Experimental Report on Hyper-parameter Optimization

Xuedong Shang

No Institute Given

Abstract.

1 Introduction

The algorithms being considered here for the moment are Hyperband [1], a Bayesian-based approach TPE [2], two hierarchical approaches HOO [3], HCT [4], and a baseline method random search. Current datasets used are MNIST dataset and some datasets from UCI machine learning dataset archive.

2 MNIST

We consider here Logistic Regression, Multi-Layer Perceptron (MLP) and Convolutional Neural Networks (CNN) as classifiers to be hyper-optimized. This part of code (code for classifers with eventual usage of GPU) is based on code available at `http://deeplearning.net/`.

Dataset The MNIST dataset is pre-split into three parts: training set D_{train}, validation set D_{valid} and test set D_{test}.

Hyper-parameters The hyper-parameters to be optimized are listed below in Table 1, 2 and 3. For logistic regression, the hyper-parameters to be considered are learning rate and mini-batch size (since we are doing mini-batch SGD). For MLP, we take into account an additional hyper-parameter which is the l_2 regularization factor. For CNN (or LeNet), we take into account the number of kernels used in the two convolutional-pooling layers.

Hyper-parameter	**Type**	**Bounds**
learning_rate	$\mathbb{R}^+$	$\left[10^{-3}, 10^{-1}\right]$ (log-scaled)
batch_size	$\mathbb{N}^+$	$[1, 1000]$

Table 1. Hyper-parameters to be optimized for logistic regression with SGD.

Hyper-parameter	**Type**	**Bounds**
learning_rate	$\mathbb{R}^+$	$\left[10^{-3}, 10^{-1}\right]$ (log-scaled)
batch_size	$\mathbb{N}^+$	$[1, 1000]$
l_2_reg	$\mathbb{R}^+$	$\left[10^{-4}, 10^{-2}\right]$ (log-scaled)

Table 2. Hyper-parameters to be optimized for MLP with SGD.

Resource Allocation The type of resource considered here is the number of epochs, where one epoch means a pass of training through the whole training set using SGD. Note that this is similar to the original Hyperband paper where one unit of resources corresponds to 100 mini-batch iterations for example. One epoch may contain a various number of mini-batch iterations depending on the mini-batch size.

Hyper-parameter	Type	Bounds
learning_rate	$\mathbb{R}^+$	$\left[10^{-3}, 10^{-1}\right]$ (log-scaled)
batch_size	$\mathbb{N}^+$	$[1, 1000]$
k_2	$\mathbb{N}^+$	$[10, 60]$
k_1	$\mathbb{N}^+$	$[5, k_2]$

Table 3. Hyper-parameters to be optimized for CNN with SGD.

Experimental Design In this section, we focus on neural network-typed classifiers, we will thus be maximizing the likelihood of the training set D_{train} under the model parameterized by θ (in this section, θ corresponds to (W, b) where W is the weight matrix and b is the bias vector):

$$\mathcal{L}(\theta, D_{\text{train}}) = \frac{1}{|D_{\text{train}}|} \sum_{i=1}^{|D_{\text{train}}|} \log(\mathbb{P}(Y = y^{(i)}|x^{(i)}, \theta),$$

which is equivalent to minimize the loss function:

$$\ell(\theta, D_{\text{train}}) = -\mathcal{L}(\theta, D_{\text{train}}).$$

At each time step t, we give one unit of resources (one epoch of training here) to the current algorithm, who will run one epoch of training on the training set D_{train}. The trained model is then used to predict output values $\hat{y}_{\text{pred},t}$ and $\tilde{y}_{\text{pred},t}$ respectively over validation set D_{valid} and test set D_{test}. We then compute the number that were misclassified by the model, a.k.a. the zero-one loss on the validation and test set:

$$\ell_t(\hat{\theta}_t, D_{\text{valid}}) = \frac{1}{|D_{\text{valid}}|} \sum_{i=1}^{|D_{\text{valid}}|} \mathbb{1}_{\{\hat{y}^{(i)}_{\text{pred},t} \neq y^{(i)}\}},$$

$$\ell_t(\hat{\theta}_t, D_{\text{test}}) = \frac{1}{|D_{\text{valid}}|} \sum_{i=1}^{|D_{\text{valid}}|} \mathbb{1}_{\{\tilde{y}^{(i)}_{\text{pred},t} \neq y^{(i)}\}}.$$

During the experiment, we keep track of the best validation error and its associated test error. At each time step t, if the new validation error is smaller than the current best validation error, then we update the best validation error, and report the new test error. Otherwise we just report the test error associated with the previous best validation error.

Note that there is a very important 'keep training' notion here, which means if we are running the classifier with a same hyper-parameter configuration, then we do not restart the training from scratch. In contrary, we always keep track of previously trained weight matrix and bias vector with respect to the current hyper-parameter configuration, and train the model from these pre-trained parameters.

The total budget for Hyperband would be $B = R \times s_{\max}$, and each configuration can be evaluted for $R_{\max}$ times (this $R_{\max}$ depends only on R and $s_{\max}$). For HCT, we just need to feed the algorithm the total budget B, and the number of times that each configuration is evaluated will be decided by the algorithm itself. While for TPE, HOO and Random Search, we evaluate each configuration for $R_{\max}$ times in order to make a fair comparison, which means $B/R_{\max}$ configurations will be evaluted.

Comparison All plots below are averaged on 10 trials of experiments.

Discussion

3 UCI Datasets

We consider here Adaptive Boosting (AdaBoost), Gradient Boosting Machine (GBM), k-Nearest Neighbors (KNN), Multi-Layer Perceptron (MLP), Support Vector Machine (SVM), Decision Tree and Random Forest from Scikit-learn.

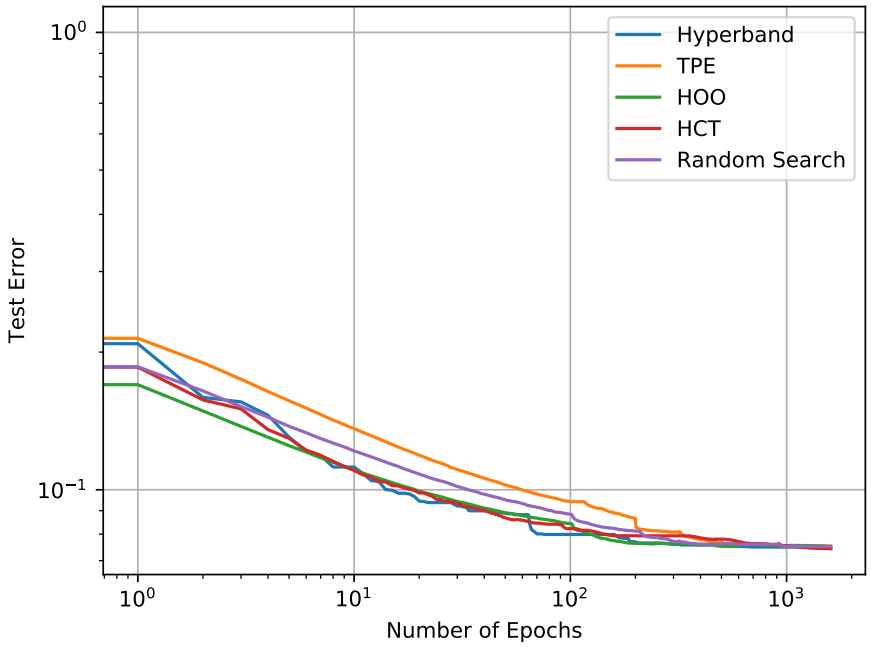

Fig. 1. Performance (log-scale) comparison of different hyper-parameter optimization algorithms on Logistic Regression, trained on MNIST Dataset.

Dataset Several datasets on UCI dataset archive (e.g. Wine, Breast Cancer, etc) are being used. They are all pre-split into a training set D_{train} and a test set D_{test}.

Hyper-parameters The hyper-parameters to be optimized are listed below in Table 4, 5, 6, 7, 8, 9 and 10.

Parameter	**Type**	**Bounds**
`learning_rate`	$\mathbb{R}^+$	$[10^{-5}, 10^{-1}]$
`n_estimators`	Integer	$\{5, \ldots, 200\}$

Table 4. Hyper-parameters to be optimized for AdaBoost models.

Resource Allocation One unit of resources in this setting is one iteration of training, which means one complete training of each classifier/regressor over the whole training set.

Experimental Design In this section we use the logarithmic loss, also known as cross-entropy for classification problem, defined by:

$$\ell(\theta, D) = -\frac{1}{|D|} \sum_{i=1}^{|D|} \sum_{j=1}^{m} y_j^{(i)} \log(\hat{p}_j^{(i)}),$$

where $\hat{p}_{ij}$ is the predicted probability of a sample i belonging to class j, and m is the number of classes considered. And for regression problems, the loss that we use is the typical mean squared error, defined by:

$$\ell(\theta, D) = -\frac{1}{|D|} \sum_{i=1}^{|D|} \left(y^{(i)} - \hat{y}_{\text{pred}}^{(i)} \right)^2.$$

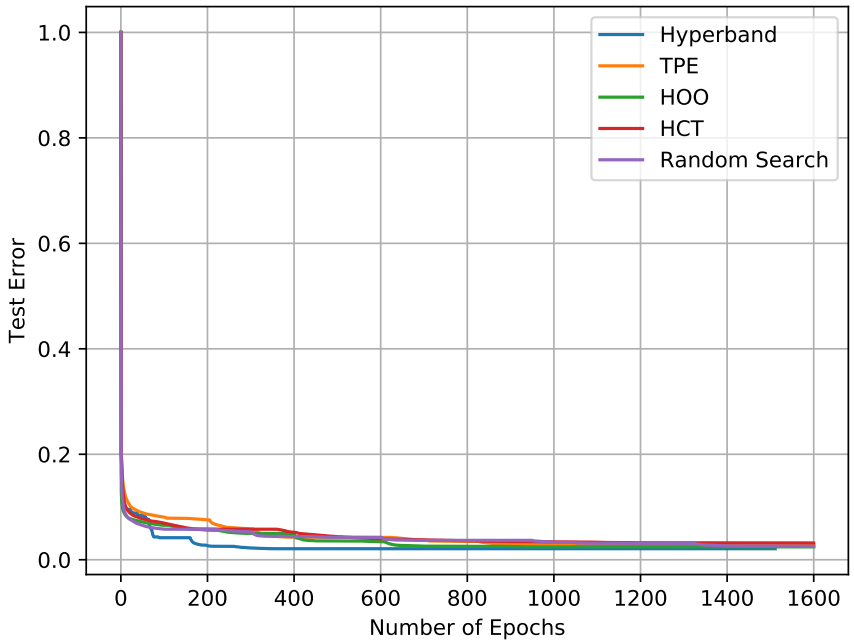

Fig. 2. Performance comparison of different hyper-parameter optimization algorithms on MLP, trained on MNIST Dataset.

Parameter	Type	Bounds
`learning_rate`	$\mathbb{R}^+$	$[10^{-5}, 10^{-2}]$
`n_estimators`	Integer	$\{10, \dots, 100\}$
`max_depth`	Integer	$\{2, \dots, 100\}$
`min_samples_split`	Integer	$\{2, \dots, 100\}$

Table 5. Hyper-parameters to be optimized for GBM models.

And for this part of experiments, we choose to perform a shuffled $k = 5$ cross-validation scheme on D_{train} at each time step t. In practice, this means that we fit 5 models with the same architecture to different train/validation splits and average the loss results in each. More precisely, for every cross-validation split $\texttt{cv}_j, j = 1 \dots 5$, we get a loss $\ell_{j,t}(\hat{\theta}_{j,t}, D_{\text{valid},j,t}) = \frac{1}{n}\sum_{i=1}^{n}\left(y_j^{(i)} - \hat{y}_{\text{pred},j,t}^{(i)}\right)^2$, where $n = |D_{\text{valid}}|$ (here we take MSE as an example, it's the same for log-loss). Thus the validation loss at time t is

$$\frac{1}{5}\sum_{j=1}^{5}\ell_{j,t}(\hat{\theta}_{j,t}, D_{\text{valid},j,t}) = \frac{1}{5n}\sum_{j=1}^{5}\sum_{i=1}^{n}\left(y_j^{(i)} - \hat{y}_{\text{pred},j,t}^{(i)}\right)^2.$$

Just like in the previous section, we can then compute and report the test error on the holdout test set D_{test}:

$$\ell_t(\hat{\theta}_t, D_{\text{test}}) = \frac{1}{|D_{\text{test}}|}\sum_{i=1}^{|D_{\text{test}}|}\left(y^{(i)} - \tilde{y}_{\text{pred},t}^{(i)}\right)^2.$$

Note that under this experimental environment, 'keep training' does not make sense anymore. Thus for HOO, TPE and Random Search, we only need to evaluate each configuration once, contrarily to what we did in the previous setting. While for Hyperband, we still need to evaluate each configuration for a certain times based on R and $s_{\max}$.

Comparison Each plot here is averaged on 20 runs of experiments.

Parameter	Type	Bounds
k	Integer	$\{10, \ldots, 50\}$

Table 6. Hyper-parameters to be optimized for KNN models.

Parameter	Type	Bounds
`hidden_layer_size`	Integer	$[5, 50]$
`alpha`	$\mathbb{R}^+$	$[0, 0.9]$

Table 7. Hyper-parameters to be optimized for MLP models.

Discussion

References

1. Li, L., Jamieson, K., DeSalvo, G., Rostamizadeh, A., & Talwalkar, A. (2016). Hyperband: A novel bandit-based approach to hyperparameter optimization. arXiv preprint arXiv:1603.06560.
2. Bergstra, J. S., Bardenet, R., Bengio, Y., & Kgl, B. (2011). Algorithms for hyper-parameter optimization. In Advances in neural information processing systems (pp. 2546-2554).
3. Bubeck, S., Munos, R., Stoltz, G., & Szepesvri, C. (2011). X-armed bandits. Journal of Machine Learning Research, 12(May), 1655-1695.
4. Azar, M. G., Lazaric, A., & Brunskill, E. (2014). Online stochastic optimization under correlated bandit feedback. In Proceedings of the 31st International Conference on Machine Learning (ICML-14) (pp. 1557-1565).

Parameter	Type	Bounds
C	$\mathbb{R}^+$	$[10^{-5}, 10^5]$ (log-scaled)
γ	$\mathbb{R}^+$	$[10^{-5}, 10^5]$ (log-scaled)

Table 8. Hyper-parameters to be optimized for SVM models.

Parameter	Type	Bounds
`max_features`	$\mathbb{R}^+$	$[0.01, 0.99]$
`max_depth`	Integer	$\{4, \ldots, 30\}$
`min_samples_split`	$\mathbb{R}^+$	$[0.01, 0.99]$

Table 9. Hyper-parameters to be optimized for Decision Tree models.

Parameter	Type	Bounds
`max_features`	$\mathbb{R}^+$	$[0.1, 0.5]$
`n_estimators`	Integer	$\{10, \ldots, 50\}$
`min_samples_split`	$\mathbb{R}^+$	$[0.1, 0.5]$

Table 10. Hyper-parameters to be optimized for Random Forest models.