A Comparative Study Using Cross-validation to Optimize Hyper-parameters for Supervised Learning Classifiers

Binh Nguyen[1,2]

[1]Department of Bioengineering, University of California, San Diego
[2]Department of Cognitive Science, University of California, San Diego

Abstract

Supervised learning algorithms provide researchers and scientists the tools to apply machine learning in classifying and predicting new data. While their applications can be applied to general problems, some perform better depending on the problem being solved. Particular algorithms produce better results for certain datasets while some do poorly on others. Cross-validation provides a robust away to hyper-parameterize such algorithms. This paper evaluates some supervised learning algorithms to emphasize these key points in building accurate models.

1. Introduction

The applications of machine learning algorithms have become increasingly prevalent in recent decades. Due to modern advances in technology, the improved performance and efficiency of processors have alleviated the high computational cost in evaluating large data. As computers became faster, new algorithms were developed to tackle more complex learning problems. Supervised learning is a paradigm in machine learning responsible for classifying similar data using knowledge of some part of the data. Supervised learning algorithms utilize a labeled training set and a testing set of data to *learn* possible patterns that can be represented by a function. This function is able to map new data from what it has learned during training.

Many supervised learning algorithms exist to tackle different machine learning problems. Its performance depends greatly on what kind of problem is being solved. It is helpful to know in what situation a particular supervised learning algorithm will perform the best. In Caruana and Niculescu-Mizil's paper, several supervised learning algorithms are tested to explore this (Caruana et al.). This paper will attempt to elucidate the performances of a few supervised learning algorithms. In particular, bagged decision trees, k-nearest neighbors, and artificial neural networks will be explored. These algorithms were tested on 3 binary classification problems.

2. Methods

The first dataset used is the LETTER recognition dataset from the UCI Machine Learning Repository. It consists of multivariate data having 20000 data points and 16 features. The objective is to identify black-and-white rectangular pixel displays as one of 26 capital letters in the English alphabet. In this problem, letters A-M are treated as positive and letters N-Z are treated as negative. This allows a balanced problem where positives and negatives are approximately even.

The second dataset used is the IndianPine92 AVIRIS sensor dataset which collected 145 x 145 pixels and 224 spectral reflectance bands in wavelength range $0.4 - 2.5 * 10^{-6}\, meters$. It is comprised of roads, highways, low density housing, and other structures. The version used in this paper is the corrected version where the number of bands was reduced from 224 to 200 by removing regions of water absorption. The data is grouped into 16 classes where the class Soybean-mintill is treated as positive. The objective is to accurately distinguish that part of the band from other parts. The y label in this case is the 'ground truth' dataset which is stored on a separate file.

The last dataset used is the Yeast dataset from the UCI Machine Learning Repository having 1484 points and 8 features. The goal is to predict the cellular localization sites of proteins. A one-hot-encoding method could have been done on this problem, but the dataset is relatively small and the preliminary accuracies of the supervised learning algorithms were significantly low. Instead, the extracellular class was chosen to be positive and everything else as negative.

For each of these datasets, bagged decision trees (BAG-DT), k-nearest neighbors (KNN), and artificial neural networks (ANN) were used. The conditions and parameters used in Caruana and Niculescu-Mizil's paper were followed as close as possible, given some constraints. Tree depths of 1, 10, 100, and 1000 were tested with max_samples and max_features set to 0.5 in BAG-DT. In the first two datasets, the k-nearest neighbors method proved to be highly computational. To reduce computation time, a sample was taken from a randomized set of the original data. The first 1000 data points were selected for evaluation. A stochastic gradient descent (SGD) method was chosen for artificial neural networks to minimize the objective function as a sum of differentiable functions. This kind of neural network is a feedforward with SGD learning. One hidden layer with 100 neurons were used while momentum values for gradient descent update were 0, 0.2, 0.5, and 0.9.

In all algorithms, cross-validation using k-folds was implemented. Five folds were used thereby training on 80% of the training partition at a time and testing it on the remain 20% validation partition. By cross-validating before parameter update, the aim is to compare the accuracy of the folds and to determine the highest average training and validation accuracies for varying parameters. The accuracy was obtained using Scikit-learn's score method. Higher numbers mean better accuracy.

3. Experiment and Discussion

Table 1 shows the score for the best validation score for each algorithm for the 80% train and 20% test split. Tables 2 and 3 show best accuracies for 50/50 and 20/80 split, respectively. From these tables, it appears that k-nearest neighbors does the best in both Indian Pines and Yeast datasets. The only exception being that k-nearest neighbors does marginally worse than bagged decision tree in the 20/80 split for Indian Pines. However, both bagged decision trees and artificial neural networks have better accuracies than k-nearest neighbors in the LETTER dataset. It is difficult to conclude that KNN suffers from the lack of training data in Table 3 because their accuracies are less than a hundredth apart.

Algorithm	LETTER	Indian Pines	Yeast
BAG-DT	0.932	0.880	0.737
KNN	0.869	0.900	0.759
ANN	0.903	0.881	0.710

Table 1. Best validation accuracy for 80% train, 20% test

Algorithm	LETTER	Indian Pines	Yeast
BAG-DT	0.913	0.881	0.744
KNN	0.820	0.922	0.756
ANN	0.884	0.881	0.709

Table 2. Best validation accuracy for 50% train, 50% test

Algorithm	LETTER	Indian Pines	Yeast
BAG-DT	0.887	0.888	0.747
KNN	0.755	0.880	0.797
ANN	0.835	0.875	0.753

Table 3. Best validation accuracy for 20% train, 80% test

It may be more beneficial to analyze the algorithms separately, based on the selected values for their parameters. By comparing parameters, it is possible to find the hyper-parameter which minimizes the objective function and to obtain the best accuracy of parameters tested. Table 4 depicts values for a cross-validation set to illustrate the optimization of hyper-parameters.

D	Average training accuracy	Average validation accuracy
1	0.880499	0.880499
10	0.885924	0.878835
100	0.926635	0.874435
1000	0.926159	0.873306

Table 4. Validation set for Bagging Decision Trees with 80/20 split

Table 4 demonstrates why it is important to carefully choose a parameter based on the validation set. For instance, although a tree depth of 100 had the best average training accuracy, it did not correlate to having the best average validation accuracy as well. The optimal tree depth to choose then is 1 because it gives the best validation accuracy. It is not without warrant, however, to think about other factors that may adversely impact the model of the data.

Under-fitting and over-fitting the data can play a major role in selecting an optimal parameter. In Table 5, the validation accuracy is much lower than the training accuracy which suggests that the model over-fit the data. This means that it was too specific in formulating a classification boundary for the training set. When the validation set is tested, it does poorly because it is not general enough to consider additional points. The most likely cause is the chosen d value of 1000. With increasing tree depth, models gain specificity but lose generality, leading to over-fitting of the data.

D	Training	Validation
1000	0.915123	0.729578

Table 5. BAG-DT for tree depth 1000 on 80/20 split

Now some comparisons can be made to the empirical study by Caruana and Niculescu-Mizil. Of the common datasets classified, LETTER P.2 (A-M as positive) and HS (IndianPine92) were used. Table 6 compares the accuracies obtained in both this paper and theirs.

Algorithm	LETTER P.2 (Caruana et al)	LETTER P.2 (this paper)	HS (Caruana et al)	HS (this paper)
BAG-DT	0.911	0.886	0.856	0.888
KNN	0.936	0.755	0.801	0.880
ANN	0.901	0.835	0.932	0.875

Table 6. BAG-DT comparison: 20/80 split used

Since no calibration was done in this study, values will be compared to non-calibrated models. Some results are similar but most of the accuracies obtained here are below the ones in the paper.

The supervised learning models used in their paper may be must more robust in terms of both algorithm and normalization. Of course, the results obtained vary every instance the program is run, so future iterations may provide either lower or greater accuracies, depending on the shuffles made to the training and test sets.

4. Conclusion

Overall, it is indicative that specific supervised learning algorithms will having varying performance on the dataset involved. Compared to Caruana and Niculescu-Mizil's paper, the accuracies obtained were not far off. Better implementation of cross-validation and selection of parameters can improve results. While these results are not comprehensive, they illustrate the usefulness of defining a relevant objective function and setting optimal parameters.

5. References

Caruana, Rich, and Alexandru Niculescu-Mizil. "An Empirical Comparison of Supervised Learning Algorithms." *Proceedings of the 23rd International Conference on Machine Learning* (2006): 161-68. Print.

6. Dataset Sources

1. https://archive.ics.uci.edu/ml/datasets/letter+recognition

2. http://www.ehu.eus/ccwintco/index.php?title=Hyperspectral_Remote_Sensing_Scenes

3. https://archive.ics.uci.edu/ml/datasets/Yeast