Comparison of Supervised Learning Methods

Ravi Patel

Abstract

There are a number of supervised learning methods out there, with each method having its own sets of optimizable parameters. This study compares a handful of these algorithm by optimizing its parameters in an attempt to rank these supervised learning methods.

1. Introduction

Empirical comparison of supervised learning algorithm has been conducted previously (Caruana and Niculescu-Mizil, 2006). This study performs a similar, but on a smaller scale examination for the validity of past study (Caruana and Niculescu-Mizil, 2006). The results are not as quiet as what is expected.

In terms of supervised learning method, there are seven main algorithm that are tested: Decision Tree, K-Nearest Neighbor (KNN), Support Vector Machine (SVM), Logistic Regression, Random Forest, Boosting with Decision Tree, Multilayer Perceptron Artificial Neural Network (ANN) model.

2. Methodology

2.1. Performance Metric:

The only performance metric used in ranking each algorithm is the classification error. However, optimization is done with 5-fold cross validation to ensure the parameters for each algorithm is the best as it can be for the given datasets. In the end, parameters that provides the smallest validation errors are chosen. Furthermore, to ensure greater accuracy and generalization of the classifier, each algorithm is tested first with 80% training size and 20% testing size, and then with 60% training size and 40% testing size.

2.2. Learning Algorithm

In running these algorithm, an attempt to fine tune some of the parameters has been made, while making sure the computation is not intractable. Here is a summary of how the parameters for each algorithm were optimized:

Decision Tree (DT): optimization is done in deciding the depth of the tree. Initially, tree depth starts with unity height and iteratively increase the height until all the nodes are expanded until all leaves are pure, where at each iteration height increments by 100. After the initial run, a suboptimal depth, h_{sub}, is attained. Unless h_{sub} is 1, another iteration starting at $(h_{sub} - 100)$ and ending at $(h_{sub} + 100)$ is done to optimize the depth of the decision tree, with a change of 10 at each iteration.

KNN: optimization is done in deciding the number of nearest neighbors, with the standard Euclidean 2-norm distance metric. The optimal number of nearest neighbors is chosen in the same fashion as the depth of the decision tree.

SVM: both the linear and the Gaussian (RBF) kernel is tested with SVM. Optimization on the penalty parameter is done for both kernel. The following penalty parameter ranges from 0.01 to 10, incrementing by 0.49.

Logistic Regression (LOGREG): regularized logistic regression while varying the rigid (regularization) parameter by taking 200 evenly spaced points starting from 0.01 (inclusive) to 10 (exclusive).

Random Forest (RF): optimization on the number of decision trees and on the depth of any such collection of decision trees is done. Both the depth and the number of trees ranges from 1 (inclusive) to 100 (inclusive).

Adaboost (ADB): boosting with decision tree classifier optimizes the number of trees and the learning rates. Number of decision trees ranges from 1 to 100. Learning rate is tested with 10 evenly spaced samples from 0.01 to 5.

ANN: multi-layer perceptron (MIP) with stochastic gradient descent is optimized while varying the number of layers from taking 25 evenly spaced samples from 1 to 200, and 20 evenly spaced samples from 0 to 1 for momentum.

3. Experiment

The following three datasets were being used: Tic-Tac-Toe Endgame and Car Evaluation datasets from the UCI repository and the handwritten digits dataset from the MNIST database.

Important to note is that the Car Evaluation and handwritten digits datasets creates multi-class classification problem. However, for the purposes of this study, these multi-class classification is turned into binary classification problem.

The Tic-Tac-Toe Endgame dataset concerns with whether the first player has any chance of winning, given the current condition of the game. Handwritten digit dataset concerns with classifying the digits 1 or not 1, and Car Evaluation dataset concerns with classifying a car as acceptable or not acceptable.

3.1. Data Cleaning

In order to perform quantitative analysis on categorical features, the method of One-hot Encoding is utilized. All of the features of Tic-Tac-Toe Endgame dataset and Car Evaluation dataset are encoded with One-hot Encoding.

3.2 Results

After performing the 5-fold cross validation and running the algorithm with the optimal parameters, the results are summarized in the following tables:

Table 1: Classification Error - 80% Training Size, 20% Testing Size

Models	Tic-Tac-Toe	Handwritten Digits	Car Evaluation
DT	1.56%	1.4%	1.15%
KNN	16.15%	0.80%	4.04%
SVM - linear kernel	3.13%	1.20%	3.76%
SVM - rbf kernel	3.13%	0.40%	0.58%
LOGREG	3.13%	0.8%	4.33%
RF	4.20%	1.20%	1.16%
ADB	3.13%	0.80%	4.91%

MLP	3.13%	0.40%	0.58%

Table 2: Classification Error: 60% Training Size, 40% Testing Size

Models	Tic-Tac-Toe	Handwritten Digits	Car Evaluation
DT	6.25%	1.4%	1.16%
KNN	19.79%	0.80%	16.76%
SVM - linear kernel	2.83%	1.20%	5.90%
SVM - rbf kernel	2.83%	0.40%	0.29%
LOGREG	2.83%	0.8%	3.90%
RF	3.125%	1.20%	1.17%
ADB	2..87%	0.80%	4.92%
MLP	2.83%	0.40%	0.24%

Table 1 show the classification error of each of the algorithm when the training size is 80% and testing size is 20% of the entire dataset. The first column indicates the algorithm being perform. The second column indicates the classification error of the algorithm in the first column for the Tic-Tac-Toe dataset. The third column indicates the classification error for the Handwritten Digits datasets, and the last column indicates the last column is the classification error for the Car Evaluation Datasets. Table 2 show the same information with training size being 60% and testing size being 40% of the entire dataset.

3.3. Analysis

Looking at the overall performance of the algorithm, MLP outshines every other models in terms of classification by ranking first (or tied to first) five out of the six times MLP is implemented. In the second place comes SVM with Gaussian kernel by ranking first (or tied to first) four out of the six times the algorithm is implemented. Table 3 summarizes this ranking.

Table 3: Ranking Place of Algorithm

Models	1st	2nd	3rd	4th	5th	6th	7th	8th
DT	1/6	2/6	0	4/6	0	0	0	0
KNN	0	0	2/6	1/6	2/6	0	0	1/6
SVM - linear kernel	1/6	1/6	2/6	1/6	0	0	1/6	0

SVM - rbf kernel	4/6	2/6	0	0	0	0	0	0
LOGREG	0	1/6	3/6	0	1/6	1/6	0	0
RF	0	0	6/6	0	0	0	0	0
ADB	1/6	2/6	0	0	0	0	1/6	1/6
MLP	5/6	1/6	0	0	0	0	0	0

The first column of Table 3 corresponds to algorithm of interest, while the (i^{th} + *1)* column represent the number to time the algorithm came in i^{th} place, $1 < i < 9$.

Referring back to study conducted by Caruana and Niculescu-Mizil, these results are not as quiet as what is expected (Caruana and Niculescu-Mizil, 2006). Caruana and Niculescu-Mizil were able to show some form of boosting algorithm with decision tree as coming first majority of the time. However, in this study neural networks surpass any other algorithm by by a fairly good margin while SVM with Gaussian kernel coming in second place.

4. Conclusion

It is possible that the reason for not attaining similar results as Caruana and Niculescu-Mizil is due to the datasets being used or due to errors in optimization. Nevertheless, one thing stands clear, no one single algorithm is best for every scenario.

Reference

Rice Caruana, Alexandru Niculescu-Mizil (2006), *An Empirical Comparison of Supervised Learning Algorithm.* ICML

Decision Tree Classifier.
http://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeClassifier.html

K Nearest Neighbor Classifier.
http://scikit-learn.org/stable/modules/generated/sklearn.neighbors.KNeighborsClassifier.html#sklearn.neighbors.KNeighborsClassifier

Support Vector Machine Classifier.
http://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html

Logistic Regression Classifier.
http://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html

Random Forest Classifier.
http://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html

Adaboost Classifier.
http://scikit-learn.org/stable/modules/generated/sklearn.ensemble.AdaBoostClassifier.html

Multilayer Perceptron Classifier.
http://scikit-learn.org/stable/modules/generated/sklearn.neural_network.MLPClassifier.html#sklearn.neural_network.MLPClassifier

One-hot Encoding.
http://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.OneHotEncoder.html

Tic-Tac-Toc Dataset.
https://archive.ics.uci.edu/ml/datasets/Tic-Tac-Toe+Endgame

Car Evaluation Dataset.
http://archive.ics.uci.edu/ml/datasets/Car+Evaluation

Handwritten Digit Dataset.
http://yann.lecun.com/exdb/mnist/