An in-Depth Empirical Study of Supervised Learning Classification

Jia Shi
Unversity of California, San Diego
jis283@ucsd.edu

March 22, 2020

Abstract

Throughout the past few decades, various of Machine Learning methods had been developed. With the introduction of Artificial intelligence, learning had became an important approach to heuristically mimic the human learner by correcting from mistakes. From the early introduction of Perceptron in 1957, methods like KNN, SVM, Random Forest and Boosting had been introduced. In this paper, we will present an empirical comparison between six representative supervised learning methods and their performance in different fields. We will test K-Nearest Neighbors, Random Forest, Gradient Boosting, Multi-layer perceptron, Decision Tree and Linear Support Vector Machine and their performance in basis machine learning, computer vision and natural language processing tasks.

1 Introduction

In this study, we had choose six different classifiers to perform empirical comparison. And each of the classifiers is representing different kinds of learning method. We had choose Linear Support Vector Machine as the representation of linear classier, instead of the Logistic Regression, and had choice K-Nearest Neighbors because it's a non-parametric classification method. Also, we also examined two special structure model: tree-base model decision tree which make decision on each of the variable separately and non-linear basic neural network model Multi-layer perceptron. Finally, we had examined two ways of ensemble learning, Random Forest of Bagging method and Gradient Boosting of Boosting method.

2 Datasets and Methods

For the six representative classifier we had mentioned above, we had examined them in several different datasets. We had also applied various data-preprocessing methods on each datasets to amplify the scale of the study.

2.1 Datasets

First datasets is a customer churn prediction study in telecommunication field, which use information like user's daily phone call time, texts number in Morning and night to predict whether the customer will subscribe to our companies next month. It have 5000 data in the dataset. This is a binary prediction, and the label is true/false. We had made comparison study on whether categorical variables was being used.

Second datasets is Mnist, the 'hello world' dataset in computer vision. It include 4000 hand-written digits of 784 pixel grayscale and 1-d as the target. We will train on the grayscale values of each pixels of the 28*28 pictures and predict the actual value of hand-written digits from 0-9. For this datasets, we will transform it into several one-one classification and takes the average accuracy as target. For example, we will first compare value of 0, vs value that is not 0, next on value of 1 and not 1, etc. We will also applied dimension reduction method like PCA and projection to amplify the scale of the study.

Third datasets is Spam classification in NLP. There's 580 data in this dataset. We need to perform training on 4003 vectorized words of each email. There's a 4003 word dictionary, and for each word in the dictionary, we mark 1 if this word shown in the email, and 0 otherwise. This is

Six Classifiers	
Classifiers	Features
KNN	non-parametric model
SVM	linear model
MLP	non-linear model
Decision Tree	tree base model
Random Forest	ensemble bagging
Gradient boosting	ensemble boosting

Figure 1: The six classifiers in this study and their unique feature.

also an binary prediction of whether the email is Spam or not.

2.2 Methods

For each of datasets, we will perform 80/20, 50/50, 20/80 train-test data split, and grid search for some hyper-parameters search of each classier. We will apply 3-fold cross validation for each of the search parameters and repeat this process by 3 times in order to prevents randomness. We will compare the performance for each of the classifiers for each of the section.
Just to clarify, 80/20 split stands for 80% training data and 20% of testing data.

2.3 Classifers

For each classier, we perform grid search with 3-fold cross validation on different hyper- parameters

KNN: the prediction result accordingly to the labels of closest neighbors, we had choose number of neighbor=[1,3,5,7,9]. If the k is too large, it can overfit the data distribution of the very specific dataset

SVM: we need to maximum the distance between different classes, that is, the margin distance between support vectors. We will try on C=[0.001,0.01,0.1,1,10]

Decision Tree: multi-layer decision stumps that establish a tree-shape graph that direct the data points into different directions base on each variable, we will try on criterion=[gini, entropy], max-features:[3,5,7,9,11,13,15],max-depth:[10,20...100]

Random Forest: this is an example of ensemble bagging classier with multiple decision tree. Compared to regular decision tree, it introduce more randomness to prevent overfitting. We will try hyper-parameter of criterion=[gini, entropy], max features of [‘auto’,’log2’,’None’] and max depth of the tree= from 10 to 90 with gap of 10

Gradient Boosting: another example of ensemble learning method which combine the advantage of multiple weak classifiers to train the best ‘leader’,we will try to examine on, learning rate of each update=[0.001,0.01,0.1], number of estimators=[10,50,100,200], max features to consider:[‘auto’,’log2’,None]

MLP: it’s one of the most basic form of neural networks, which combine multiple linear perception in each layer with activation function to form a non-linear classier. We will try number of hidden layer and neural nets number of of [(100,),(500,)(500,500,500)] , learning rate decay method=[”constant”, ”invscaling”, “adaptive”], activation function =[‘identity’, ‘logistic’, ‘tanh’, ‘relu’], and learningrate of [0.001, 0.01, 0.1] Constant: stay at constant of default 0.001 invscaling:gradually decreases the learning rate at each time step ‘t’ using an inverse scaling exponent adaptive: keeps the learning rate constant to ‘learningrateinit’ as long as training loss keeps decreasing.

3 Experiments

3.1 Churn Dataset

For the dataset1(churn dataset), we had applied comparison study in two way.

1. Train our classier without categorical variables
2. Train our class with categorical variables (with

Churn Dataset				
Classifier	Without categorical		With categorical	
	Train/Test Accuracy		Train/Test Accuracy	
KNN (80/20)	90.6%	88.4%	74.6%	72.6%
KNN (50/50)	90.7%	88.8%	76.8%	74.2%
KNN (20/80)	89.6%	87.9%	75.4%	72.5%
SVM (80/20)	85.6%	86.2%	75.2%	76.2%
SVM (50/50)	85.6%	86.2%	74.6%	75.6%
SVM (20/80)	87.6%	85.5%	74.4%	74.2%
Decision Tree (80/20)	98.5%	93.8%	100.0%	99.9%
Decision Tree (50/50)	98.7%	93.8%	100.0%	99.9%
Decision Tree (20/80)	98.4%	91.3%	100.0%	99.9%
Random Forest (80/20)	99.4%	95.0%	100.0%	99.8%
Random Forest (50/50)	98.2%	94.8%	100.0%	99.6%
Random Forest (20/80)	99.2%	90.3%	100.0%	98.5%
Gradient Boosting (80/20)	98.5%	95.0%	100.0%	99.9%
Gradient Boosting (50/50)	98.4%	94.9%	100.0%	99.8%
Gradient Boosting (20/80))	99.9%	93.3%	100.0%	99.7%
MLP (80/20)	87.7%	86.6%	100.0%	99.9%
MLP (50/50)	85.7%	87.3%	100.0%	99.8%
MLP (20/80)	89.2%	85.9%	100.0%	95.5%

Figure 2: Study of the Churn Dataset

one-hot encoding)
For this dataset, in order to predict whether a customer would keep subscribe to the same telecommunication company(for example, Att), we may only train our classifier with numeric variable like total day minutes and total eve minutes. But we can also decide whether to use categorical variable like, which state does the customer came from. We will compare the performance of six classifiers in those two setting.

Informative observation of hyper-parameters:

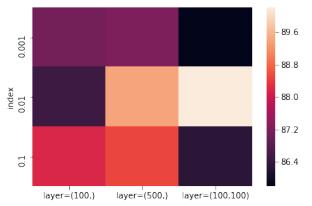

Figure 3: An heatmap of gird search of MLP on no categorical data on hidden layer size and learning rate

The best parameters of 80/20 split:
KNN: k=7
DT:max-treedepth=10,criterion=entropy,max-feature=15
GB:learning rate=1,clf-max-depth=3,max-feature=None, estimator=100
RF:criterion=gini,max-depth=10,max-feature=None
SVM:C=10
MLP:learning rate=0.01,decay=constant,acti-function=relu,hidden-layer=(100,100)
KNN: with 80/20 split, the best model choose 7 neighbors, while both 50/50 and 20/80 split choose 9 neighbors. That is because the classier would want to have more neighbor points as reference when the number of training data is not enough to well fit the data distribution.
Decision tree: Both 80/20 and 50/50 split choose max-tree depth of 10, while 20/80 split choose 90. Normally, the deeper the tree is, the better the performance is. However, 80/20 split and 50/50 split choose to stick with max depth of 10. Because for decision tree, it's very easy to overfit. In order to decrease the testing error, they choose not to extend the tree for too deep.

Overall observation:
In general, ensemble methods like random forest and gradient boosting perform the best among the six classifiers, and linear classier of SVM and shallow neural nets method of MLP perform the worse. That is because the dataset is not easily linearly separable because of the combination of multiple independent variables. Thus, the non-linear classifiers like decision tree, which take consideration of variables one by one will reach decent score which is close to ensemble learning method.
What is more, an interesting observation is that, after applying categorical variables, some classifiers of 80/20 split except for KNN and SVM reach almost perfect testing accuracy of 99.9%. The reason why KNN perform badly with extra information is it's very non-robust in handling noise. After one-hot encoding, the number of variables change from 17 to 72. Since KNN try to find data points that is most similar to the testing point, the intro of extra information may interfere with the decision making process. For SVM, the intro of extra variables even increase the dimensions of data to a very high number, and 50 of them are actually the information of the one hot coding of a single variable, 'state', which make some of the variables contain less information. Since we didn't try to study on L1 penalty which may help on features selection, what we have right not would only make linear classification harder. That is why the accuracy of KNN and SVM would drop after putting in more categorical information.
80/20 perform the best among the three data split. Since we would assume all data came from the same data generating process and have the same distribution, we can better fit the data distribution with more data. As a result, the result that 80/20 split perform the best is very intuitive.
However, because each of the split may choose different hyper-parameters as the best model, their accuracy may vary from time to time.
All the training accuracy is higher than the testing accuracy, and stronger classier would have 1.0 training accuracy with categorical variables.

Mnist Dataset				
Classifier	784-d		20-d with PCA	
	Train/Test Accuracy		Train/Test Accuracy	
KNN (80/20)	85.0 %	93.8%	79.9%	74.0%
KNN (50/50)	100.0 %	91.3%	77.1%	70.9%
KNN (20/80)	100.0 %	87.1%	1.0%	63.3%
SVM (80/20)	74.4 %	8.5%	10.3%	8.5%
SVM (50/50)	10.8 %	9.3%	10.1%	9.5%
SVM (20/80)	10.11 %	9.5%	9.7%	9.3%
Decision Tree (80/20)	89.2 %	74.3%	100.0%	47.8%
Decision Tree (50/50)	100.0 %	70.9%	100.0%	43.3%
Decision Tree (20/80)	100.0 %	65.5%	100.0%	42.2%
Random Forest (80/20)	89.0 %	85.6%	99.6%	59.9%
Random Forest (50/50)	100.0 %	84.1%	97.0%	59.3%
Random Forest (20/80)	98.8 %	80.0%	99.0%	53.0%
Gradient Boosting (80/20)	83.9 %	90.8%	96.1%	68.4%
Gradient Boosting (50/50)	100.0 %	89.1%	99.2%	65.1%
Gradient Boosting (20/80))	100.0 %	84.7%	1.0%	59.2%
MLP (80/20)	88.2 %	90.8%	100.0%	76.0%
MLP (50/50)	100.0 %	90.4%	100.0%	75.0%
MLP (20/80)	99.5 %	84.8%	100.0%	68.8%

Figure 4: Study of Mnist dataset.

3.2 Mnist Dataset

For the dataset2(Mnist dataset), we had applied comparison study in two way.

1. Standard Mnist dataset with variables of the grayscale of each pixels (784)

2. Mnist dataset with variables being from project from 784-d to 20-d with PCA

This is an experiment of apply traditional machine learning methods into novel application like computer vision. We had transform the 784-d variables and applied principle component analysis to project the data into a lower dimension with the selection of its Eigenvectors.
It's worth to notice that this is a multi-class classification, and we decide not to make it into one-many classification in order to test the real accuracy in fitting data of multiple labels.

Informative observation of hyper-parameters:
The best parameters of 80/20 split:
KNN: k=1
DT:max-treedepth=90,criterion=entropy,max-feature=15
GB:learning rate=1,clf-max-depth=3,max-feature=None, estimator=500
RF:criterion=gini,max-depth=90,max-feature=None
SVM:C=10
MLP:decay=constant,activation function=relu,hidden layer=(500,)
For 80/20 and 50/50 split of KNN, they had choose nearest 1 neighbor as best model, while 20/80 choose 3. And after projection, 80/20 and 50/50 had choose 7 and 9 as value of k. That is because in a very high dimension, the distribution of data variable will be very noisy. Thus, data points would only stick to a limit number of neighbor to reduce noise and make better prediction. While the dimension is lower, each dimension will contain more information and less noisy, thus it will choose more k as reference to make prediction.

Observation of performance:
80/20 perform the best among three split, that is because it have trained more data points to fit the data distribution. Overall, KNN perform the best in both 784-d data and 20-d data, while gradient boosting place second in testing accuracy. SVM reach only less than 10% accuracy which is basically random guessing because there's 10 different labels, which means pure linear classier is not working at all in this high dimension dataset. Decision tree would alway have a good ranking in regular machine learning tasks, however, it's the second to the worst in this task. That is because digit classification rely on the relationship between grayscale of different pixels, but not each of them. Decision tree would only build node for each of the variable, which deprive the relationship between pixels. Similar reason for the poor performance of random forest as well. That is also why KNN will perform so good because it keep the distribution of each variables. For MLP, the performance is relatively good because the activation function in each hidden layer had transform it into a non-linear classifier, which will perform better compare to other classification methods. It also chose hyper-parameter with only 1 hidden layer but 500 nodes, which make it complex enough to fit the data.
After applying projection, all of the classifiers perform worse compare to before. That is because projection have lost so information. However, it increase the training and testing speed tremendously because of the decrement of variables. About training and testing accuracy. One of the intriguing point to notice is that, for 784-d data, the 50/50 and 20/80 split would accuracy have training accuracy of close to 100, which is very much overfitting(except for SVM, it's not fitting at all). One of my hypothesis is that the classier would be more easy to overfit small training dataset(like with 50/50 and 20/80 split). The exact reason would need to be further researched.

3.3 Spam Dataset

For dataset3(spam dataset), we apply six classifiers on this dataset directly without any comparison condition.
This is an experiment that applied supervised learning method into basic NLP mission that determine whether a email is a spam with vectorized word lists. Each emails was vectored into 4003-d one hot encoding data, with binary label of T/F representing whether it's a spam email or not.

Informative observation of hyper-parameters:
KNN: k=7
DT:max-treedepth=40,criterion=entropy,max-feature=12
GB:learning rate=1,clf-max-depth=3,max-feature=None, estimator=500

Spam Dataset		
Classifier	Train/Test Accuracy	
KNN (80/20)	94.1 %	72.9%
KNN (50/50)	99.3 %	72.8%
KNN (20/80)	100.0 %	73.2%
SVM (80/20)	84.6%	83.9%
SVM (50/50)	83.3 %	83.5%
SVM (20/80)	85.2 %	83.2%
Decision Tree (80/20)	100.0 %	93.9%
Decision Tree (50/50)	100.0 %	93.3%
Decision Tree (20/80)	100.0 %	89.2%
Random Forest (80/20)	100.0 %	97.4%
Random Forest (50/50)	100.0 %	98.6%
Random Forest (20/80)	100.0 %	93.8%
Gradient Boosting (80/20)	100.0 %	98.3%
Gradient Boosting (50/50)	100.0 %	96.3%
Gradient Boosting (20/80))	100.0 %	94.2%
MLP (80/20)	100.0 %	100.0%
MLP (50/50)	100.0 %	99.6%
MLP (20/80)	100.0 %	98.6%

Figure 5: Study of Spam Dataset.

RF:criterion=gini,max-depth=60,max-feature=None
SVM:C=1
MLP:decay=constant,activation function=relu,hidden layer=(100,100)
All MLP have stick to max number of hidden layer of 2. One hypothesis is that the data points with same labels would all have similar distribution of variables, but not any single of them isolated. Thus our classier need to refer to more information in order to store the distribution information of data points.

Observation of performance:
In general, Decision Tree, Random Forest, Gradient Boosting and MLP have all reach training accuracy of 100%. And linear SVM perform the worst, which means that this dataset is not easily linearly separable. And the fact the MLP perform the best means that data with same label are well clustering together and can be separate by a non-linear decision boundary. Decision Tree perform relatively poor because, similar to Mnist dataset, the target was determined by the distribution of variables, but not single one of them. Since decision tree only process a single variable at a time, it's not so well-fit compare to others.
In general, 80/20 split perform better than 2 other splits, and training accuracy is higher than testing accuracy. Due to randomness, some data may not well-fit those two rules all the time.

4 Conclusion

Throughout our in-depth empirical studies that compared 6 very representative classifiers with 3 dataset from very different fields, we had reach a conclusion that is very similar to Caruana's paper, that ensemble method like random forest and gradient boosting would perform very stable with very competitive accuracy compare with others, and decision tree could be very easily overfit our training data, while perform not so good on the testing data. Also, when the prediction label have a strong relation with the distribution of data variables, not any single one of them,

Decision tree would not preform so well, because it only consider one variable at a time.
MLP is a very competitive classifier that can well-fit many non-linear dataset, while linear SVM do not. KNN would well fit the data when there's less noise, and it's very sensitive to noisy data. In general, Gradient Boosting and MLP spending the longest time in training, and KNN take the longest time in testing.
Throughout this study, we had notice the features of different classifiers, and each of them will perform well in different data setting. Thus, data distribution is essential to consider before performing supervised learning.

5 Reference

Caruana, R., Niculescu-Mizil, A. (2006). An emperical emparison of supervied learning algorithms. *Proceedings of the 23rd International Conference on Machine Learing - ICML 06.* 10.1145/1143844.1143865

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.

Scikit-learn documentation *http://scikit-learn.ory/stable*

6 Bonus Points

In this study, I had made comparison studies on 6 different classifiers, and 3 datasets with various setting. For example, for Churn dataset, I had applied study on both with categorical variables and without. For Mnist setting, I had applied PCA to make comparison study of 784-d and 20-d data.
I had listed the unique features of those six classfiers and why I chose them, which can be considered very intuitive for this study.
Also, I had applied machine learning methods on datasets from three different fields,ML,CV and NLP, which can be considered a new and insightful comparison studies.
I had used Latex to write this paper, which is also time-consuming.