Sequential Model-Based Ensemble Optimization

Alexandre Lacoste*
Alexandre.Lacoste.1@ulaval.ca
Département IFT-GLO
Université Laval
Québec, Canada

Hugo Larochelle
Hugo.Larochelle@usherbrooke.ca
Département d'informatique
Université de Sherbrooke
Québec, Canada

Mario Marchand
Mario.Marchand@ift.ulaval.ca
Département IFT-GLO
Université Laval
Québec, Canada

François Laviolette
Francois.Laviolette@ift.ulaval.ca
Département IFT-GLO
Université Laval
Québec, Canada

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.

1 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., 2014], 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 result-

ing 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 2). We follow with a description of the Agnostic Bayes framework and present a bootstrap-based implementation of it (Section 3). Then, we describe the proposed algorithm for automatically constructing an ensemble using SMBO (Section 4). Finally, related work is discussed (Section 5) and the experimental comparisons are presented (Section 6).

2 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. $\mathcal{H} \stackrel{\text{def}}{=} \{h_\gamma | \gamma \in \Gamma\}$.

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. $R_D(h_\gamma) \stackrel{\text{def}}{=} \mathop{\mathbf{E}}_{x,y\sim D} \left[\mathcal{L}\left(h_\gamma(x), y\right)\right]$. Finally, the *best*[1] function is simply the one minimizing the risk, i.e.

$$h^\star \stackrel{\text{def}}{=} \mathop{\text{argmin}}_{h_\gamma \in \mathcal{H}} R_D(h_\gamma).$$

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 $R_S(h_\gamma) \stackrel{\text{def}}{=} f(\gamma)$ 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. 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

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

where $R_S\left(h_{\gamma_i}\right)$ 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

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

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*:

$$\text{EI}(\gamma^*; \mathcal{R}) \stackrel{\text{def}}{=} \text{E}\left[\max\{r_{\text{best}} - f(\gamma^*), 0\} | \mathcal{R}\right] \tag{2}$$

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

which can be shown to be equal to

$$\sigma^2(\gamma^*;\mathcal{R})\left(d(\gamma^*;\mathcal{R})\Phi(d(\gamma^*;\mathcal{R}))+\mathcal{N}(d(\gamma^*;\mathcal{R}),0,1)\right) \tag{3}$$

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

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

The acquisition function thus maximizes Equation 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 γ}
  $\mathcal{R} \leftarrow \mathcal{R} \cup \{(\gamma, R_S(h_\gamma))\}$ {Add to collected data}
end for
$\gamma^* \leftarrow \underset{(\gamma, R_S(h_\gamma))\in\mathcal{R}}{\text{argmin}} R_S(h_\gamma)$
return h_{γ^*}

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.

3 AGNOSTIC BAYES

In this section, we offer a brief overview of the Agnostic Bayes learning paradigm presented in Lacoste et al. [2014] and serving as a basis for the algorithm we present in this paper. Agnostic Bayes learning was used in Lacoste et al. [2014] 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 $y^\star \stackrel{\text{def}}{=} h^\star(x)$

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

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

$$E^\star(x) \stackrel{\text{def}}{=} \underset{y\in\mathcal{Y}}{\text{argmax}}\, p_{y^\star}(y|x,S). \tag{4}$$

To estimate $p_{h^\star}(h_\gamma|S)$, Agnostic Bayes learning uses the set of losses $l_{\gamma,i} \stackrel{\text{def}}{=} \mathcal{L}(h_\gamma(x_i), y_i)$ of each example $(x_i, y_i) \in S$ as evidence for inference. In Lacoste et al. [2014], a few different approaches are proposed and analyzed. A general strategy is to assume a joint prior $p(\mathbf{r})$ over the risks $r_\gamma \stackrel{\text{def}}{=} R_D(h_\gamma)$ 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

$$p_{h^\star}(h_\gamma|S) \quad = \quad \mathrm{E}_{\mathbf{r}}\left[I[r_\gamma < r_{\gamma'}, \forall\gamma' \neq \gamma]|S\right] \tag{5}$$

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

[2] As noted in Lacoste et al. [2014], $p_{y^\star}(y|x,S)$ 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

ensemble decision rule of Equation 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. [2014] 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., 2014].

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

$$\widetilde{h^\star} = \operatorname*{argmin}_{\gamma\in\Gamma} R_{S'}(h_\gamma), \tag{6}$$

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 extended 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. [2014] 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. [2014].

4 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 3, 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 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 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**
  $v \leftarrow (k-1) \text{ modulo } N + 1$
  $\gamma \leftarrow \text{SMBO}(\mathcal{R}_v)$
  $h_\gamma \leftarrow \mathcal{A}_\gamma(T)$
  $\mathcal{E} \leftarrow \mathcal{E} \cup \{h_\gamma\}$
  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**
  $h_j \leftarrow \operatorname*{argmin}_{h_\gamma \in \mathcal{E}} R_{S'_j}(h_\gamma)$
  $\mathcal{H}' \leftarrow \mathcal{H}' \cup \{h_j\}$
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., 2014].

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 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.

5 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.

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

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 hold-out 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.

6.1 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.

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

[5]http://scikit-learn.org/

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 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.

6.2 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, 2006]. 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., $\mathbf{E}[\mathrm{R}]_i \stackrel{\text{def}}{=} \frac{1}{L}\sum_{j=1}^{L} \text{Rank}_{h_i, S_j^{\text{test}}}$. When comparing algorithm $\mathcal{A}_i$ against algorithm $\mathcal{A}_l$, the winning frequency[7] of $\mathcal{A}_i$ is

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

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

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, 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 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].

6.3 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 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.

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

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

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

Table 1: Significance Notation Used in Result Tables.

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

Table 2: Pairwise Win Frequency For the 3 Different Regression Hyperparameter Spaces (Refer to Section 6.3 for the notation).

Support Vector Regressor

	ESMBO	ERS	SMBO	RS	E[rank]
ESMBO	0.50	0.66$^{\bullet}$	0.82$^{\bullet\bullet}$	0.86$^{\bullet\bullet}$	1.66
ERS	0.34	0.50	0.50	0.77$^{\bullet\bullet}$	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$^{\bullet\bullet}$	0.86$^{\bullet\bullet}$	1.84
ESMBO	0.48	0.50	0.77$^{\bullet\bullet}$	0.91$^{\bullet\bullet}$	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$^{\bullet\bullet}$	0.91$^{\bullet\bullet}$	1.80
ERS	0.47	0.50	0.72$^{\bullet\bullet}$	1.00$^{\bullet\bullet}$	1.81
SMBO	0.24	0.28	0.50	0.66	2.82
RS	0.09	0.00	0.34	0.50	3.57

6.4 ANALYSIS

Looking at the overall results over 7 different hyperparameter spaces in Table 2 and Table 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.

As described in Section 4, 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 1 and Figure 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.

Table 3: Pairwise Win Frequency for the 4 Different Classification Hyperparameter Spaces (Refer to Section 6.3 for the notation).

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$^{\circ\bullet}$	2.25
ERS	0.49	0.50	0.59	0.64$^{\circ\circ}$	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 $^{\circ}$	0.64$^{\circ\bullet}$	2.24
ESMBO	0.48	0.50	0.60	0.67$^{\circ\bullet}$	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$^{\bullet\bullet}$	0.75$^{\bullet\bullet}$	1.92
SMBO	0.43	0.50	0.68$^{\circ\bullet}$	0.68$^{\circ\bullet}$	2.21
ERS	0.24	0.32	0.50	0.54	2.91
RS	0.25	0.32	0.46	0.50	2.96

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 $\Pr(\text{ESMBO} \succ \text{SMBO})$, 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.

7 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 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.

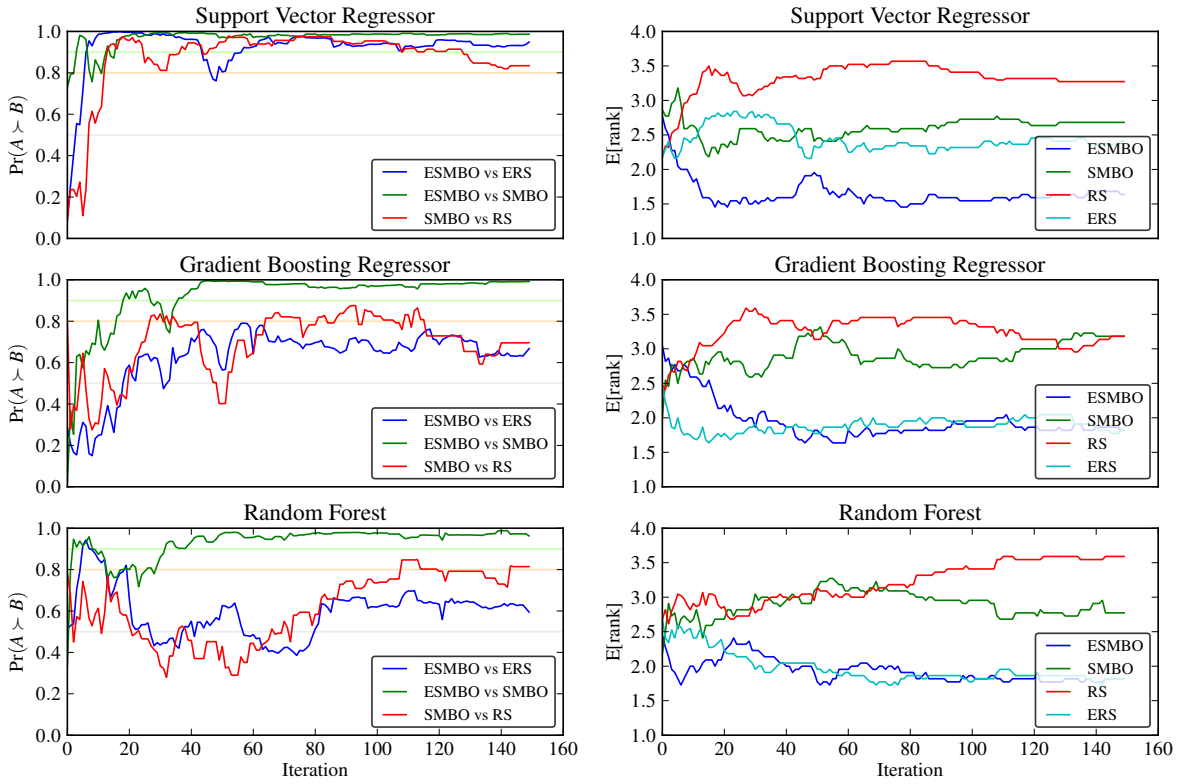

Figure 1: PB Probability and Expected Rank over Time for the 3 Regression Hyperparameter Spaces.

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.).

References

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

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.

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.

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

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, 2014.

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

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

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

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

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

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.

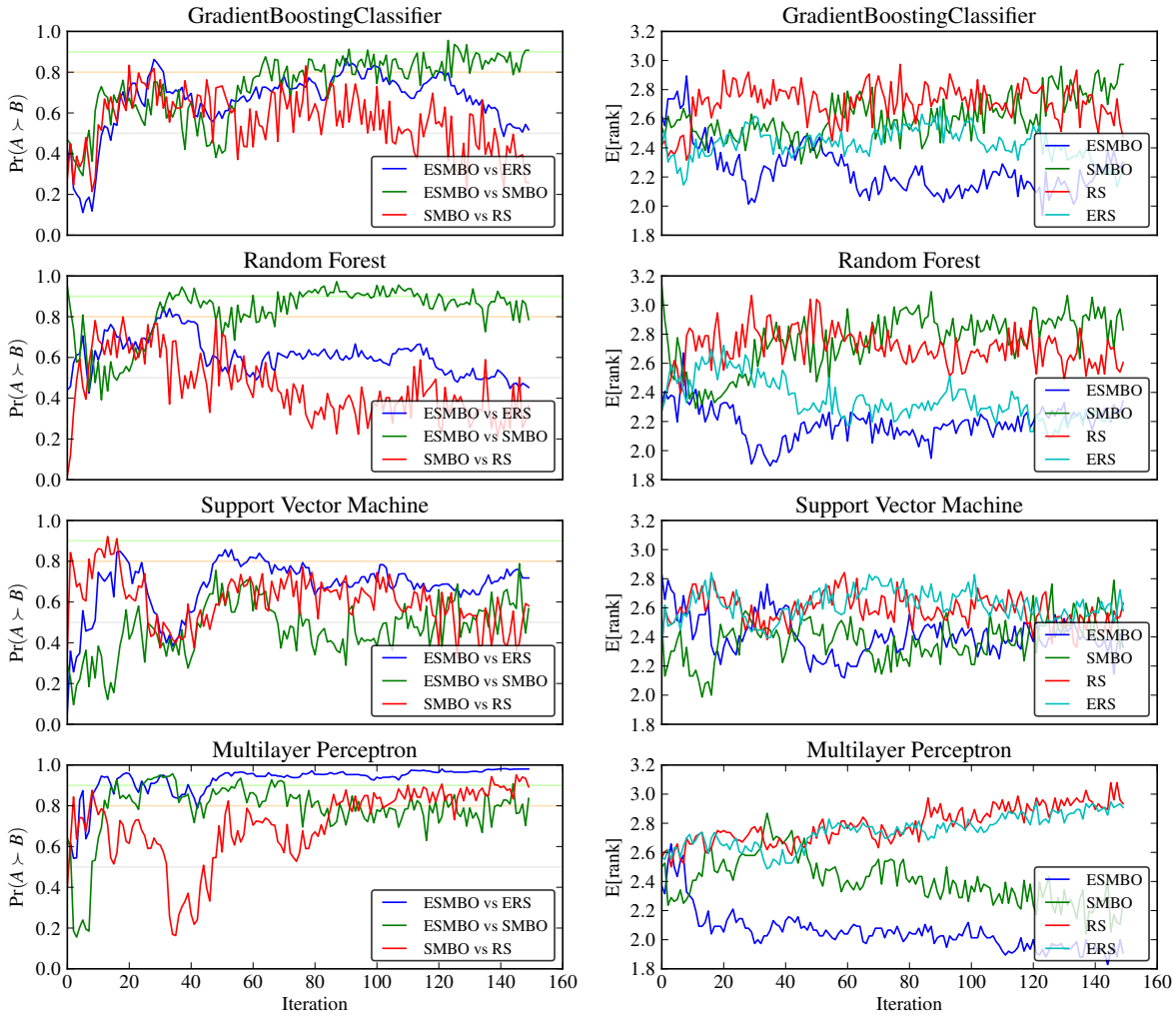

Figure 2: PB Probability and Expected Rank over Time for the 4 Classification Hyperparameter Spaces.

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

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

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

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

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.