Bayesian Optimization for hyperparameter Tuning in Random Forests

Anonymous Author(s)

Abstract

Ensemble classifiers are in widespread use now because of their promising empirical and theoretical properties. However, they tend to be computationally expensive because of the problem of hyperparameter tuning. In this study, we investigate the use of an aspiring method, Bayesian optimization, to solve this problem for one such ensemble classifier; a Random Forest. To test the effectiveness of our method, we apply it to three typical yet fundamentally different classification tasks: Optical Character Recognition, Twitter Sentiment Classification and Named Entity Recognition. Our results demonstrate that Bayesian optimization is able to find good estimates for hyperparameters in very few model evaluations, thus decreasing the cost of computation significantly.

1 Introduction

In this world of information explosion and tremendous computational capability, researchers have increasingly making progress in developing interesting generic techniques, such as deep learning [17], and Random Forests [10] to model and make prediction on huge datasets. The training of an ensemble of classifiers following certain properties and averaging over their prediction is an increasingly popular approach because of its promising empirical and theoretical results. A lot of success has been achieved in this area but at the same time it raises new questions, of which an important one is how to tune various hyperparameters. As parameters increase in numbers, so does their state space increase exponentially, and finding their optimal setting becomes intractable as we grow our collection of classifiers. This problem is commonly referred to as the curse of dimensionality.

In this study, we investigate the use of a Random Forest classifier on three heterogeneous domains. Random Forests (RFs) were first introduced by Breiman in [7]. They have proven especially useful in domains where our datasets consist of relatively few data point and a large number of features [27]. Formally, an RF is defined by Breiman, as a classifier consisting of a collection of decision trees \( g(x, Z_1), \ldots, g(x, Z_T) \), where the \( Z_1, \ldots, Z_T \) are independent random vectors, identically distributed, and each tree casts a unit vote for the most popular class at input \( x \).

The parameter vector \( Z \) mentioned above usually depends on the problem we are trying to solve, but as it happens they are further parametrized by hyperparameters that control their build. Among these parameters are values like the minimum inner-node size, the maximum number of features sampled at each node, and the maximum depth allowed for the decision tree. The most obvious hyperparameter is the number of decision trees that make up our forest. The accuracy of an RF is greatly affected by the values of these hyperparameters and thus, researchers have tried to carefully handtune their values with cross-validation or grid search if computationally feasible. However,
these methods are highly inefficient, mostly because they don’t use the information collected in the previous experiments. An automated method that makes educated guesses is therefore highly desirable.

To reduce the number of experiments performed to optimize the hyperparameters, researchers have investigated optimization methods that model the evaluation history, and then exploit this information to direct further search [3] [22]. These optimization techniques can become very difficult there isn’t an analytical expression of the function we wish to evaluate and by extension the effect of the hyperparameters on the accuracy of our classifier. In such cases all we have is a black box, which can take the parameter values and return an estimation of our accuracy. A technique that solves this problem very effectively is Bayesian optimization[20]. Bayesian Optimization (BO) works by assuming that the unknown function was sampled from a Gaussian process and updates a Gaussian process posterior distribution for the function as new observations are made [25]. We will use this technique to optimize the hyperparameters of an RF to achieve near-optimal accuracy in as few iterations as possible, thereby saving precious computing power and time.

In the next Section, we briefly review RFs and BO. In Sec. 3, we will describe our methodology, and in Sec. 4 the three classification domains are described and the corresponding experiments we perform to show the performance of RFs in combination with BO. Finally, we draw conclusions and present some future work.

2 Background and Related Work

In the last decade considerable success has been gained in applying RFs to classification tasks in diverse domains. These include medical imaging [19], the popular Microsoft Kinect [10], speech emotion recognition [13] and many more. In [2], theoretical foundations of the RF model are established and it is shown that its accuracy is dependent on the strength of the individual decision tree classifiers it encapsulates and a measure of dependence between them. This foundation is improved in [6], where a statistical analysis of RFs is performed and the fact demonstrated that to improve accuracy, the randomness that gets injected to the forest has to minimize the correlation between individual trees while still maintaining a measure of strength. To achieve this, different researchers have constructed variations of RFs by exploring different ways of introducing randomness, with methods ranging from extreme random splitting strategies like [6] and [11] to more data-dependent strategies such as [2] and [12].

In [4], the authors extend the work done by [7] and [2] to prove a groundbreaking theoretical result. Namely, that if the number of trees, \( T \), is large, an RF classifier is well approximated by simply averaging the outputs of all the different decision trees. In support of that conclusion they also prove a number of probabilistic properties that further cement the excellent performance demonstrated by RFs in many problem domains. These results provide us with the key motivation for confidently choosing RFs, in conjunction with the fact that training individual trees is highly parallelizable and thus computationally scalable.

A lot of work has recently been done to investigate the use of BO for optimizing functions that unfortunately don’t have a closed form expression, are expensive to evaluate, do not have easily available derivatives, or are non-convex [30]. The basic assumption made in BO is that the unknown function which we aim to optimize is smooth in some sense, meaning it is sampled from a Gaussian process. The optimization is then solved by maintaining a posterior distribution for this function as observations are made. The posterior computation relies on the familiar Sherman-Morrison-Woodbury formula for cleverly updating the inverse of a matrix [1]. Given promising results, and increased computing power, using BO to tune the hyperparameters of our RF would seem like a good idea.

Formally, given previous observations \( x_{1:t} \), with corresponding function values \( f_{1:t} \), where \( f_i = f(x_i) \), and a new point \( x^\ast \), then the joint distribution and the properties of the updated GP
posterior are given by

\[
\begin{bmatrix}
  f_{1:t} \\
  f^*
\end{bmatrix}
\sim \mathcal{N}
\left(
  \begin{bmatrix}
    m(x_{1:t}) \\
    K(x_{1:t}, x^*)
  \end{bmatrix},
  \begin{bmatrix}
    K(x_{1:t}, x_{1:t}) & K(x_{1:t}, x^*) \\
    K(x^*, x_{1:t}) & K(x^*, x^*)
  \end{bmatrix}
\right)
\]

(1)

\[
P(f^* \mid D_t, x^*) \sim \mathcal{N}(\mu(x^* \mid D_t), \sigma(x^* \mid D_t))
\]

(2)

\[
\mu(x^* \mid D_t) = k(x^*, x_{1:t}) \mathbf{K}(x_{1:t}, x_{1:t})^{-1} f_{1:t}
\]

(3)

\[
\sigma(x^* \mid D_t) = k(x^*, x^*) - k(x^*, x_{1:t}) \mathbf{K}(x_{1:t}, x_{1:t})^{-1} k(x_{1:t}, x^*)
\]

(4)

where \(D_t = \{x_{1:t}, f_{1:t}\}\) is the data from our previous evaluations. Now, two major components characterize a BO method. The first one is the choice of the kernel matrix, \(k\), for the GP prior. The squared exponential kernel,

\[
K_{SE}(x, x') = \theta_0 e^{-\frac{1}{2} r^2(x, x')}
\]

(5)

where

\[
r^2(x, x') = \sum_{d=1}^{D} (x_d - x'_d)^2 / \theta_d^2
\]

(6)

is widely used in literature, but making assumptions of such smooth behavior when tuning hyperparameters in a high dimensional state space is unrealistic. We are therefore forced to look for better suited kernels. In this work we will investigate the so-called ARD Matern 5/2 kernel,

\[
K_{52}(x, x') = \theta_0 \left( 1 + \frac{\sqrt{5} r(x, x')}{\theta_0} + \frac{5}{3} r^2(x, x') \right) e^{-\frac{\sqrt{5} r(x, x')}{\theta_0}}
\]

(7)

which has nice properties like twice differentiability without being as smooth as the exponential kernel.

The second choice which defines BO is the choice of an acquisition function. The acquisition function is a heuristic that guides the Bayesian optimizer in choosing the next point to evaluate. It usually tries to balance the tradeoff between exploitation and exploration. Exploitation represent the desire to investigate areas which are known to be promising, i.e. areas that have high function evaluations, whereas exploration attempts to explore areas which have high variance to get a better estimate of our function space. Various acquisition functions are mentioned in literature, common ones being the Probability of Improvement [18], Expected Improvement [21], and GP Upper Confidence Bound [26]. In this paper, we will focus on Expected Improvement,

\[
\mu(x \mid D_t) = \mathbb{E} [\max \{0, f_{t+1}(x) - f(x^*)\} \mid D_t]
\]

(8)

where

\[
x_+ = \arg\max_{x \in \{x_{1:t}\}} f(x)
\]

(9)

and our next choice of point is

\[
x_{t+1} = \arg\max_{x \in X} \mu(x \mid D_t)
\]

(10)

as our acquisition function for BO. The intuition behind using Expected Improvement is that since we are trying to model our true function using a Gaussian process, we will just go ahead and choose as our next point, the point \(x\) that minimizes the value between the next function evaluation and our current best function value, irrespective of what the true function is.

3 Methodology

In this Section, we explain the three supervised learning classification problems we attempt to solve using RFs with optimized hyperparameters.
Table 1: The table shows the ranges over which Bayesian optimization was performed for parameters of our model for optical character recognition.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Range</th>
</tr>
</thead>
<tbody>
<tr>
<td>Max. tree depth</td>
<td>[2, 20]</td>
</tr>
<tr>
<td>Min. # of samples for inner nodes</td>
<td>[1, 100]</td>
</tr>
<tr>
<td>Max. # of features considered</td>
<td>[1, 64]</td>
</tr>
</tbody>
</table>

Table 2: The table shows the ranges over which Bayesian optimization was performed for parameters of our model for Twitter sentiment classification.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Range</th>
</tr>
</thead>
<tbody>
<tr>
<td>Max. tree depth</td>
<td>[2, 20]</td>
</tr>
<tr>
<td>Min. # of samples for inner nodes</td>
<td>[1, 100]</td>
</tr>
<tr>
<td>Max. # of features considered</td>
<td>[1, 64]</td>
</tr>
</tbody>
</table>

3.1 Optical Character Recognition

OCR is the mechanical or electronic conversion of handwritten or printed text into machine-encoded text [24]. It has widespread applications in machine translation and text mining and has achieved a lot of attention from researchers for decades [8] [15]. In our work, we perform OCR with an RF and automatically tune its parameters using BO. The dataset is available at [29] and contains 5620 data points with 64 attributes. We optimize respectively, the maximum depth of trees, the minimum size of an inner-node, and the maximum number of features considered for splitting an inner-node. We decided not to optimize the number of trees in our forest, because the accuracy of RFs increases monotonically in the number of trees [30]. Table 1 shows the ranges for the various parameters of the RF for which BO was performed.

3.2 Twitter Data Sentiment Classification

The next supervised learning classification problem we looked into was classification of tweets based on sentiments. We only considered the problem of binary classification; happy and sad tweets. We used the dataset available at [28] to perform this task. The dataset which we have used has 40000 tweets processed in a bag-of-words model [31], with the feature vector of 10000 dimensions. Here our RF will attempt to perform classification in a very high dimensional and sparse state space. Again we have use BO to tune the depth of the tree, the minimum size of an inner-node and the maximum number of features considered for splitting at a node. Table 2 shows the ranges for the various parameters of the RF for which BO was performed.

3.3 Named entity recognition

In computational linguistics the task of Named Entity Recognition (NER) is to identify known entity names such as people, places, and organizations. It seeks to classify every word in a given text into any number of predefined categories and a special category for none-of-the-above. NER is a subfield of information extraction, the process of finding specific kinds of information from unstructured or semi-structured text. For example, the sentence below contains three named entities of different classes: Pope Benedict XVI is a person, the Catholic church is arguably an organization, and Rome is a location.

[PER Pope Benedict XVI], the leader of [ORG the Catholic church] wants to retire in [LOC Rome].

Information extraction has many layers of sophistication, the highest asking detailed questions about the text and its subject matter. To successfully retrieve such information an accurate NER for the task at hand is critical. From this follows that any question-answering or machine translation
<table>
<thead>
<tr>
<th>Parameter</th>
<th>Range</th>
</tr>
</thead>
<tbody>
<tr>
<td>Max. tree depth</td>
<td>[2,20]</td>
</tr>
<tr>
<td>$B$</td>
<td>[0.01,0.40]</td>
</tr>
<tr>
<td>Max. # of features considered</td>
<td>[1,11]</td>
</tr>
</tbody>
</table>

Table 3: The table shows the ranges over which Bayesian optimization was performed for parameters of our model for NER.

systems can substantially benefit from a relevant NER classification [5]. After its definition as a separate task in the Message Understanding Conferences, NER has been extended from the task of pure information extraction to as varied applications as bioinformatics, molecular biology, text classification, and most recently to Twitter data.

Since named entity recognition can so easily be viewed as classification most standard Machine Learning methods are applicable to the problem. Like before we use an RF to perform the classification on a dataset consisting of 5000 sentences, extracted from news documents in the Reuters corpus [23]. From our corpus we extract 11 features and the total number of points is roughly 160,000. Because the data is heavily imbalanced, with many more untagged data than tagged, we follow the advice of [14] and under-sample a fraction $B$ of our untagged data and throw the rest out of our training set. Now we use BO to set the values of the maximum tree depth, maximum number of features used, and our sampling parameter, $B$. The parameter subspace is given in Table 3.

4 Experiments and Results

Here we discuss the experiments we performed and their results. We will see that they demonstrate the effectiveness of using an RF with parameters tuned automatically with BO. The experiments below were carried out on a Linux based server with 16 cores and 64GB of memory. Our experiments are coded in Python, with all BO done using software provided at [16].

4.1 Optical Character Recognition

To train our RF classifier we perform a 10-fold cross validation on our OCR dataset. For 10 iterations the model is trained on 90% of the data and tested on the remaining 10%. We output the average accuracy from these iterations as our model accuracy. To optimize our 3 free hyperparameters we perform 50 iterations of BO with an EI acquisition function. To compare the accuracy attained by our RF classifier, we also perform the classification using a Support Vector Machine [9]. We chose SVM as a comparison because it is known to do reasonably well on this task. Fig. 1 shows how our best accuracy changes with the number of iterations of the optimization process. The figure shows that BO was able to find good estimates of the optimal hyperparameters in very few iterations, and the accuracy level attained was comparable with SVM.

4.2 Twitter Data Sentiment Classification

We repeat the process from above of training the RF classifier using 10-fold cross validation on 20,000 tweets and predicting on a different set of the same size. We perform 50 iterations of BO to optimize the same set of hyperparameters as in our previous experiment above. In comparison to the accuracy of our model we also ran our data through a Support Vector Machine and a Naive Bayes classifier. As in the previous experiment we see that BO is able to find good estimates of our parameters relatively quickly. However, our classifier still doesn’t perform as well as our baseline. This could be due to the fact that we only used 10 trees for training because of computational restrictions.
Figure 1: On the (left) the graph shows the variation of the current best accuracy against the number of iterations in the Bayesian optimization process for OCR. On the (right) is a corresponding graph for our Twitter sentiment classification.

Figure 2: F-measure increases with the # of iterations of the Bayesian optimization (left) and the precision-recall curve (right) made from the output of the optimization procedure.

4.3 Named entity recognition

In our experiments for NER our method of using 10-fold cross validation remains the same. We use our method of under-sampling mentioned above to train on 90% of roughly $B \times 150,000 + 10,000$ data points and testing on the remaining 10%. Again we ran 50 iterations of our BO algorithm and output the F-measure, defined as

$$F = \frac{2PR}{P + R}$$  \hspace{1cm} (11)

where $P$ is precision and $R$ is recall. Fig. 2 shows the best F-measure achieved through the iterations of the BO procedure. It is compared to two baselines; the first is random tagging of tags found in the training set, and the second is constructed from gazettes of common and unambiguous named entities. It is worth noting that the inclusion of the bootstrapping parameter $B$ significantly improves our results. Fig. 2 also shows the precision-recall curve of the points evaluated in our optimization procedure. Contrary to the expected inverse relationship that usually exists between precision and recall, the curve shows a proportional relationship.

5 Conclusions

Taken together, the results of our experiments demonstrate that BO is a powerful method for quickly getting good estimates of the hyperparameter values for an RF. This notion of speed is in terms
of the number of model evaluations, a key consideration when dealing with huge datasets and sophisticated models in practice. The upside of using BO is consistent among experiments on three fundamentally different, yet standard datasets and classification tasks. This suggests a good generalizing behavior. By altering the hyperparameters to be optimized we also find that the BO process is robust to different kinds of model parameters, another promising hint of good generalization. As a side note, our NER experiment demonstrates the expected result that under-sampling improves our results a lot when presented with highly imbalanced data.

6 Future work

Although we are able to demonstrate nice behavior of hyperparameter tuning with BO our work raises a number of questions and suggests roadmaps for continuing work. In this study we focused on BO using only Expected Improvement as our acquisition function. It would be informative to replicate our experiments with other common heuristics, especially ones that are known to scale better when the number of parameters grows rapidly. While doing so, optimizing the hyperparameters of our baseline models would also provide a stronger comparison.

As mentioned above, optimizing the number of trees for an RF would not have been very interesting. However, if we were to redefine our model output, and include time as another dimension in our performance criterion, this would likely make the number of trees a more interesting player as a hyperparameter. This is based on the fact that with increasing number of trees, follows increased computation time. In our world of limited resources this directly maps to higher monetary costs, with obvious practical implications.

References


