samples, we have to deal with the uncertainty and perform model averaging.

A.3.1 Estimating Pr($\gamma|S$) with Bootstrap

Algorithm 3 ESTIMATE_POSTERIOR(*observations* $\{\boldsymbol{l}_j\}_{j=1}^m$, *number_of_samples* N)

```
Initialize $p_\gamma \leftarrow 0$    ;$\forall\gamma \in \Gamma$
for  $i = 0$ to $N$  do
    Sample with replacement a new set $\left\{\widetilde{\boldsymbol{l}}_j\right\}_{j=1}^m$ from the original set $\{\boldsymbol{l}_j\}_{j=1}^m$
    $\widehat{\mathcal{L}}(\gamma) \leftarrow \sum_{j=1}^m \frac{1}{m}\widetilde{l}_{\gamma,j}$    ;$\forall\gamma \in \Gamma$
    $\gamma^\star \leftarrow \operatorname{argmin}_{\gamma\in\Gamma} \widehat{\mathcal{L}}(\gamma)$ {If there are more than one $\gamma^\star$, select one at random}
    $p_{\gamma^\star} \leftarrow p_{\gamma^\star} + \frac{1}{N}$
end for
```

Up to now, we've been supposing that the posterior distribution Q was given and that we could sample from it. Unfortunately, there are several dependencies induced when training and testing different models with the same data set. This prevents us from factorizing Q into independent simple models, yielding a complex posterior to deal with. To simplify this problem, we will use the bootstrap method Efron [1979] designed for evaluating the uncertainty of an estimator. It has been showed by Rubin Rubin [1981] that the Bayesian bootstrap, a smoothed version of Efron's bootstrap, is equivalent to performing Bayesian inference with a Dirichlet prior of parameter $(0, 0, \dots, 0)$.

Table A.1: Toy example showing the $l_{\gamma,j}$ values for $|\Gamma| = 5$ and $m = 3$ with the zero-one loss. The last column gives $P(\gamma|S)$ for each model.

	$\boldsymbol{l}_1$	$\boldsymbol{l}_2$	$\boldsymbol{l}_3$	$P(\gamma\|S)$
γ_1	1	0	0	0.136
γ_2	1	0	0	0.136
γ_3	1	0	0	0.136
γ_4	0	1	0	0.295
γ_5	0	0	1	0.295

For conveniences, in this work, we will use the original bootstrap scheme. However, we still have to deal with the dependencies induced during cross validation. To do so, we consider $\boldsymbol{l}_j \overset{\text{def}}{=} \left(l_{\gamma_1,j}, l_{\gamma_2,j}, \ldots, l_{\gamma_{|\Gamma|},j}\right)$ as being one sample. Thus, the collection of $\{\boldsymbol{l}_j\}_{j=1}^m$ represents the set of samples that we have for estimating Q. Unfortunately, there are still dependencies among these samples and taking them into account presents a great challenge Bengio and Grandvalet [2004]. To be able to move forward, we will suppose that this only affects the effective sample size, i.e., we will suppose that we have $m' < m$ i.i.d. samples. Finally, to estimate $\Pr(\gamma|S)$, we sample with replacement from $\{\boldsymbol{l}_j\}_{j=1}^m$ to produce $\left\{\widetilde{\boldsymbol{l}}_j\right\}_{j=1}^{m'}$ and estimate the proportion of the time that $\widehat{\mathcal{L}}(\gamma)$ has the smallest value, where $\widehat{\mathcal{L}}(\gamma) \overset{\text{def}}{=} \sum_{j=1}^{m'} \frac{1}{m'} \widetilde{l}_{\gamma,j}$. This procedure is expressed more formally in Algorithm 3.

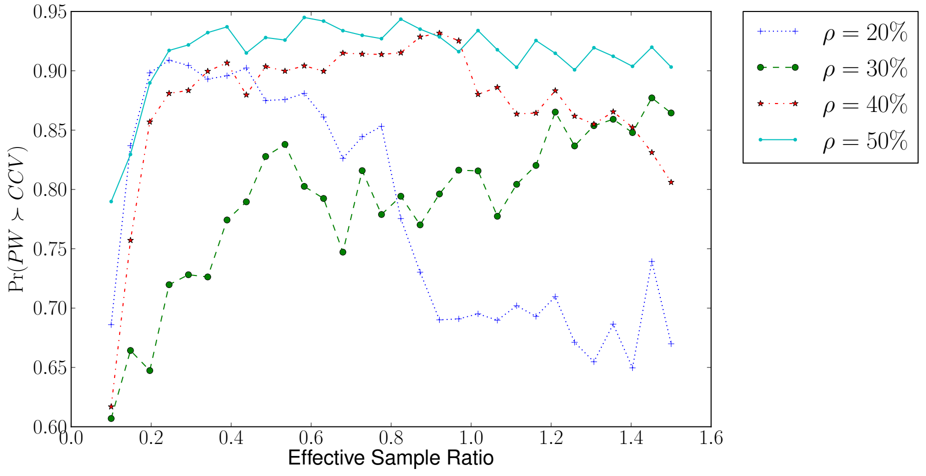


Figure A.1: Results showing the probability that the Predictor Weighting (PW) approach is better than the Classical Cross Validation (CCV), using the Poisson binomial test. $\rho \overset{\text{def}}{=} |S|/(|S| + |T|)$ determines the amount of samples taken from the original data set to produce the training set S. The rest is used for building the test set T.

Appendix B

Sequential Model-Based Ensemble Optimization, Early Work

B.1 Abstract

In this paper, we propose an extension of SMBO methods that automatically constructs ensembles trained models. This method builds on a recently proposed ensemble construction paradigm known as agnostic Bayesian learning. In experiments on 22 regression and 39 classification data sets, we confirm the success of this proposed approach, which is able to outperform model selection with SMBO.

B.2 Introduction

The automation of hyperparameter selection is an important step towards making the practice of machine learning more approachable to the non-expert and increasing its impact on data reliant sciences. Significant progress has been made recently, with many methods reporting success in tuning a large variety of algorithms [Bergstra et al., 2011, Hutter et al., 2011, Snoek et al., 2012, Thornton et al., 2013]. One successful general paradigm is known as Sequential Model-Based Optimization (SMBO). It is based on a process that alternates between the proposal of a new hyperparameter configuration to test and the update of an adaptive model of the relationship between hyperparameter configurations and their holdout set performances. Thus, as the model learns about this relationship, it increases its ability to suggest improved hyperparameter configurations and gradually converges to the best solution.

While finding the single best model configuration is useful, better performance is often obtained by combining several (good) models into an ensemble. However, constructing such ensembles is just as tedious as performing model selection and at least as important in the successful deployment of machine-learning-based systems. Moreover, unlike the model selection case for which SMBO can be used, little effort has been put on automating ensemble construction thus far. One exception is the work of Thornton et al. [2013], which can support the construction of ensembles, but only of up to 5 models.

In this paper, we propose a method for leveraging the recent research on SMBO in order to generate a (possibly large) ensemble of models, as opposed to the single best model. The proposed approach builds on the agnostic Bayes framework [Lacoste et al., 2014b], which provides a successful strategy for weighting a predetermined and finite set of models (already trained) into an ensemble. Using a successful SMBO method, we show how we can effectively generalize this framework to the case of an infinite space of models (indexed by its hyperparameter space). The resulting method is simple and highly efficient. Our experiments on 22 regression and 39 classification data sets confirm that it outperforms the regular SMBO model selection method.

B.3 Hyperparameter selection with SMBO

Let us first lay down the notation we will be using to describe the task of model selection for a machine learning algorithm. In this setup, a task D corresponds to a probability distribution over the input-output space $\mathcal{X} \times \mathcal{Y}$. Given a set of examples $S \sim D^m$ (which will be our holdout validation set), the objective is to find, among a set $\mathcal{H}$, the *best* function $h^\star : \mathcal{X} \to \mathcal{Y}$. In general, $\mathcal{H}$ can be any set and we refer to a member as a predictor. In the context of hyperparameter selection, $\mathcal{H}$ corresponds to the set of models trained on a training set $T \sim D^n$ (disjoint from S), for different configurations of the learning algorithm's hyperparameters γ. Namely, let $\mathcal{A}_\gamma$ be the learning algorithm with a hyperparameter configuration $\gamma \in \Gamma$, we will note $h_\gamma = \mathcal{A}_\gamma(T)$ the predictor obtained after training on T . The set $\mathcal{H}$ contains all predictors obtained from each $\gamma \in \Gamma$ when $\mathcal{A}_\gamma$ is trained on T, i.e. $\mathcal{H} \overset{\text{def}}{=} \{h_\gamma \mid \gamma \in \Gamma\}$.

To assess the quality of a predictor, we use a loss function $\mathcal{L} : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}$ that quantifies the penalty incurred when h_γ predicts $h_\gamma(x)$ while the true target is y. Then, we can define the risk $R_D(h_\gamma)$ as being the expected loss of h_γ on task D, i.e. $R_D(h_\gamma) \overset{\text{def}}{=} \mathop{\mathbf{E}}_{x,y\sim D} \left[\mathcal{L}\left(h_\gamma(x), y\right)\right]$. Finally, the *best*[1] function is simply the one minimizing the risk, i.e. $h^\star \overset{\text{def}}{=} \operatorname*{argmin}_{h_\gamma \in \mathcal{H}} R_D(h_\gamma)$. Here, estimating $h^\star$ thus corresponds to hyperparameter selection.

There has been an increasing amount of work on automatic hyperparameter optimization [Bergstra et al., 2011, Hutter et al., 2011, Snoek et al., 2012, Thornton et al., 2013]. Most rely on an approach called sequential model-based optimization (SMBO). The idea consists in treating $R_S(h_\gamma) \overset{\text{def}}{=} f(\gamma)$ as a learnable function of γ, which we can learn from the observations $\{(\gamma_i, R_S(h_{\gamma_i}))\}$ collected during the hyperparameter selection process. We must thus choose a model family for f, and a common choice is a Gaussian process (GP) representation, which

[1]The best solution may not be unique but any of them are equally good.

allows us to represent our uncertainty about f, i.e. our uncertainty about the value of $f(\gamma^*)$ at any unobserved hyperparameter configuration γ^*. This uncertainty can then be leveraged to determine an *acquisition function* that suggests the most promising hyperparameter configuration to test next. A popular acquisition function is the *expected improvement*, which can be maximized by gradient ascent initialized at points distributed across the hyperparameter space according to a Sobol sequence (to maximize the chance of finding a global optima). One advantage of expected improvement is that it directly offers a solution to the exploration-exploitation trade-off that hyperparameter selection faces.

An iteration of SMBO requires fitting the GP to the current set of tested hyperparameters $\mathcal{R}$ (initially empty), invoking the acquisition function, running the learning algorithm with the suggested hyperparameters and adding the result to $\mathcal{R}$. Fitting the GP corresponds to learning the mean and covariance functions hyperparameters to the collected data. This can be performed either by maximizing the data's marginal likelihood or defining priors over the hyperparameters and sampling from the posterior (see Snoek et al. [2012] for more details).

While SMBO hyperparameter optimization can produce very good predictors, it can also suffer from overfitting on the validation set, especially for high-dimensional hyperparameter spaces. This is in part why an ensemble of predictors are often preferable in practice. Properly extending SMBO to the construction of ensembles is, however, not obvious. Here, we propose one such successful extension, building on the framework of Agnostic Bayes learning, described in the next section.

B.4 Agnostic Bayes

In this section, we offer a brief overview of the Agnostic Bayes learning paradigm presented in Lacoste et al. [2014b], where it was used to construct ensembles for finite sets $\mathcal{H}$. In this paper, we will thus later generalize this approach to the infinite case.

Agnostic Bayes learning attempts to directly address the problem of inferring what is the *best* function $h^\star$ in $\mathcal{H}$, according to the loss function $\mathcal{L}$. It infers a posterior $p_{h^\star}(h_\gamma|S)$, i.e. a distribution over how likely each member of $\mathcal{H}$ is the best predictor. This is in contrast with standard Bayesian learning, which implicitly assumes that $\mathcal{H}$ contains the true data-generating model and infers a distribution for how likely each member of $\mathcal{H}$ has generated the data (irrespective of what the loss $\mathcal{L}$ is). From $p_{h^\star}(h_\gamma|S)$, by marginalizing, we obtain a probabilistic estimate for the best prediction $y^\star \overset{\text{def}}{=} h^\star(x)$

$$p_{y^\star}(y|x,S) = \sum_{\gamma\in\Gamma} p_{h^\star}(h_\gamma|S)\,\mathrm{I}[h_\gamma(x) = y].$$

Finally, to commit to a final prediction, for a given x, we use the most probable answer[2].

[2]As noted in Lacoste et al. [2014b], $p_{y^\star}(y|x,S)$ does not corresponds to the probability of observing y given x and cannot be used with the optimal Bayes theory, thus justifying the usage of the most probable answer

This yields the following ensemble decision rule

$$E^\star(x) \overset{\text{def}}{=} \underset{y\in\mathcal{Y}}{\text{argmax}}\; p_{y^\star}(y|x,S). \tag{B.1}$$

To estimate $p_{h^\star}(h_\gamma|S)$, Agnostic Bayes learning uses the set of losses $l_{\gamma,i} \overset{\text{def}}{=} \mathcal{L}(h_\gamma(x_i), y_i)$ of each example $(x_i, y_i) \in S$ as evidence for inference. In Lacoste et al. [2014b], a few different approaches are proposed and analyzed. A general strategy is to assume a joint prior $p(\mathbf{r})$ over the risks $r_\gamma \overset{\text{def}}{=} R_D(h_\gamma)$ of all possible hyperparameter configurations and choose a joint observation $p(l_{\gamma,i}\ \forall\gamma \in \Gamma|\mathbf{r})$ for the losses. From Bayes rule, we obtain the posterior $p(\mathbf{r}|S)$ from which we can compute

$$p_{h^\star}(h_\gamma|S) \quad = \quad \mathrm{E}_{\mathbf{r}}\left[\mathrm{I}[r_\gamma < r_{\gamma'}, \forall\gamma' \neq \gamma]|S\right] \tag{B.2}$$

with a Monte Carlo estimate. This would result in repeatedly sampling from $p(\mathbf{r}|S)$ and counting the number of times each γ has the smallest sampled risk r_γ to estimate $p_{h^\star}(h_\gamma|S)$. Similarly, samples from $p_{h^\star}(h_\gamma|S)$ could be obtained by sampling a risk vector $\mathbf{r}$ from $p(\mathbf{r}|S)$ and returning the predictor h_γ with the lowest sampled risk. The ensemble decision rule of Equation B.1 could then be implemented by repeatedly sampling from $p_{h^\star}(h_\gamma|S)$ to construct the ensemble of predictors and using their average as the ensemble's prediction.

Among the methods explored in Lacoste et al. [2014b] to obtain samples from $p(\mathbf{r}|S)$, the bootstrap approach stands out for its efficiency and simplicity. Namely, to obtain a sample from $p(\mathbf{r}|S)$, we sample with replacement from S to obtain S' and return the vector of empirical risks $[R_{S'}(h_{\gamma_1}), \ldots, R_{S'}(h_{\gamma_{|\Gamma|}})]^\top$ as a sample. While bootstrap only serves as a "poor man's" posterior, it can be shown to be statistically related to a proper model with Dirichlet priors and its empirical performance was shown to be equivalent [Lacoste et al., 2014b].

When the bootstrap method is used to obtain samples from $p_{h^\star}(h_\gamma|S)$, the complete procedure for generating each ensemble member can be summarized by

$$\widetilde{h^\star} = \underset{\gamma\in\Gamma}{\text{argmin}}\; R_{S'}(h_\gamma), \tag{B.3}$$

where $\widetilde{h^\star}$ corresponds to a sample from $p_{h^\star}(h_\gamma|S)$.

B.5 Agnostic Bayes ensemble with SMBO

We now present our proposed method for automatically constructing an ensemble, without having to restrict Γ (or, equivalently $\mathcal{H}$) to a finite subset of hyperparameters.

As described in Section B.4, to sample a predictor from the Agnostic Bayes bootstrap method, it suffices to obtain a bootstrap S' from S and solve the optimization problem of Equation B.3. In our context where $\mathcal{H}$ is possibly an infinite set of models trained on the training set T for any hyperparameter configuration γ, Equation B.3 corresponds in fact to hyperparameter optimization where the holdout set is S' instead of S. We could thus perform this step using SMBO.

This suggests a simple procedure for building an ensemble of N predictors according to agnostic Bayes i.e., that reflects our uncertainty about the true best model $h^{\star}$. We could repeat the full SMBO hyperparameter optimization process N times, with different bootstrap S'_j, for $j \in \{1, 2, \ldots, N\}$, each process carrying its own history $\mathcal{R}_j$ of tested hyperparameters. However, for large ensembles, performing N runs of SMBO can be computationally expensive, since each run would need to train its own sequence of models.

We can notice however that predictors are always trained on the same training set T, no matter in which run of SMBO they were trained on. We propose a handy trick that exploits this observation to greatly accelerate the construction of the ensemble by almost a factor of N. Specifically, we propose to simultaneously optimize all N problems in a round-robin fashion. Thus, while we maintain the N different histories of evaluation $\mathcal{R}_j$, when a new predictor $h_\gamma = \mathcal{A}_\gamma(T)$ is obtained, we update all $\mathcal{R}_j$ with $(\gamma, R_{S'_j}(h_\gamma))$. Notice that the different histories $\mathcal{R}_j$ contain the empirical risks on different bootstrap holdout sets, but they are all updated at the cost of training only a single predictor. Also, to avoid recalculating multiple times $\mathcal{L}(h_\gamma(x_i), y_i)$, these values can be cached and shared in the computation of each $\mathcal{R}_j$. This leaves the task of updating all $\mathcal{R}_j$ insignificant compared to the computational time usually required for training a predictor. For more details and a pseudocode on this procedure, see Lacoste et al. [2014a].

By updating all $\mathcal{R}_j$ at the same time, we *trick* each SMBO run by updating its history with points it did not suggest. This implies that the GP model behind each SMBO run will be able to condition on more observations then it would if the runs had been performed in isolation. This can only benefit the GPs and improve the quality of their suggestions. Let M be the total number of hyperparameter configurations we are willing to test, in our experiments, we fix $N = \lfloor \frac{M}{2} \rfloor$. This maximizes the number of samples used to estimate $p_{y^\star}(y|x, S)$ while ensuring at least one SMBO step with a reasonably large history for each bootstrap.

B.6 Experiments

We now compare the SMBO ensemble approach (ESMBO) to three alternative methods for building a predictor from a machine learning algorithm with hyperparameters:

- A single model, whose hyperparameters were selected by hyperparameter optimization with SMBO (SMBO).
- A single model, whose hyperparameters were selected by a randomized search (RS), which in practice is often superior to grid search [Bergstra and Bengio, 2012].

- An Agnostic Bayes ensemble constructed from a randomly selected set of hyperparameters (ERS).

Both ESMBO and SMBO used GP models of the holdout risk, with hyperparameters trained to maximize the marginal likelihood. A constant was used for the mean function, while the Matérn 5/2 kernel was used for the covariance function, with length scale parameters. The GP's parameters were obtained by maximizing the marginal likelihood and a different length scale was used for each dimension[3].

Each method is allowed to evaluate 150 hyperparameter configurations. To compare their performances, we perform statistical tests on 7 different hyperparameter spaces (i.e. different learning algorihtm) over two different collections of data sets.

The different methods presented in this paper are generic and are meant to work across different tasks. It is thus crucial that we evaluate them on several data sets using metrics that do not assume commensurability across tasks [Demšar, 2006b]. The metrics of choice are thus the expected rank and the pairwise winning frequency. When the winning frequency $\rho_{i,l} > 0.5$, we say that method $\mathcal{A}_i$ is better than method $\mathcal{A}_l$. However, to make sure that this is not the outcome of chance, we use statistical tests such as the sign test and the Poisson Binomial test (PB test) [Lacoste et al., 2012]. The PB test derives a posterior distribution over $\rho_{i,l}$ and integrates the probability mass above 0.5, denoted as $\Pr(A \succ B)$. When $\Pr(A \succ B) > 0.8$, we say that the result is significant and when $\Pr(A \succ B) > 0.9$, we say that it is highly significant. Similarly for the sign test, when the p-value is lower than 0.1, it is significant and when lower than 0.05, it is highly significant.

To build a substantial collection of data sets, we used the AYSU collection [Ulaş et al., 2009] coming from the UCI and the Delve repositories and we added the MNIST data set. The resulting benchmark contains 39 classification data sets. We have also collected 22 regression data sets from the Louis Torgo collection[4].

Looking at the overall results over 7 different hyperparameter spaces in Table B.1[5], we observe that ESMBO is never significantly outperformed by any other method and often outperforms the others. More precisely, it is either ranked first or tightly following ERS. Looking more closely, we see that the cases where ESMBO does not significantly outperform ERS concerns hyperparameter spaces of low complexity. For example, most hyperparameter configurations

[3]We used the implementation provided by spearmint: `https://github.com/JasperSnoek/spearmint`

[4]These data sets were obtained from the following source : `http://www.dcc.fc.up.pt/~ltorgo/Regression/DataSets.html`

[5]The result tables present the winning frequency for each pair of methods, where grayed out values represent redundant information. As a complement, we also add the expected rank of each method in the rightmost column and sort the table according to this metric. To report the conclusion of the Sign test and the PB test, we use colored dots, where orange means significant and green means highly significant. The first dot reports the result of the PB test and the second one, the Sign test. For more stable results, we average the values obtained during the last 15 iterations.

Table B.1: Pairwise win frequency for the 3 different regression hyperparameter spaces (left) and 4 different classification hyperparameter spaces (right).

Support Vector Regressor

	ESMBO	ERS	SMBO	RS	**E**[R]
ESMBO	0.50	0.66 ●	0.82 ●●	0.86 ●●	1.66
ERS	0.34	0.50	0.50	0.77 ●●	2.38
SMBO	0.18	0.50	0.50	0.64 ○	2.68
RS	0.14	0.23	0.36	0.50	3.27

Gradient Boosting Regressor

	ERS	ESMBO	RS	SMBO	**E**[R]
ERS	0.50	0.52	0.77 ●●	0.86 ●●	1.84
ESMBO	0.48	0.50	0.77 ●●	0.91 ●●	1.85
RS	0.23	0.23	0.50	0.42	3.12
SMBO	0.14	0.09	0.58	0.50	3.19

Random Forest

	ESMBO	ERS	SMBO	RS	**E**[R]
ESMBO	0.50	0.53	0.76 ●●	0.91 ●●	1.80
ERS	0.47	0.50	0.72 ●●	1.00 ●●	1.81
SMBO	0.24	0.28	0.50	0.66	2.82
RS	0.09	0.00	0.34	0.50	3.57

Gradient Boosting Classifier

	ESMBO	ERS	RS	SMBO	**E**[R]
ESMBO	0.50	0.51	0.59	0.65 ○●	2.25
ERS	0.49	0.50	0.59	0.64 ○○	2.28
RS	0.41	0.41	0.50	0.55	2.64
SMBO	0.35	0.36	0.45	0.50	2.83

Random Forest

	ERS	ESMBO	RS	SMBO	**E**[R]
ERS	0.50	0.52	0.60 ○	0.64 ○●	2.24
ESMBO	0.48	0.50	0.60	0.67 ○●	2.25
RS	0.40	0.40	0.50	0.57	2.63
SMBO	0.36	0.33	0.43	0.50	2.89

Multilayer Perceptron

	ESMBO	SMBO	ERS	RS	**E**[R]
ESMBO	0.50	0.57 ○	0.76 ●●	0.75 ●●	1.92
SMBO	0.43	0.50	0.68 ○●	0.68 ○●	2.21
ERS	0.24	0.32	0.50	0.54	2.91
RS	0.25	0.32	0.46	0.50	2.96

of Random Forest yield good generalization performances. Thus, these cases do not require an elaborate hyperparameter search method. On the other hand, when looking at more challenging hyperparameter spaces such as Support Vector Regression and Multilayer Perceptrons, we clearly see the benefits of combining SMBO with Agnostic Bayes.

B.7 Conclusion

We described a successful method for automatically constructing ensembles without requiring hand-selection of models or a grid search. Its generalization performance on 22 regression and 39 classification data sets was shown to be competitive. The method can adapt the SMBO hyperparameter optimization algorithm so that it can produce an ensemble instead of a single model. Theoretically, the method is motivated by an Agnostic Bayesian paradigm which attempts to construct ensembles that reflect the uncertainty over which a model actually has the smallest true risk. The resulting method is easy to implement and comes with no extra computational cost at learning time.

Bibliography

M. Abramowitz and I.A. Stegun. *Handbook of mathematical functions with formulas, graphs, and mathematical tables*. Dover publications, 1964.

Ethem Alpaydm. Combined 5× 2 cv f test for comparing supervised classification learning algorithms. *Neural computation*, 11(8):1885–1892, 1999.

Christophe Andrieu, Nando De Freitas, Arnaud Doucet, and Michael I Jordan. An introduction to MCMC for machine learning. *Machine learning*, 50(1-2):5–43, 2003.

Jean-Yves Audibert and Olivier Bousquet. Combining PAC-Bayesian and generic chaining bounds. *The Journal of Machine Learning Research*, 8:863–889, 2007.

K. Bache and M. Lichman. UCI machine learning repository, 2013. URL `http://archive.ics.uci.edu/ml`.

Mikhail Belkin, Partha Niyogi, and Vikas Sindhwani. Manifold regularization: A geometric framework for learning from labeled and unlabeled examples. *The Journal of Machine Learning Research*, 7:2399–2434, 2006.

Robert M Bell, Yehuda Koren, and Chris Volinsky. The bellkor solution to the netflix prize. *KorBell Team's Report to Netflix*, 2007.

A Benavoli, F Mangili, G Corani, M Zaffalon, and F Ruggeri. A Bayesian wilcoxon signed-rank test based on the dirichlet process. In *Proceedings of The 31st International Conference on Machine Learning*, 2014.

Y. Bengio and Y. Grandvalet. No unbiased estimator of the variance of k-fold cross-validation. *The Journal of Machine Learning Research*, 5:1089–1105, 2004.

Y. Bengio and Y. Grandvalet. Bias in estimating the variance of k-fold cross-validation. *Statistical modeling and analysis for complex data problems*, pages 75–95, 2005.

Yoshua Bengio. Learning deep architectures for ai. *Foundations and trends® in Machine Learning*, 2(1):1–127, 2009.

Y. Benjamini and Y. Hochberg. Controlling the false discovery rate: a practical and powerful approach to multiple testing. *Journal of the Royal Statistical Society. Series B (Methodological)*, pages 289–300, 1995.

James Bergstra and Yoshua Bengio. Random search for hyper-parameter optimization. *Journal of Machine Learning Research*, 13:281–305, 2012.

James Bergstra, Rémi Bardenet, Yoshua Bengio, and Balázs Kégl. Algorithms for hyper-parameter optimization. In *NIPS*, pages 2546–2554, 2011.

James Bergstra, Daniel Yamins, and David D. Cox. Making a science of model search: Hyperparameter optimization in hundreds of dimensions for vision architectures. In *ICML (1)*, pages 115–123, 2013.

L. Breiman. Bagging predictors. *Machine learning*, 24(2):123–140, 1996.

L. Breiman. Random forests. *Machine learning*, 45(1):5–32, 2001.

Olivier Catoni. PAC-Bayesian supervised classification: The thermodynamics of statistical learning. *arXiv preprint arXiv:0712.0248*, 2007.

Minmin Chen, Zhixiang Eddie Xu, Kilian Q. Weinberger, and Fei Sha. Marginalized denoising autoencoders for domain adaptation. In *ICML*. icml.cc / Omnipress, 2012.

S.X. Chen and J.S. Liu. Statistical Applications of the Poisson-Binomial and conditional Bernoulli distributions. *Statistica Sinica*, 7:875–892, 1997.

X.H. Chen, A.P. Dempster, and J.S. Liu. Weighted finite population sampling to maximize entropy. *Biometrika*, 81(3):457, 1994.

R. Collobert, J. Weston, L. Bottou, M. Karlen, K. Kavukcuoglu, and P. Kuksa. Natural language processing (almost) from scratch. *Journal of Machine Learning Research*, 12: 2493–2537, 2011.

C. Cortes and V. Vapnik. Support-vector networks. *Machine learning*, 20(3):273–297, 1995.

M.H. DeGroot. *Optimal statistical decisions*, volume 82. Wiley-interscience, 2005.

J. Demšar. Statistical comparisons of classifiers over multiple data sets. *The Journal of Machine Learning Research*, 7:1–30, 2006a.

Janez Demšar. Statistical comparisons of classifiers over multiple data sets. *The Journal of Machine Learning Research*, 7:1–30, 2006b.

L. Devroye and T. Wagner. Distribution-free performance bounds for potential function rules. *IEEE Transactions on Information Theory*, 25(5):601–604, 1979.

T.G. Dietterich. Approximate statistical tests for comparing supervised classification learning algorithms. *Neural computation*, 10(7):1895–1923, 1998.

Harris Drucker, Chris JC Burges, Linda Kaufman, Alex Smola, and Vladimir Vapnik. Support vector regression machines. *Advances in neural information processing systems*, pages 155–161, 1997.

B. Efron. Bootstrap methods: another look at the jackknife. *The annals of Statistics*, 7(1): 1–26, 1979.

Clément Farabet, Camille Couprie, Laurent Najman, and Yann LeCun. Learning hierarchical features for scene labeling. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 35(8):1915–1929, 2013.

M. Fernandez and S. Williams. Closed-Form Expression for the Poisson-Binomial Probability Density Function. *Aerospace and Electronic Systems, IEEE Transactions on*, 46(2):803–817, 2010.

Charles W Fox and Stephen J Roberts. A tutorial on variational bayesian inference. *Artificial Intelligence Review*, 38(2):85–95, 2012.

Y. Freund and R. Schapire. A desicion-theoretic generalization of on-line learning and an application to boosting. In *Computational learning theory*, pages 23–37. Springer, 1995.

Jerome H Friedman. Greedy function approximation: a gradient boosting machine. *Annals of Statistics*, pages 1189–1232, 2001.

M. Friedman. The use of ranks to avoid the assumption of normality implicit in the analysis of variance. *Journal of the American Statistical Association*, 32(200):675–701, 1937. ISSN 0162-1459.

Pascal Germain, Alexandre Lacasse, François Laviolette, and Mario Marchand. A PAC-Bayes risk bound for general loss functions. In Bernhard Schölkopf, John C. Platt, and Thomas Hoffman, editors, *NIPS*, pages 449–456. MIT Press, 2006. ISBN 0-262-19568-2.

Pascal Germain, Alexandre Lacasse, François Laviolette, and Mario Marchand. PAC-Bayesian learning of linear classifiers. In *Proceedings of the 26th Annual International Conference on Machine Learning*, pages 353–360. ACM, 2009.

Pascal Germain, Alexandre Lacoste, François Laviolette, Mario Marchand, and Sara Shanian. A PAC-Bayes sample-compression approach to kernel methods. In *ICML*, pages 297–304, 2011.

Pierre Geurts, Damien Ernst, and Louis Wehenkel. Extremely randomized trees. *Machine learning*, 63(1):3–42, 2006.

Zoubin Ghahramani. Bayesian non-parametrics and the probabilistic approach to modelling. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 371(1984):20110553, 2013.

Sébastien Giguère, Alexandre Drouin, Alexandre Lacoste, Mario Marchand, Jacques Corbeil, and François Laviolette. MHC-NP: Predicting peptides naturally processed by the MHC. *Journal of immunological methods*, 400:30–36, 2013.

Peter Grünwald and John Langford. Suboptimal behavior of bayes and MDL in classification under misspecification. *Machine Learning*, 66(2-3):119–149, 2007.

Isabelle Guyon, Amir Saffari, Gideon Dror, and Gavin Cawley. Model selection: Beyond the bayesian/frequentist divide. *The Journal of Machine Learning Research*, 11:61–87, 2010.

Y. Hochberg and A.C. Tamhane. *Multiple comparison procedures*, volume 82. Wiley Online Library, 1987.

Jennifer A Hoeting, David Madigan, Adrian E Raftery, and Chris T Volinsky. Bayesian model averaging: a tutorial. *Statistical science*, pages 382–401, 1999.

Frank Hutter, Holger H Hoos, and Kevin Leyton-Brown. Sequential model-based optimization for general algorithm configuration. In *Learning and Intelligent Optimization*, pages 507–523. Springer, 2011.

Il'dar Abdulovich Ibragimov and Rafail Z Has'minskii. *Statistical estimation.* Springer, 1981.

E.T. Jaynes. Information theory and statistical mechanics. II. *Physical review*, 108(2):171, 1957.

Thorsten Joachims. Transductive inference for text classification using support vector machines. In *ICML*, volume 99, pages 200–209, 1999.

M.J. Kearns, R.E. Schapire, and L.M. Sellie. Toward efficient agnostic learning. *Machine Learning*, 17(2):115–141, 1994.

Hyun-Chul Kim and Zoubin Ghahramani. Bayesian classifier combination. *Journal of Machine Learning Research - Proceedings Track*, 22:619–627, 2012.

Alexandre Lacasse, François Laviolette, Mario Marchand, Pascal Germain, and Nicolas Usunier. PAC-Bayes bounds for the risk of the majority vote and the variance of the Gibbs classifier. In *NIPS*, pages 769–776, 2006.

Alexandre Lacoste, François Laviolette, and Mario Marchand. Bayesian comparison of machine learning algorithms on single and multiple datasets. *Journal of Machine Learning Research - Proceedings Track*, 22:665–675, 2012.

Alexandre Lacoste, Hugo Larochelle, Mario Marchand, and François Laviolette. Sequential model-based ensemble optimization. In *Proceedings of the 30th Conference on Uncertainty in Artificial Intelligence (UAI)*, pages 440–448, 2014a.

Alexandre Lacoste, Mario Marchand, François Laviolette, and Hugo Larochelle. Agnostic bayesian learning of ensembles. In *Proceedings of The 31st International Conference on Machine Learning*, pages 611–619, 2014b.

J. Langford. Tutorial on practical prediction theory for classification. *Journal of Machine Learning Research*, 6(1):273, 2006.

John Langford and John Shawe-Taylor. PAC-Bayes & margins. In *NIPS*, pages 423–430, 2002.

F. Laviolette, M. Marchand, and J.F. Roy. From PAC-Bayes bounds to quadratic programs for majority votes. *moment*, 1500:Q2, 2011.

W. Mendenhall. Nonparametric statistics. *Introduction to Probability and Statistics*, 604, 1983.

M. Minsky and S. Papert. Perceptrons. *MIT Press,Cambridge, MA*, 1969.

Andrew Y. Ng and Michael I. Jordan. On discriminative vs. generative classifiers: A comparison of logistic regression and naive bayes. In Thomas G. Dietterich, Suzanna Becker, and Zoubin Ghahramani, editors, *NIPS*, pages 841–848. MIT Press, 2001.

E. Parzen. On estimation of a probability density function and mode. *The annals of mathematical statistics*, 33(3):1065–1076, 1962. ISSN 0003-4851.

Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, et al. Scikit-learn: Machine learning in python. *The Journal of Machine Learning Research*, 12: 2825–2830, 2011.

Carl Rasmussen and Chris Williams. Gaussian processes for machine learning. *Gaussian Processes for Machine Learning*, 2006.

Christian P Robert. The Bayesian choice: From decision-theoretic foundations to computational implementation (springer texts in statistics) by. 2001.

M. Rosenblatt. Remarks on some nonparametric estimates of a density function. *The Annals of Mathematical Statistics*, 27(3):832–837, 1956. ISSN 0003-4851.

Jean-Francis Roy, François Laviolette, and Mario Marchand. From PAC-Bayes bounds to quadratic programs for majority votes. In *ICML*, pages 649–656, 2011.

D.B. Rubin. The bayesian bootstrap. *The annals of statistics*, 9(1):130–134, 1981.

Jayaram Sethuraman. A constructive definition of dirichlet priors. Technical report, DTIC Document, 1991.

Jasper Snoek, Hugo Larochelle, and Ryan P. Adams. Practical bayesian optimization of machine learning algorithms. In *NIPS*, pages 2960–2968, 2012.

Chris Thornton, Frank Hutter, Holger H. Hoos, and Kevin Leyton-Brown. Auto-WEKA: Combined selection and hyperparameter optimization of classification algorithms. In *Proc. of KDD-2013*, pages 847–855, 2013.

Simon Tong and Daphne Koller. Support vector machine active learning with applications to text classification. *The Journal of Machine Learning Research*, 2:45–66, 2002.

Aydın Ulaş, Murat Semerci, Olcay Taner Yıldız, and Ethem Alpaydın. Incremental construction of classifier and discriminant ensembles. *Information Sciences*, 179(9):1298–1318, April 2009.

Leslie G. Valiant. A theory of the learnable. *Communications of the ACM*, 27(11):1134–1142, 1984.

G. Wahba. *Spline models for observational data*, volume 59. Society for Industrial Mathematics, 1990.

YH Wang. On the number of successes in independent trials. *Statistica Sinica*, 3(2):295–312, 1993.

Kilian Weinberger, Anirban Dasgupta, John Langford, Alex Smola, and Josh Attenberg. Feature hashing for large scale multitask learning. In *Proceedings of the 26th Annual International Conference on Machine Learning*, pages 1113–1120. ACM, 2009.

F. Wilcoxon. Individual comparisons by ranking methods. *Biometrics Bulletin*, 1(6):80–83, 1945. ISSN 0099-4987.

Tong Zhang. Information-theoretic upper and lower bounds for statistical estimation. *Information Theory, IEEE Transactions on*, 52(4):1307–1321, 2006.

Yi Zhang, Samuel Burer, and W Nick Street. Ensemble pruning via semi-definite programming. *The Journal of Machine Learning Research*, 7:1315–1338, 2006.

Xiangxin Zhu and Deva Ramanan. Face detection, pose estimation, and landmark localization in the wild. In *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, pages 2879–2886. IEEE, 2012.