Özyeğin University

CS551 – Introduction to Artificial Intelligence

Assignment-3

Hand-written Digit Recognition

Furkan Cantürk

06.09.2021

1. INTRODUCTION

In this assignment, three supervised machine learning algorithms are deployed to classify handwritten digits, which are Naïve Bayes, Decision Tree, and K-Nearest Neighbor. Each algorithm is tuned by a Grid Search of its parameters over validation set, which is detailed in the section.

Naïve Bayes

Naive Bayes Classifier (NB) is a probabilistic machine learning model used for classification task. It works based on Bayes' theorem.

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)}$$

Using Bayes' theorem, we can find the probability of it being A, given the occurrence of B. P(A) and P(B) are prior knowledge about that event. The assumption made here is that the predictors/features are independent. This means that the presence of a particular feature does not affect the other. That is why it is called naive.

Decision Tree

Decision Tree (DT) is a tree where each node represents a feature, each branch represents a decision (rule) and each leaf represents an outcome. The whole idea is to create a tree like this for the entire data and process a single outcome at every leaf or minimize the error in every leaf. While implementing a DT, the main issue arises that how to select the best attribute for the root node and for sub-nodes. Information Gain and Gini Index are the popular attribute selection measures for branching. Information gain is the measurement of changes in entropy (a measure of impurity in a node) after the segmentation of a dataset based on an attribute. It calculates how much information a feature provides us about a class. A decision tree algorithm always tries to maximize the value of information gain, and a node/attribute having the highest information gain is split first.

K-Nearest Neighbors

K-Nearest Neighbors (KNN) first identifies the k points in the training data that are closest to the test value and calculates the distance between all those categories. The test value will belong to the category whose distance is the least. It does not have a specialized training phase and uses all the data for training while classification. KNN is a non-parametric learning algorithm because it does not assume anything about the underlying data. KNN uses ‘feature similarity’ to predict the values of new datapoints which further means that the new data point will be assigned a value based on how closely it matches the points in the training set.

2. EXPERIMENTAL SETUP

```
['nb'] = {'alpha': [0.05, 0.1, 0.25, 0.5, 1],
          'fit_prior': [False, True]}

['dt'] = {'max_depth': [10, 20, 50, 100],
          'min_samples_split': [5, 10, 20, 40],
          'min_samples_leaf': [1, 5, 10, 20],
          'max_features' : [None, "sqrt", "log2"]}

['knn'] = {'n_neighbors': [1, 2, 3, 4, 5],
           'algorithm': ['ball_tree', 'kd_tree', 'brute'],
           'weights' : ['uniform', 'distance'],
           'leaf_size': [10, 20, 30, 40, 50]}
```

Figure 1. Tuning Parameters of Naïve Bayes, Decision Tree, and KNN Classifiers

In Figure 1, each parameter to be tuned and corresponding values are presented for each classifier. By using Grid Search, classifiers are tuned over 4-fold validation sets from the training dataset. Training and test dataset size percentages are %67 and %33, respectively.

Number of dimensions of digit vectors was reduced to 100 from 768 by using Truncated Singular Value Decomposition, which is more efficient for sparse matrices, so I can conduct faster experiments since benchmarking is not mainly focused in this assignment. Then, this lower dimensional dataset was min-max scaled.

As increasing time and memory complexity with number of observations, only 2500 observations were used to tune parameters of KNN.

The class distributions in the dataset are as follows.

```
1    0.111524
7    0.104786
3    0.103595
9    0.099714
2    0.099452
6    0.098500
0    0.098381
4    0.096952
8    0.096738
5    0.090357
```

3. EVALUATION

Effects of Naive Bayes Parameters

	alpha	fit_prior	train_acc	val_acc
0	0.05	0.0	0.850	0.847
2	0.10	0.0	0.850	0.847
4	0.25	0.0	0.850	0.847
6	0.50	0.0	0.850	0.847
8	1.00	0.0	0.850	0.847
1	0.05	1.0	0.778	0.776
3	0.10	1.0	0.778	0.776
5	0.25	1.0	0.778	0.775
7	0.50	1.0	0.778	0.775
9	1.00	1.0	0.778	0.775

Table 1. Accuracy by Naïve Bayes Classifiers Given Parameters

According to Table 1, learning class prior probabilities (fit_prior) in NB decreases accuracy as it directly causes a bias towards to given observations. Therefore, using uniform class probabilities is better for generalization here since classes are not imbalanced. Besides, the smoothing parameter (alpha) is invariant to accuracies since it accounts for features not present in the learning samples and prevents zero probabilities in further computations but all features are always given in this experimental setup.

	alpha	fit_prior	train_acc	val_acc
alpha	1.000	0.0	-0.002	-0.002
fit_prior	0.000	1.0	-1.000	-1.000
train_acc	-0.002	-1.0	1.000	1.000
val_acc	-0.002	-1.0	1.000	1.000

Table 2. Correlation Between Parameters of Naïve Bayes and Accuracy

Effects of Decision Tree Parameters

	max_depth	min_samples_split	min_samples_leaf	max_features	train_acc	val_acc
100	50.0	5.0	5.0	All	0.922	0.809
101	50.0	10.0	5.0	All	0.922	0.809
148	100.0	5.0	5.0	All	0.922	0.809
149	100.0	10.0	5.0	All	0.922	0.809
52	20.0	5.0	5.0	All	0.922	0.808
53	20.0	10.0	5.0	All	0.922	0.808
56	20.0	5.0	10.0	All	0.885	0.808
57	20.0	10.0	10.0	All	0.885	0.808
58	20.0	20.0	10.0	All	0.885	0.808
104	50.0	5.0	10.0	All	0.885	0.807

Table 4. Accuracy by Top 10 Decision Tree Classifiers Given Parameters

Decision trees with higher depths can learn better given learning sample as seen in Table 4 but they are readily prone to overfit as well. For this dataset, 50 decision levels as an middle value is good for Decision Tree classifiers.

Another parameter of Decision Trees to control over-fitting is min_samples_split. Higher values prevent a model from learning relations which might be highly specific to the particular sample selected for a tree. Too high values can also lead to under-fitting. However, as seen in Table 5, there is not remarkable

correlation between min_samples_split and validation accuracy score although decrease in it increases the training accuracy.

min_samples_leaf controls the number samples in leafs of decision trees, which is another overfitting parameter, and it is not much effective for the prediction performance for this dataset. Its effect on the predication performance is similar to the previous parameter. It is noted that min_samples_split and min_samples_leaf are affected by max_depth as well.

	max_depth	min_samples_split	min_samples_leaf	train_acc	val_acc
max_depth	1.00	0.00	0.00	0.26	0.12
min_samples_split	0.00	1.00	0.00	-0.20	-0.02
min_samples_leaf	0.00	0.00	1.00	-0.39	-0.07
train_acc	0.26	-0.20	-0.39	1.00	0.82
val_acc	0.12	-0.02	-0.07	0.82	1.00

Table 5. Correlation Between Parameters of Decision Tree and Accuracy

max_features	count	mean	std	min	25%	50%	75%	max
All	64.0	0.80	0.01	0.79	0.79	0.80	0.81	0.81
log2	64.0	0.58	0.03	0.50	0.57	0.59	0.60	0.61
sqrt	64.0	0.64	0.03	0.58	0.64	0.65	0.66	0.67

Table 6. Accuracy Statistics of Values for Max_Features of Decision Tree Classifiers

Another important parameter for Decision Trees is the number features used to split samples. According to Table 6, using all features provides better classification as expected. Indeed, this parameter is critical for acquiring generalized ensemble models of Decision Trees, but, we have only one tree to classify the dataset, so we should use all features for the branching.

Effects of K-Nearest Neighbors Parameters

	n_neighbors	algorithm	weights	leaf_size	train_acc	val_acc
0	1.0	ball_tree	uniform	10.0	1.0	0.86
1	1.0	ball_tree	distance	10.0	1.0	0.86
3	2.0	ball_tree	distance	10.0	1.0	0.86
10	1.0	ball_tree	uniform	20.0	1.0	0.86
11	1.0	ball_tree	distance	20.0	1.0	0.86
13	2.0	ball_tree	distance	20.0	1.0	0.86
20	1.0	ball_tree	uniform	30.0	1.0	0.86
21	1.0	ball_tree	distance	30.0	1.0	0.86
23	2.0	ball_tree	distance	30.0	1.0	0.86
30	1.0	ball_tree	uniform	40.0	1.0	0.86

Table7. Accuracy by Top 10 KNN Classifiers Given Parameters

Using the nearest neighbor observation for the classification of handwritten digits performs best as seen in Table 7. It is reasonable when vectors of each class is well different to each other and we have enough representations of each class. On the other, leaf size of tree search did not differ to effect

prediction performance as seen Table 8. It seen that using more than 10 observations provides a performance like brute force.

	n_neighbors	leaf_size	train_acc	val_acc
n_neighbors	1.00	0.0	-0.36	-0.49
leaf_size	0.00	1.0	0.00	0.00
train_acc	-0.36	0.0	1.00	0.91
val_acc	-0.49	0.0	0.91	1.00

Table 8. Correlation Between Parameters of KNN and Accuracy

	count	mean	std	min	25%	50%	75%	max
weights								
distance	75.0	0.85	0.01	0.84	0.85	0.86	0.86	0.86
uniform	75.0	0.83	0.02	0.82	0.82	0.82	0.84	0.86

Table 9. Accuracy Statistics of Values for Weighting Parameter of KNN Classifiers

Weighting in voting for class based on the distance of neighbor observations can increase the learning performance as seen in the Table 9. This increase is limited to %2 more accuracy compare to equally (uniformly) weighting because used neighborhood sizes are small which leads to the voting by the observations all close to each other.

	count	mean	std	min	25%	50%	75%	max
algorithm								
ball_tree	50.0	0.84	0.02	0.82	0.82	0.85	0.86	0.86
brute	50.0	0.84	0.02	0.82	0.82	0.85	0.86	0.86
kd_tree	50.0	0.84	0.02	0.82	0.82	0.85	0.86	0.86

Table 10. Accuracy Statistics of Values for KNN Classifier Algorithms

For efficient exploration of neighbors, tree algorithms are used in KNN which lead the same prediction performance compared to each other and brute force search. So, we can reliably use one of that tree searches in KNN for MNIST dataset classificaiton.

4. RESULTS

According to the experiment results, the best setting of each classifier as follows.

MultinomialNB(alpha=0.05, fit_prior=False)
DecisionTreeClassifier(max_depth=50, min_samples_leaf=5, min_samples_split=5)
KNeighborsClassifier(algorithm='ball_tree', leaf_size=10, n_neighbors=1)

Using those parameters, each classifier is trained over all of the training dataset and scored for the test as well. Table 11 presents accuracies by the best models for the training and testing observations.

We see that Naïve Bayes provides a generalized performance for classification MNIST dataset. But compared to other classifiers it remains naïve since it follows some assumptions about the dataset.

KNN outperforms Decision Tree Classifier naturally since it memorizes a large dataset. In addition, this Decision Tree misclassifies the test samples at %10 more compared to the training performance which indicates that it overfitted the given learning sample.

	train_acc	test_acc
nb	0.850	0.852
dt	0.927	0.823
knn	1.000	0.948

Table 11. Accuracies by the Best Models

Figure 2. Class Accuracies by the Best Models

Figure 2 presents, prediction accuracies for each class of the digits by the best models. Also, confusion matrices of training and test predictions are provided in Table 12. From Figure 2, we can imply that Naïve Bayes follows almost same prediction accuracy in both training and test samples. Also, Decision Tree more misclassifies the digits other than 0 and 1 in the test sample.

		Training											Test									
Naive Bayes		**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**		**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**
	0	0.91	0	0	0.01	0	0.03	0.03	0	0.02	0	**0**	0.9	0	0	0.01	0	0.03	0.03	0	0.02	0
	1	0	0.96	0.01	0	0	0.01	0	0	0.02	0	**1**	0	0.96	0	0	0	0.01	0.01	0	0.02	0
	2	0.01	0.03	0.8	0.02	0.03	0.01	0.02	0.02	0.04	0.01	**2**	0.01	0.03	0.8	0.02	0.04	0.01	0.02	0.02	0.04	0.01
	3	0	0.02	0.03	0.8	0	0.05	0.01	0.02	0.04	0.03	**3**	0	0.02	0.03	0.8	0	0.06	0.01	0.02	0.04	0.04
	4	0	0.01	0.01	0	0.86	0	0.01	0	0.01	0.08	**4**	0	0.01	0	0	0.87	0	0.01	0	0.01	0.09
	5	0.01	0.02	0	0.07	0.03	0.76	0.02	0.02	0.03	0.02	**5**	0.01	0.02	0	0.06	0.03	0.77	0.03	0.01	0.02	0.02
	6	0.01	0.01	0.01	0	0.02	0.03	0.91	0	0.01	0	**6**	0.01	0.01	0.01	0	0.03	0.03	0.89	0	0.01	0
	7	0	0.03	0.01	0	0.03	0	0	0.87	0	0.06	**7**	0	0.03	0.01	0	0.03	0	0	0.85	0	0.07
	8	0.01	0.03	0.01	0.05	0.01	0.05	0.01	0.01	0.78	0.03	**8**	0.01	0.03	0	0.05	0.02	0.04	0.01	0	0.8	0.03
	9	0.01	0.01	0	0.01	0.06	0.01	0	0.04	0.01	0.84	**9**	0.01	0.01	0	0.01	0.06	0.01	0	0.04	0.01	0.85
Decision Tree		**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**		**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**
	0	0.96	0	0	0	0	0.01	0.01	0	0.01	0	**0**	0.92	0	0.01	0	0	0.02	0.02	0.01	0.01	0.01
	1	0	0.98	0	0	0	0	0	0	0	0	**1**	0	0.96	0.01	0	0	0	0	0	0.01	0
	2	0.01	0.01	0.93	0.01	0.01	0	0.01	0.01	0.01	0	**2**	0.02	0.01	0.83	0.02	0.02	0.02	0.03	0.02	0.03	0.01
	3	0.01	0.01	0.02	0.92	0.01	0.02	0	0	0.02	0.01	**3**	0.01	0.01	0.04	0.8	0.01	0.06	0	0.01	0.05	0.03
	4	0	0.01	0	0	0.94	0	0.01	0.01	0.01	0.02	**4**	0.01	0.01	0.01	0	0.81	0.01	0.02	0.02	0.01	0.1
	5	0.01	0	0.01	0.02	0.01	0.91	0.01	0	0.02	0.01	**5**	0.03	0.01	0.01	0.05	0.02	0.75	0.02	0.01	0.05	0.02
	6	0.01	0	0.01	0	0.01	0.01	0.94	0	0	0	**6**	0.03	0.01	0.03	0.01	0.02	0.03	0.86	0.01	0.01	0
	7	0	0.01	0.01	0.01	0.02	0.01	0	0.93	0	0.02	**7**	0.01	0.01	0.02	0.01	0.03	0.01	0	0.83	0.02	0.07
	8	0.01	0	0.02	0.02	0.02	0.03	0	0.01	0.87	0.01	**8**	0.02	0.01	0.05	0.06	0.03	0.07	0.02	0.02	0.7	0.02
	9	0.01	0	0.01	0.01	0.04	0.01	0	0.02	0.01	0.89	**9**	0.01	0.01	0.01	0.01	0.12	0.01	0.01	0.06	0.02	0.73
KNN		**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**		**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**
	0	1	0	0	0	0	0	0	0	0	0	**0**	0.99	0	0	0	0	0	0	0	0	0
	1	0	1	0	0	0	0	0	0	0	0	**1**	0	1	0	0	0	0	0	0	0	0
	2	0	0	1	0	0	0	0	0	0	0	**2**	0.02	0.02	0.93	0.01	0	0	0	0.02	0	0
	3	0	0	0	1	0	0	0	0	0	0	**3**	0	0	0	0.94	0	0.03	0	0.01	0.01	0.01
	4	0	0	0	0	1	0	0	0	0	0	**4**	0	0.01	0	0	0.94	0	0	0.01	0	0.03
	5	0	0	0	0	0	1	0	0	0	0	**5**	0	0	0	0.04	0	0.91	0.01	0	0.01	0.01
	6	0	0	0	0	0	0	1	0	0	0	**6**	0.01	0	0	0	0	0	0.98	0	0	0
	7	0	0	0	0	0	0	0	1	0	0	**7**	0	0.01	0	0	0	0	0	0.96	0	0.02
	8	0	0	0	0	0	0	0	0	1	0	**8**	0.01	0.02	0.01	0.02	0.01	0.04	0.01	0	0.88	0.02
	9	0	0	0	0	0	0	0	0	0	1	**9**	0.01	0	0	0.01	0.01	0	0	0.03	0	0.93

Table 12. Confusion Matrices of Training and Test Predictions by Naïve Bayes, Decision Tree, and KNN Classifiers