Applied Machine Learning

Naïve Bayes and Decision Tree for Digit Recognition

Now that we have learned two classification algorithms, Decision Tree and Naïve Bayes, let's think further on the question of choosing algorithms for a specific task. Note that there is no silver bullet in terms of algorithm comparison – no algorithm would outperform all other algorithms on all data sets. Therefore, choosing appropriate algorithms is an important decision, and it requires knowledge of both the data set and the candidate algorithms. In this homework, you will compare naïve Bayes and decision tree for handwriting recognition.

Task description: The data set comes from the Kaggle Digit Recognizer competition. The goal is to recognize digits 0 to 9 in handwriting images. Because the original data set is too large, I have systematically sampled 5% each of the original training and testing data, stored in files called *digit-train.csv* and *digit-test.csv*, respectively. You are going to use the training data to construct prediction models using both naïve Bayes and decision tree algorithms. Tune their parameters to get the best model (measured by cross validation) and compare which algorithms provide better performance on the testing data.

Report structure:

Section 1: Introduction

Briefly describe the classification problem and general data preprocessing. Note that some data preprocessing steps maybe specific to a particular algorithm. Report those steps under each algorithm section.

Section 2: Decision Tree

Build a Decision Tree model. Tune the parameters and report the 5-fold CV accuracy.

Section 3: Naïve Bayes

Build a Naïve Bayes model. Tune the parameters, such as the smoothing alpha, to compare results.

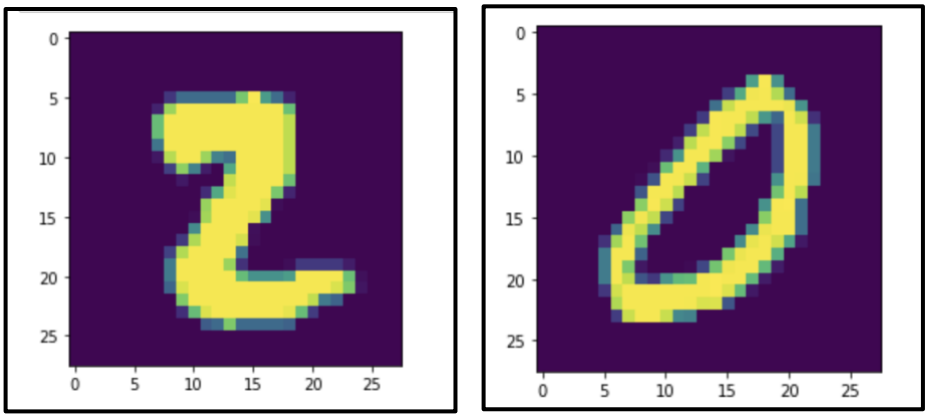
Section 4: Algorithm performance comparison

Compare the results from the two algorithms. Which one reached higher accuracy? Which one runs faster? Can you explain why?

Introduction

We are working on data set that contains the value of digits ranging from 0 to 9 in handwriting. It is also referred as **MNIST** data that stands for **Modified National Institute of Standards and Technology** data set. My task is to classify a given image of handwritten data into one of the 10 classes representing integer values from 0 to 9, inclusively. The image size is 28*28 pixels. The data is given in two separate .CSV files i.e., train.csv and test.csv. We don't need to do any further train and test split. From the given train.csv file I created X_train with all my independent features and y_train containing my dependent feature the exact same process was carried out for X_test and y_test from test.csv file. We know for a fact that pixel values for each image in the data set are unsigned integers in the range between black and white, or 0 and 255. We will normalize the pixel values of greyscale images by dividing the values with 255 e.g. rescale them to range [0,1]. I will implement two models i.e., Decision Tree and Naïve Bayes. In Decision Tree we will observe the model accuracy before and after hyperparameter tuning. In Naïve bayes I will use different smoothing parameters to obtain the best model.

Below are the images of **handwritten digit's 2 and 0**



Decision Tree Model

I constructed a decision tree model on the digit recognizer data to classify digits from 0 to 9. Without doing any hyperparameter tuning on the decision tree and by predicting on the test set it gave me an accuracy of about **100% on the train set** and **78% on the test set** which results in overfitting which is always the case with decision trees. In order to control overfitting we create a parameter grid and define the maximum depth, minimum samples at leaf node and the criterion. Maximum depth of the tree means how long we want the tree to be i.e., the number of nodes we want. Minimum Samples at leaf node means the number of samples required to leaf node. Hyperparameter tuning gives me the best model from the **5 fold Cross validation** i.e., running over 350 possibilities. The best model has **max depth of 10 and minimum samples at leaf node are 5**. **Train Accuracy of the model is 89.3%** and **Test Accuracy of the model is 76.8%**. **We get the decision tree after hyperparameter tuning but still model is overfitting.**

Naïve Bayes Model

Naïve Bayes model works on the concept of conditional probability. In which we find the probability of dependent event given the probability of an independent event. There are different variants of Naïve Byes Algorithm such as **Bernoulli, Multinomial and Gaussian Naïve Byes**. **Gaussian Naïve Byes** is used when we have features in the continuous form, features are distributed according to gaussian distribution which is also referred as Normal distribution. **Bernoulli Naïve Byes** is used when we have features that have

values in the binary form for e.g. Yes/No, True/False , **1/0** etc. Multinomial NB is used when we need to find the discrete count and frequency. It is actually more popular with Natural language processing. In digit recognizer data I checked my accuracy for Gaussian and Bernoulli NB. For **Gaussian NB** my model accuracy comes out to be **56.53% on train data** and **52.67%** on test data. As per our data set, we will implement **Bernoulli NB** because we have values in the range of 0 and 1 in our features and the model is not overfitting on the train data. For **Bernoulli NB** my model accuracy comes out to be **84.09% on train data** and **82.23%** on test data. We carry level smoothing in Naïve Bayes to avoid the problem of zero probability. Even after trying multiple level smoothing parameters there was no change in the model accuracy.

Algorithm performance comparison

After performing both the machine learning algorithms on the data set. Even though we get high accuracy on Decision tree algorithm but the model overfits when we observe the model accuracy on train and test i.e., **89.3%** on Train and **76.8%** on test data. In case of Bernoulli Naïve byes algorithm we get an accuracy of **84.09% on train data** and **82.23% on test data**. I printed the classification report for both the models to observe the **precision value**.

Decision Tree-Model Accuracy

Before Hyper parameter Tuning		After Hyper parameter Tuning		Model Status
Train	**Test**	**Train**	**Test**	
100%	78.3%	89.3%	76.8%	**Overfitting**

Naïve Bayes-Model Accuracy

Before Smoothing aplha		After Smoothing alpha-0.001		Model Status
Train	**Test**	**Train**	**Test**	
84.09%	82.23%	84.09%	82.23%	**Good**

For **Naive Bayes model** I get **precision** and **recall** value for each and every digit and it as follows:

Digit	Precision	Recall
0	0.92	0.91
1	0.89	0.93
2	0.86	0.85
3	0.74	0.76
4	0.78	0.78
5	0.79	0.75
6	0.91	0.90
7	0.93	0.87
8	0.80	0.78
9	0.67	0.73

In the above table **Precision** specifies **the no. of times the predicted value is same as actual value.** For **digit-7** we have the highest precision of 93%. Recall specifies out of all the actual results how many were actually right. Precision is the lowest for **digit-9** i.e., **67%**.