An Empirical Evaluation of Supervised Classifiers: Revisited

Simon Haxby
Department of Mathematics
University of California, San Diego
9500 Gilman Drive, La Jolla, CA 92642
shaxby@ucsd.edu

Abstract

In the last decade, several comprehensive empirical evaluations of supervised machine learning algorithms has been undertaken most notably by Caruna and Niculescu-Mizil in 2006 and 2008. In this report we explore a light extension and small-scale replication of their work; comparing the performance of many prominent supervised learning algorithms across several disparate datasets. While, this report did not replicate their procedures exactly, it was executed largely in the same spirit.

1 Introduction

In the 1990s and early 2000s machine learning researchers saw a flurry of high-powered, easy-to-use, minimal fine-tuning supervised machine learning algorithms emerge; most of which are still widely in use today as first-choice tools. However, it was not until 2006 and 2008 when Caruna and Niculescu-Mizil [2]-[3] executed a comprehensive empirical evaluation of these methods side by side across numerous datasets and under various metrics. The 2006 study compared 10 supervised machine learning algorithms across 11 datasets under 8 different metrics; while latter emulated the former but using high dimensional datasets. Their results reflected the consequences of the No-Free-Lunch Theorem [9]: that in given class of supervised learning problems, no one algorithm will outperform all other algorithms on every problem in that class. However, there did emerge several algorithms that were superior on *most* problems averaged over the different metrics: those being boosted decision trees, random forests and bagged trees[1]. And in the 2008 study they showed that some algorithms scaled better than others and/or performed better (or significantly worse) in high-dimensional spaces [3].

In this report we undertook a light extension of their work, by applying their methodology to two datasets previously unexamined in their papers, and to a very prominent benchmark dataset in computer vision [4], which was explored in their 2008 study [3]. Our choice of algorithms was motivated by reasons stated in the section below. One notable omission however is artificial neural networks (ANN); our reasoning for doing so is stated in the discussion. Bagging was also ommitted for no other reason than the researcher forgot about it. Furthermore, in this report we did not use model calibration as was employed in both Caruna studies nor did we use bootstrap analysis.

[1] After model calibration; see [2] for details.

2 Methodology

2.1 Learning Algorithms

In this section we briefly describe the supervised classification learning algorithms used in the this experiment. These methods were chosen for their prominence, effectiveness, availability and ease of use (lack of manual fine-tuning). All of the algorithms utilized were implemented using the scikit-learn API [6], and all the hyperparameters were optimized using a 3-fold grid search cross validation (CV) scheme. In this experiment we tried to keep the hyperparameter choices and ranges as close to the [2] original study, but due computational and implementation constraints this was not always feasible.

Logistic Regression: We used a LIBLINEAR [7] implementation for the two binary classification tasks, a native scikit-learn conjugate gradient Newton (CG-Newton) method for the multi-label task; since LIBLINEAR did not support multi-label classification. Also, for the multi-label classification we used a soft-max activation. For LIBLINEAR a L1 penalty term was used, for CG-Newton only an L2 penalty was supported, and thus used. For both implementations, the penalty parameter set was: [.001,.01,.1,1,10,100].

K-Nearest Neighbors (KNN): We used a Euclidean distance metric and $1/r^2$ prototype weight, where r is the distance from the point of interest to the given prototype. The choices of k were: [1,2,3,5,7,10,15,25,50,100,500].

Support Vector Machine (SVM): We used an SVM with a polynomial kernel, varying the powers, p, from 1-3. Where the polynomial kernel, with p=1, is effectively a linear SVM. For the cost parameter C, the CV set was: [.01,1,10,100,500].

Boosting: We used gradient boosting with decision trees as the base learners [4]. The properties of the decision trees were unconstrained (depth, feature selection, etc.); but the number of learners and the learning rate were tuned via CV. For these hyperparameters, the choices were 512, 1024 and .01, .1 respectively.

Random Forests: Here the only hyperparameter tuned during CV was feature pool size at each splitting, whose value set was: [1,2,4,6,8,12,16,20]. Where the number of base decision tree constructed for each ensemble was always 1024. Furthermore, for Random Forests we exploited Out-Of-Bag (OOB) estimates [4] for CV.

2.2 Datasets

We compared the performance of the five supervised machine learning algorithms described above on three problems/datasets. The first two the datasets used in these experiments were retrieved from the UCI Machine Learning Repository [1]. These datasets were a banking-marketing dataset, whose features were mixed categorical and real-valued and whose target was binary: whether an prospective client would subscribe to a term deposit. And a mushroom dataset, whose features were entirely categorical; describing properties of 23 species of mushrooms, and the target was binary: whether the mushroom was poisonous or not. Lastly the MNIST dataset, compiled by LeCun et al. [5], consisted of 28×28 pixel images of digits 0-9; where the target label was the digit represented by the pixel-map.

For the banking-marketing dataset, 10 base features were categorical and represented using one-hot encoding thus expanding the feature number from 17 to 75. The remaining features were real-valued and consequently Z-scored.

All of the mushroom dataset features were categorical and in identical fashion to the banking-marketing data, transformed from 23 features to 113 via one-hot encoding.

Since all the base features for the MNIST dataset were real-valued (integer), from 0-255, corresponding to intensity on the grey-scale, we Z-scored each pixel accordingly.

Table 1: Dataset Properties

Problem/Dataset	# Attributes	Training	Test	+ Target Label %
Bank Marketing	17 (75)	5000	40211	88%
Mushroom	23 (113)	5000	3124	51%
Digits (MNIST)	784	10000	60000	10% x 10

Table 2: Algorithm Performances

Classifer	Bank ACC	Mushroom ACC	MNIST ACC	Mean ACC
SVM (Poly Kernel)	0.8969	1.0	0.9620	0.9530
Gradient Boosting (DT)	0.9027	1.0	0.9502	0.9510
Random Forest	0.9009	1.0	0.9502	0.9504
KNN	0.8970	1.0	0.9487	0.9485
Logistic Regression	0.9001	1.0	0.9108	0.9370

2.3 Experimental Procedure

All of the classifiers were trained using the CV scheme described above. The hyperparameters that yielded optimal performance on CV were then selected for the models that would be used to evaluate the test set; the split of training and testing can be seen in the table above. For CV and testing the scoring metric we used was raw classification accuracy. Initially we split all the training/testing sets such that the training sets contained 5000 examples, but due to the variability of the MNIST dataset more examples were required to obtain a remotely competitive scoring [2].

3 Results

The results were interesting, and somewhat unexpected. From the performance of the classifiers on the Mushroom dataset it is clear to see that the categories were linearly separable in the feature space, as the best CV accuracy and test accuracy was 100% for all classifiers. For the Bank dataset, boosting yield the best performance followed by random forests and KNN was the worst. But the differences between all the classifiers was negligible since the difference between boosting and KNN was $\approx .7\%$. Performance on the MNIST dataset was most the discriminating in terms of performances. Here the SVM outperformed all the other algorithms by $\approx 1\%$ and logistic regression underperformed the other classifiers by $> 3\%$. Boosting and random forests came jointly in 2^{nd} at 95.02% and SVM acquired an accuracy score of 96.20%. Thus overall the SVM had greatest mean performance, primarily driven by it's score on the MNIST dataset. In term of ranking, boosting had the best overall rank coming in 1^{st} and 2^{nd} on the discriminating problems, followed by random forests and then the SVM.

Table 2 describes the results of the experiments. Here ACC is simply the test accuracy of the classifier with the CV maximizing hyperparameters.

4 Discussion

The results of this paper should be taken very lightly. Compared to the original studies [2]-[3], our report only compared 5 classifiers on 3 datasets, and perhaps most importantly on one metric: accuracy. One of the datasets had more 88% of its labels in one class and considering that mean performance on the test sets was approximately 90%, it is feasible that almost all the labels could have been assigned to that class; thus the accuracy metric may not serve as a good measure of classifier performance compared to an F-score on this problem. Also, by increasing the training set size to 70% of the data on the MNIST digit classification problem, (while using the same hyperparameter configuration) we can achieve a test accuracy of 98.11% with an SVM; which is an accuracy score

[2]This was based on the researcher's previous experience working with the MNIST dataset.

very close to LeCun's result in [5] under similar model conditions. But due to computational/time constraints validating with such a large training set was not feasible.

These kinds of sensitivities motivated us to leave out artificial neural networks (ANNs) in our report. The array of implementation/design choices that an ANN user is faced with compared other prominent supervised methods is vast: from network architectures, activations, optimizers, regularizers to initializations. All of which (compared to the methods used in these paper) sensitively affect the performance of the network. Thus, we felt it unfair to honestly include ANNs in a experiment of this nature; where ANNs are by nature a specialist tool that must be heavy fine-tuned to a task for optimal performance.

Generally, there are many experimental choices and design constraints that affect experimental outcomes in these types of empirical comparisons, such as size of train/test split, CV scheme, hyperparameter choices and ranges, and also evaluation metrics. Perhaps most importantly is feature selection. An observation from the modern computer vision literature is that the classifier employed is *significantly* less important than the feature representation of the problem at hand [10]. Indeed, this is the underpinning of deep learning, where-in a rich, powerful feature hierarchy can be constructed from the base features and then be effectively classified by any of the classifiers studied here for more or less same performance, qualitatively [8].

5 Conclusion

While empirical supervised learning experiments like the one undertaken here and in [2]-[3] provide valuable insight into the types of problems and metrics that certain classifiers excel on. One should be cautious about reading to far into their results and implications. Generally, it has been observed that feature selection/extraction and engineering plays a much more pivotal role in the performance of the modern supervised learning algorithms explored in this paper, rather than the algorithm selected from this pool [4].

6 References

[1] Blake, C., Merz, C. (1998). UCI repository of machine learning databases.

[2] Caruana R. Niculescu-Mizil A. (2006, June). An empirical comparison of supervised learning algorithms. The Proceedings of the 23rd International Conference on Machine Learning (pp. 161-168). ACM.

[3] Caruana, R., Karampatziakis, N., Yessenalina, A. (2008, July). An empirical evaluation of supervised learning in high dimensions. In Proceedings of the 25th International Conference on Machine Learning (pp. 96-103). ACM.

[4] Hastie, T., Tibshirani, R., Friedman, J., Franklin, J. (2005). Elements of Statistical Learning: Data Mining, Inference and Prediction. The Mathematical Intelligence, 27(2), 83-85.

[5] LeCun, Y., Bottou, L., Bengio, Y., Haffner, P. (1998). Gradient-based learning applied to document recognition. Proceedings of the IEEE, 86(11), 2278-2324.

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

[7] R.E. Fan, K.W. Chang, C.J. Hsieh, X.-R. Wang, and C.-J. Lin. (2008). LIBLINEAR: A library for large linear classification Journal of Machine Learning Research (9), 1871-1874.

[8] Tang, Y. (2013). Deep learning using linear support vector machines. arXiv preprint arXiv:1306.0239.

[9] Wolpert, D. H. (2002). The supervised learning no-free-lunch theorems. In Soft Computing and Industry (pp. 25-42). Springer London.

[10] Zeiler, M. D., Fergus, R. (2014). Visualizing and understanding convolutional networks. In Computer visionECCV 2014 (pp. 818-833). Springer International Publishing.