Hyperopt-Sklearn: Automatic Hyperparameter Configuration for Scikit-Learn

Brent Komer BRENT.KOMER@UWATERLOO.CA
James Bergstra JAMES.BERGSTRA@UWATERLOO.CA
Chris Eliasmith CELIASMITH@UWATERLOO.CA
Centre for Theoretical Neuroscience
University of Waterloo

Abstract

Hyperopt-sklearn is a new software project that provides automatic algorithm configuration of the Scikit-learn machine learning library. Following Auto-Weka, we take the view that the choice of classifier and even the choice of pre-processing module can be taken together to represent a *single large hyperparameter optimization problem*. We use Hyperopt to define a search space that encompasses many standard components (e.g. SVM, RF, KNN, PCA, TFIDF) and common patterns of composing them together. We demonstrate, using search algorithms in Hyperopt and standard benchmarking data sets (MNIST, 20-Newsgroups, Convex Shapes), that searching this space is practical and effective. In particular, we improve on best-known scores for the model space for both MNIST and Convex Shapes.

1. Introduction

The size of data sets and the speed of computers have increased to the point where it is often easier to fit complex functions to data using statistical estimation techniques than it is to design them by hand. The fitting of such functions (training machine learning algorithms) remains a relatively arcane art, typically mastered in the course of a graduate degree and years of experience. Recently however, techniques for automatic algorithm configuration based on Regression Trees (Hutter et al., 2011), Gaussian Processes (Mockus et al., 1978; Snoek et al., 2012), and density-estimation techniques (Bergstra et al., 2011) have emerged as viable alternatives to hand-tuning by domain specialists.

Hyperparameter optimization in machine learning systems was first applied to neural networks and convnets, where the number of parameters can be overwhelming: for example Bergstra et al. (2011) tuned Deep Belief Networks with up to 32 hyperparameters, and Bergstra et al. (2013a) showed that similar methods could be useful even in a convnet model with 238 hyperparameters. Relative to DBNs and convnets, algorithms such as RBF-SVMs and Random Forests (RFs) have a small-enough number of hyperparameters that manual tuning and grid or random search provides satisfactory results.

Taking a step back though, there is often no particular reason to use either an RBF-SVM or an RF when they are both computationally viable. A model-agnostic practitioner may simply prefer to go with the one that provides greater accuracy. In this light, *the choice of*

classifier can be seen as hyperparameter alongside the C-value in the SVM and the max-tree-depth of the RF. Indeed the choice and configuration of *pre-processing* components may likewise be seen as part of the model selection / hyperparameter optimization problem. The Auto-Weka project (Thornton et al., 2013) was the first to show that an entire library of machine learning approaches (Weka (Hall et al., 2009)) can be searched within the scope of a single run of hyperparameter tuning. However, Weka is a GPL-licensed Java library, and was not written with scalability in mind. These factors limit the utility of Auto-Weka. Scikit-learn (Pedregosa et al., 2011) is another library of machine learning algorithms that is written in Python with many modules in C for greater speed, and is BSD-licensed. Scikit-learn is widely used in the scientific Python community and supports many machine learning application areas.

With this paper we introduce Hyperopt-Sklearn – a project that brings the benefits of automatic algorithm configuration to users of Python and scikit-learn.[1] Hyperopt-Sklearn uses Hyperopt (Bergstra et al., 2013b) to describe a search space over possible configurations of Scikit-Learn components, including pre-processing and classification modules. Section 2 describes our configuration space of 7 classifiers and 4 preprocessing modules that encompasses a strong set of classification systems for dense and sparse feature classification (of images and text). Section 3 presents experimental evidence that search over this space is viable, meaningful, and effective. Section 4 presents a discussion of the results, and directions for future work.

2. Searching Scikit-learn with Hyperopt

The configuration space we experiment on below includes six preprocessing algorithms and seven classification algorithms. The full search space is illustrated in Figure 1. The preprocessing algorithms were (by class name, followed by n. hyperparameters + n. unused hyperparameters): `PCA`(2), `StandardScaler`(2), `MinMaxScaler`(1), `Normalizer`(1), `None`, and `TF-IDF`(0+9). The first four preprocessing algorithms were for dense features. PCA performed whitening or non-whitening principle components analysis. The `StandardScaler`, `MinMaxScaler`, and `Normalizer` did various feature-wise affine transforms to map numeric input features onto values near 0 and with roughly unit variance. The `TF-IDF` pre-processing module performed feature extraction from text data. The classification algorithms were (by class name (used + unused hyperparameters)): `SVC`(23), `KNN`(4+5), `RandomForest`(8), `ExtraTrees`(8), `SGD`(8+4), and `MultinomialNB`(2). The `SVC` module is a fork of LibSVM, and our wrapper has 23 hyperparameters because we treated each possible kernel as a different classifier, with its own set of hyperparameters: Linear(4), RBF(5), Polynomial(7), and Sigmoid(6).

In total, our parameterization had 65 hyperparameters: 6 for preprocessing and 53 for classification. The search space includes 15 boolean variables, 14 categorical, 17 discrete, and 19 real-valued variables. Although the total number of hyperparameters is large, the number of *active* hyperparameters describing any one model is much smaller: a model consisting of `PCA` and a `RandomForest` for example, would have only 12 active hyperparameters (1 for the choice of preprocessing, 2 internal to PCA, 1 for the choice of classifier and 8 internal to the RF). Hyperopt description language allows us to differentiate between

1. Project hosted at http://hyperopt.github.com/hyperopt-sklearn

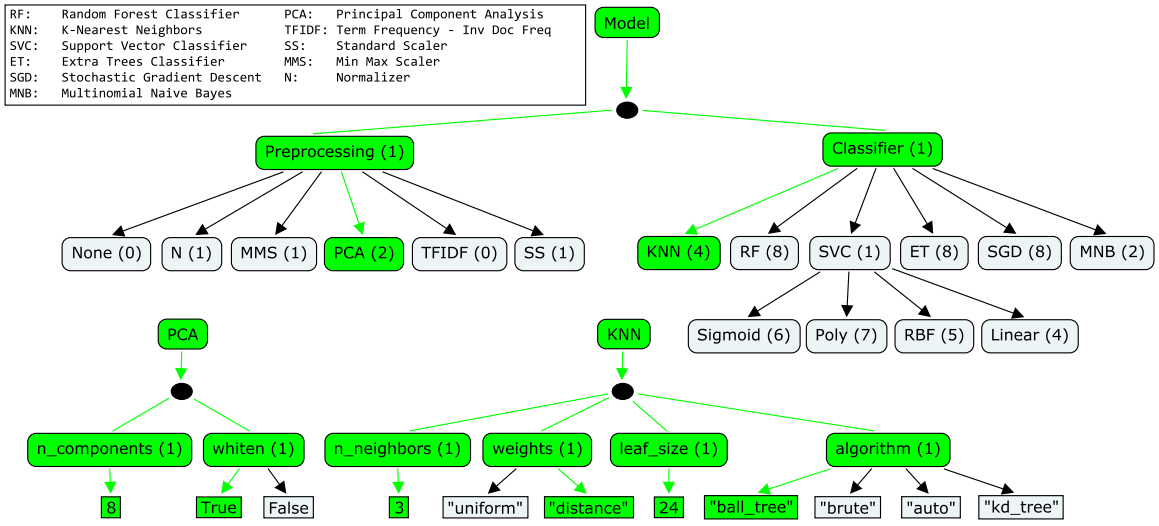

Figure 1: Hyeropt-sklearn's full search space ("Any Classifier") consists of a (preprocessing, classsifier) pair. There are 6 possible preprocessing modules and 6 possible classifiers. Choosing a model within this configuration space means choosing paths in an ancestral sampling process. The highlighted green edges and nodes represent a (PCA, K-Nearest Neighbor) model. The number of active hyperparameters in a model is the sum of parenthetical numbers in the selected boxes. For the PCA+KNN combination, 7 hyperparameters are activated.

conditional hyperparameters (which must always be assigned) and *non-conditional* hyperparameters (which may remain unassigned when they would be unused). We make use of this mechanism extensively so that Hyperopt's search algorithms do not waste time learning by trial and error that e.g. RF hyperparameters have no effect on SVM performance. Even internally within classifiers, there are instances of conditional parameters: `KNN` has conditional parameters depending on the distance metric, and `LinearSVC` has 3 binary parameters ("loss", "penalty", and "dual") that admit only 4 valid joint assignments. We also included a blacklist of (preprocessing, classifier) pairs that did not work together, e.g. PCA and MinMaxScaler were incompatible with MultinomialNB, TF-IDF could only be used for text data, and the tree-based classifiers were not compatible with the sparse features produced by the TF-IDF preprocessor. Allowing for a 10-way discretization of real-valued hyperparameters, and taking these conditional hyperparameters into account, a grid search of our search space would still require an infeasible number of evalutions (on the order of 10^{12}).

Following Scikit-learn's convention, hyperopt-sklearn provides an `Estimator` class with a `fit` method and a `predict` method. The `fit` method of this class performs hyperparameter optimization, and after it has completed, the `predict` method applies the best model to test data. Hyperopt makes it possible to parallelize the model search over a cluster, with communication handled via a MongoDB instance. Each evaluation during optimization performs training on a large fraction of the training set, estimates test set accuracy on a

Table 1: Hyperopt-sklearn scores (bold) relative to selections from literature on the three data sets used in our experiments. On MNIST, hyperopt-sklearn is one of the best-scoring methods that does not use image-specific domain knowledge (these scores and others may be found at http://yann.lecun.com/exdb/mnist/). On 20 Newsgroups, hyperopt-sklearn is competitive with similar approaches from the literature (scores taken from Guan et al. (2009)). On Convex Shapes, hyperopt-sklearn outperforms previous automatic algorithm configuration approaches (Eggensperger et al., 2013) and manual tuning (Larochelle et al., 2007).

MNIST		**20 Newsgroups**		**Convex Shapes**	
Approach	Accuracy	Approach	F-Score	Approach	Accuracy
Committee of convnets	99.8%	CFC	0.928	**hyperopt-sklearn**	**88.7**%
hyperopt-sklearn	**98.7**%	**hyperopt-sklearn**	**0.856**	hp-dbnet	84.6%
libSVM grid search	98.6%	SVMTorch	0.848	dbn-3	81.4%
Boosted trees	98.5%	LibSVM	0.843		

In the 20 Newsgroups dataset, the score reported for hyperopt-sklearn is the weighted-average F1 score provided by sklearn. The other approaches shown here use the macro-average F1 score.

validation set, and returns that validation set score to the optimizer. At the end of search, the best configuration is retrained on the whole data set to produce the classifier that handles subsequent `predict` calls.

3. Experiments

We conducted experiments on three data sets to establish that hyperopt-sklearn can find accurate models on a range of data sets in a reasonable amount of time. Results were collected on three data sets: MNIST, 20-Newsgroups, and Convex Shapes. MNIST is a well-known data set of 70K 28×28 greyscale images of hand-drawn digits (LeCun et al., 1998). 20-Newsgroups is a 20-way classification data set of 20K newsgroup messages (Mitchell (1996), we did not remove the headers for our experiments). Convex Shapes is a binary classification task of distinguishing pictures of convex white-colored regions in small (32×32) black-and-white images (Larochelle et al., 2007).

To establish that searching the full space is effective, we performed optimization runs of up to 300 function evaluations searching either the entire space, or else subspaces that corresponded to specific classifier types. We used three optimization algorithms in Hyperopt: random search, annealing, and TPE. Figure 2 shows that the performance of the model found from throughout the entire search space was not statistically inferior to the best model pulled from each classifier subspace; there was no penalty for keeping all options open during search. Table 1 lists the test set scores of the best models found by cross-validation, as well as some points of reference from previous work. Hyperopt-sklearn's scores are relatively good on each data set, indicating that with hyperopt-sklearn's parameterization, Hyperopt's optimization algorithms are competitive with human experts.

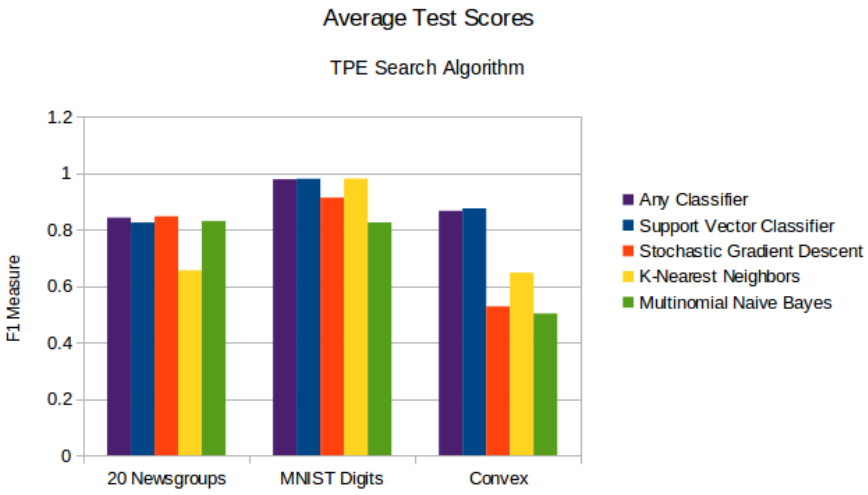

Figure 2: For each data set, searching the full configuration space ("Any Classifier") delivered performance approximately on par with a search that was restricted to the best classifier type. (Best viewed in color.)

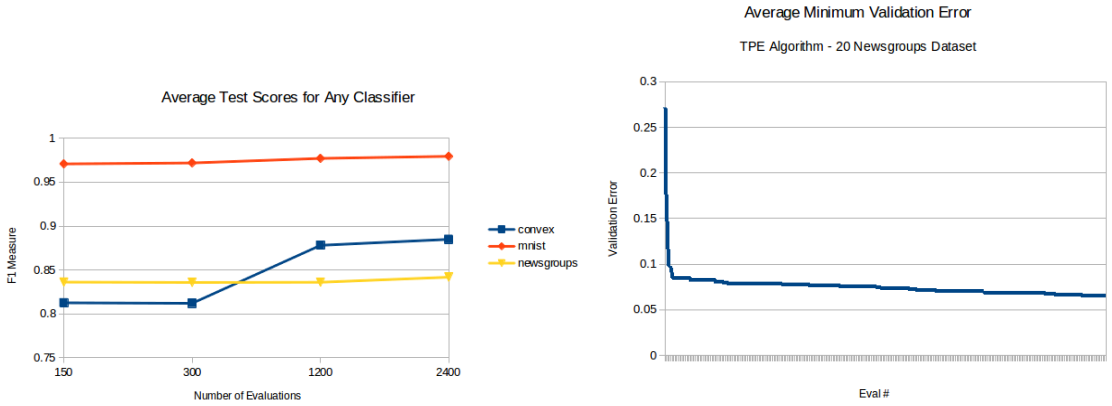

Figure 3: Left: Using Hyperopt's Anneal search algorithm, increasing the number of function evaluations from 150 to 2400 lead to a modest improvement in accuracy on 20 Newsgroups and MNIST, and a more dramatic improvement on Convex Shapes. We capped evaluations to 5 minutes each so 300 evaluations took between 12 and 24 hours of wall time. Right: TPE makes gradual progress on 20 Newsgroups over 300 iterations and gives no indication of convergence.

4. Discussion and Future Work

Hyperopt-sklearn provides many opportunities for future work. Certainly, there are more classifiers and preprocessing modules that could be included in the search space, and there are more ways to combine even the existing components. In expanding the search space,

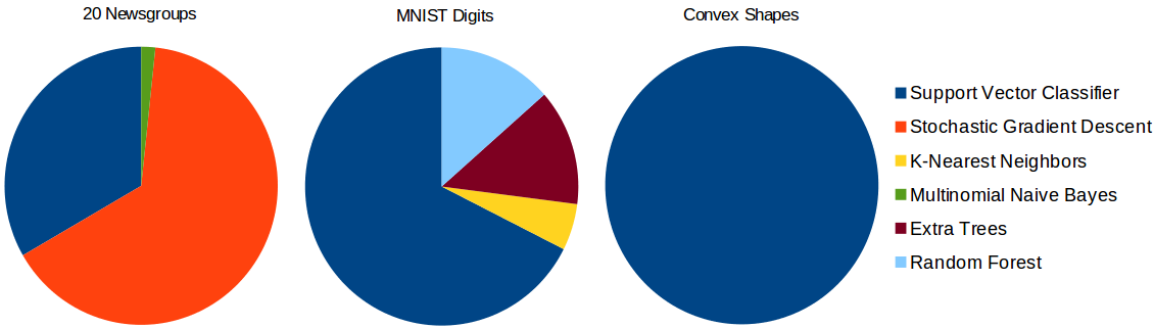

Figure 4: Looking at the best models from all optimization runs performed on the full search space (using different initial conditions, and different optimization algorithms) we see that different data sets are handled best by different classifiers. SVC was the only classifier ever chosen as the best model for Convex Shapes, and was often found to be best on MNIST and 20 Newsgroups.

care must be taken to ensure that the benefits of new models outweigh the greater difficulty of searching a larger space.

We have shown here that Hyperopt's random search, annealing search, and TPE algorithms make Hyperopt-sklearn viable, but the slow convergence in e.g. Figure 3 suggests that other optimization algorithms might be more call-efficient. The development of Bayesian optimization algorithms is an active research area, and we look forward to looking at how other search algorithms interact with hyperopt-sklearn's search spaces.

Computational wall time spent on search is of great practical importance, and hyperopt-sklearn currently spends a significant amount of time evaluating points that are un-promising. Techniques for recognizing bad performers early could speed up search enormously. Relatedly, hyperopt-sklearn currently lacks support for K-fold cross-validation. In that setting, it will be crucial to follow SMAC in the use of racing algorithms to skip un-necessary folds.

Another direction for future work is the extention of the techniques presented here in terms of classification to other types of machine learning problems (e.g. regression, density estimation, and ranking), and other types of input modalities (e.g. large images, sound, timeseries, preferences).

5. Conclusions

We have introduced Hyperopt-sklearn, a Python package for automatic algorithm configuration of standard machine learning algorithms provided by Scikit-Learn. Hyperopt-sklearn provides a unified view of 6 possible preprocessing modules and 6 possible classifiers, yet with the help of Hyperopt's optimization functions it is able to both rival and surpass human experts in algorithm configuration. We hope that it provides practitioners with a useful tool for the development of machine learning systems, and automatic machine learning researchers with benchmarks for future work in algorithm configuration.

Acknowledgements
This research was supported by the NSERC Banting Fellowship program, the NSERC Engage program and by D-Wave Systems.

References

J. Bergstra, R. Bardenet, Y. Bengio, and B. Kégl. Algorithms for hyper-parameter optimization. In *NIPS*24*, pages 2546–2554, 2011.

J. Bergstra, D. Yamins, and D. D. Cox. Making a science of model search: Hyperparameter optimization in hundreds of dimensions for vision architectures. In *In Proc. ICML*, 2013a.

J. Bergstra, D. Yamins, and D. D. Cox. Hyperopt: A Python library for optimizing the hyperparameters of machine learning algorithms. In *SciPy'13*, 2013b.

K. Eggensperger, M. Feurer, F. Hutter, J. Bergstra, J. Snoek, H. Hoos, and K. Leyton-Brown. Towards an empirical foundation for assessing bayesian optimization of hyperparameters. In *NIPS workshop on Bayesian Optimization in Theory and Practice*, 10 December 2013.

H. Guan, J. Zhou, and M. Guo. A class-feature-centroid classifier for text categorization. In *Proceedings of the 18th international conference on World wide web*, pages 201–210. ACM, 2009.

M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I. H. Witten. The weka data mining software: an update. *ACM SIGKDD explorations newsletter*, 11(1):10–18, 2009.

F. Hutter, H. Hoos, and K. Leyton-Brown. Sequential model-based optimization for general algorithm configuration. In *LION-5*, 2011. Extended version as UBC Tech report TR-2010-10.

H. Larochelle, D. Erhan, A. Courville, J. Bergstra, and Y. Bengio. An empirical evaluation of deep architectures on problems with many factors of variation. In *ICML*, pages 473–480, 2007.

Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, November 1998.

T. Mitchell. 20 newsgroups data set. http://qwone.com/ jason/20Newsgroups/, 1996.

J. Mockus, V. Tiesis, and A. Zilinskas. The application of Bayesian methods for seeking the extremum. In L.C.W. Dixon and G.P. Szego, editors, *Towards Global Optimization*, volume 2, pages 117–129. North Holland, New York, 1978.

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

J. Snoek, H. Larochelle, and R. P. Adams. Practical Bayesian optimization of machine learning algorithms. In *Neural Information Processing Systems*, 2012.

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