Project 1

Arnor Ingi Sigurdsson[1]
*Hong Kong University of Science and Technology, Hong Kong

Introduction

This project revolved around training machine learning classifiers on a training set and making predictions on a test set. Three files containing training data, training labels and test data were supplied and used for the training and performance evaluation of chosen classifiers. Five different machine learning classifiers were chosen, those being logistic regression, support vector machine (SVM), K nearest neighbors, random forest and naive Bayes. The models were evaluated using the training data with nested cross validation, containing 5 outer folds and 5 inner folds. After choosing the best hyperparameter combination for each model, the models were each trained on the whole training data and used to make predictions on the test data. For the predictions, the five models were treated as a single ensemble classifier, where a majority mode decided how to classify each test observation.

Methods

Data

The dataset provided is contains 57 features and is split into training and test data. The training data is composed of 3220 observations and their labels. The test data is composed of 1380 non-labeled data. The scale of the data is considerably varied, with some columns containing values close to 0 while others contain values up to 9088.

Preprocessing

When using an Euclidian distance measure, some classifiers (such as SVM) can be sensitive do the data being on wildly different scales. Therefore it can be a good approach to standardize or normalize the data before training machine learning models. As mentioned, the scale of the data used in this project varied by a large margin depending on which feature was being looked at. Therefore, the training data was standardized prior to training. The calculated mean and standard deviation were used to standardize the test data before making predictions.

Classifiers

As mentioned before, a total of five classifiers were trained and testing on the training and test data respectively.

Logistic Regression is a linear probability model used to predict the probability a given observation belongs to a certain class. The function it uses takes the form of $\frac{1}{1+e^{-t}}$. Assuming that t is expressed as a linear model, one can use ordinary least squares to estimate the coefficients that best fit the data. Weighting input with these coefficients and feeding them to the function above, one can interpret the result as the probability of $y = 1$ given that the input.

Support Vector Machines try to find the optimal hyperplane to separate data of certain classes. While many machine learning models find a decision boundary that separates data points of different classes from each other (such as logistic regression), SVM differ in a way that they try to find the best decision boundary that separates observations.

They do this by looking at observations in the training data belonging to different classes that are close to each other (i.e. "difficult observations", also called "support vectors"). After identifying these difficult to classify observations, the SVM attempts to find a decision boundary that maximizes its distance to these difficult observations.

Manuscript compiled: Thursday 16th November, 2017
[1]20483569, aisigurdsson@connect.ust.hk

K Nearest Neighbors is a relatively intuitive algorithm that works by looking at the distance (e.g. Euclidian distance) between data points and assigning a new point to the same class as the point(s) it is closest to. It has been called a "lazy" learning algorithm as there is not parameter tuning involved during training, but rather simply assigning new points to their closest neighbors.

Random Forest works by creating an ensemble of decision trees that each get trained on a subset (sampled with replacement, akin to bootstrapping) of the total data. At each node of each decision tree, m features are selected at random and they used to split the node samples as well as possible (measured using an impurity measure).

Naive Bayes works by applying Bayes' theorem with the assumption that features are independent. In this work a Gaussian implementation of naive Bayes is used. This implementation assumes that the continuous values associated with each class are normal distributed. Given that we have two classes, the probability of observing a given observation from the normal distributions constructed from the classes' feature values are calculated. The observation then gets assigned to the class from which it is more likely to have been created from.

Performance evaluation

A 5 fold nested cross-validation was used to estimate the generalization error and to tune the hyperparameters of each model[1]. Hyperparameter combinations were chosen with grid search, and the performance of each combination was measured in a 5 fold cross-validation in a given inner fold. In each inner fold, the hyperparameter combination that on average gave the lowest test error was chosen as the "best version of that model". This version was then trained on the whole inner fold and its generalization error estimated on the outer loop. This process was repeated iteratively for each outer fold split, therefore a given model's generalization error is estimated as the average over all outer folds.

It is important to note that different hyperparameter combinations can be chosen in each inner fold as the grid search is performed independently for each outer split. In order to settle on what hyperparameter combination to use, the combination that gave the lowest inner fold test error most often was used (i.e. m. The 2 fold nested cross-validation can be visualized in figure 1.

Test predictions

After choosing the best hyperparameter combination for each model following the nested cross validation, each model was trained on the whole training data set. Then, all the models were used to make predictions on the previously unseen test set. The five models were treated as an ensemble, and the final prediction for each test observation was decided by a majority vote. That is, if three or more of the models classified an observation as belonging to class 1 it was assigned to class 1 and vice versa for class 0. The final predictions were saved and submitted in the project hand-in.

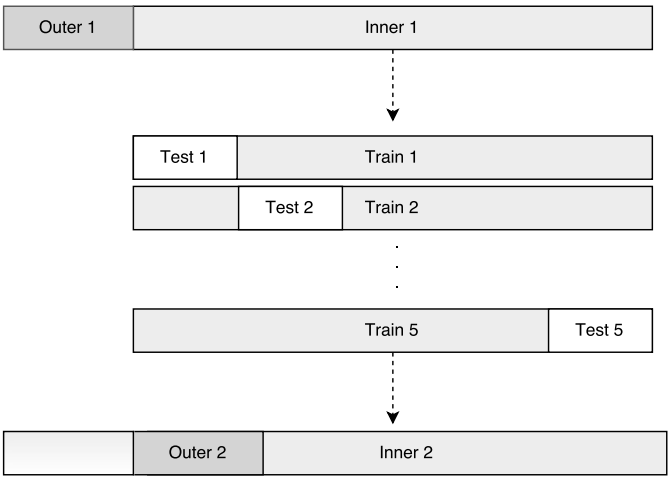

Figure 1 A diagram showing the cross-validation scheme used in the experiment. Five outer folds were chosen for the estimation of the generalization error. In each inner fold, a 5 fold cross-validation was done for model training and testing in order to choose the best model in each outer split.

Results

The inner and outer fold accuracies from the nested cross validation can be seen in table 1. As can be seen from table 1, random forest and seems to be the best performing model, with its standard deviation not overlapping with any other model. This gives a hint that random forest outperforms the other models, but a more rigorous statistical analysis is required to make any conclusive assertions.

The high accuracy results indicate that hopefully the predictions made on the test set will be relatively accurate as well. However, only after comparing the predicted labels to the real ones (currently unknown), can a measure of that be made.

[1] Although when using random forest it is not necessary to use cross-validation as out-of-bag error estimate can be used, all models were evaluated with cross-validation for consistency.

Table 1 Model Performance. Performance of the trained models on the outer and inner folds in the nested cross validation used. The chosen hyperparameter combinations for each model is also shown.

Model	Mean Inner Fold Accuracy	Mean Outer Fold Accuracy	Chosen Hyperparameters[a]
Logistic Regression	0.92 ± 0.0034	0.92 ± 0.014	C: 500,
Support Vector Machine	0.93 ± 0.0029	0.93 ± 0.011	C: 1, kernel: rbf
K Nearest Neighbors	0.90 ± 0.0047	0.91 ± 0.010	n_neighbors: 5, p: 1
Random Forest	0.95 ± 0.0023	0.95 ± 0.0065	n_estimators: 500, criterion: entropy
Naive Bayes	0.83 ± 0.0043	0.83 ± 0.010	n/a

[a] Excluding the hyperparameters chosen for tuning, other hyperparameters were set to their defaults in scikit-learn version 0.19.1.