Özyeğin University

CS551 – Introduction to Artificial Intelligence

Assignment-3

Image Classification & Dimensionality Reduction

Furkan Cantürk

03.01.2021

1. INTRODUCTION

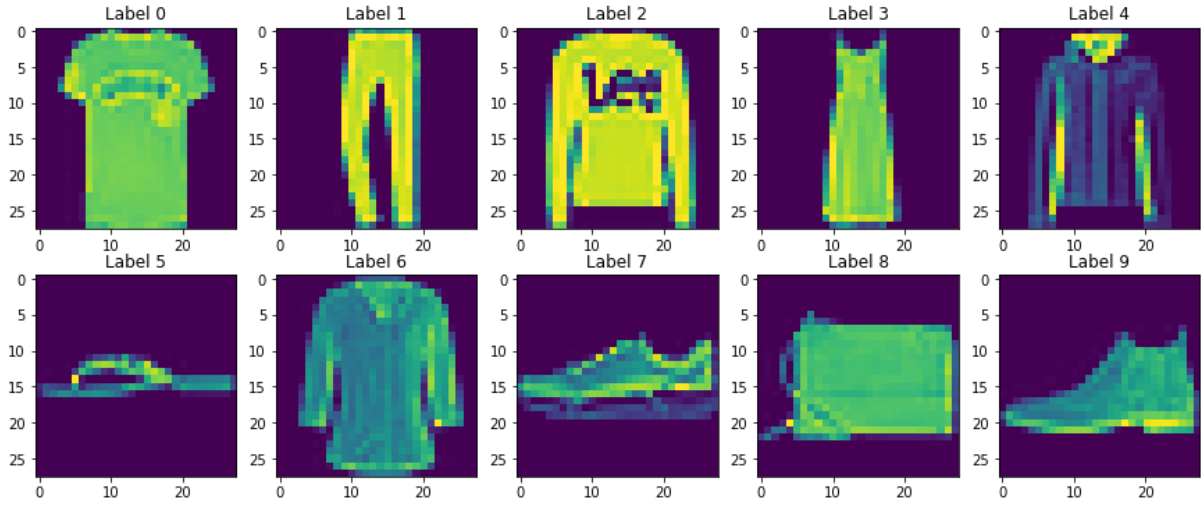

Figure 1. Fashion-MNIST Dataset

5 supervised algorithms are benchmarked in this report for a multi-classification problem, which are K-Nearest Neighbors, Perceptron, Support Vector Machine, Decision Tree and Random Forest. Firstly, all models are trained on Fashion MNIST Dataset. Secondly, cardinality of the dataset is reduced to some number of components by Principal Component Analysis and all models are trained with the reduced datasets. Lastly, the performance of models for each reduction level is compared based on test accuracy.

Fashion MNIST have 10 different labels for some 28x28 cloth images. There are 60000 training images and 10000 images. The weight of classes in both training and test dataset are balanced.

The used algorithms are briefly explained in the following subsections.

1.1. Perceptron

The Perceptron (Pt) is a linear learning algorithm for binary classification tasks. It learns using the stochastic gradient descent optimization algorithm. It consists of a single node or neuron that takes a row of data as input and predicts a class label. This is achieved by calculating the weighted sum of the inputs and a bias (set to 1). The weighted sum of the input of the model is called the activation.

$$\text{Activation} = \text{Weights} * \text{Inputs} + \text{Bias}$$

If the activation is above 0, the model will output 1; otherwise, it will output 0.[1]

One-vs-All (OvA) is implicitly used in Perceptron classifier in Scikit-Learn for multi-class problems. OvA involves splitting the multi-class dataset into multiple binary classification problems. A binary classifier is then trained on each binary classification problem and predictions are made using the model that is the most confident. [2]

1.2. 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 [3].

1.3. Support Vector Machine

Support Vector Machine Algorithm (SVM) finds a hyperplane in an N-dimensional space that distinctly classifies the data points. The objective is to find a plane that has the maximum margin (distance between data points) of classes. Maximizing the margin distance provides some reinforcement so that future data points can be classified with more confidence. [4] SVM is effective in high dimensional spaces and is also memory efficient. [5]

1.4. 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. It can be calculated using the below formula:

1.5. Random Forest

Random Forest (RF) consists of a large number of independent decision trees that operate as an ensemble. Each tree in spits out a class prediction and the class with the most votes becomes our model's prediction. When training, each tree in a random forest learns from a random sample of the data points. The samples are drawn with replacement, known as bootstrapping, which means that some samples will be used multiple times in a single tree. The idea is that by training each tree on different samples, although each tree might have high variance with respect to a particular set of the training data, overall, the entire forest will have lower variance but not at the cost of increasing the bias. Also, each tree considers random subsets of features considered when splitting node

At test time, predictions are made by averaging the predictions of each decision tree. This procedure of training each individual learner on different bootstrapped subsets of the data and then averaging the predictions is known as bagging, short for bootstrap aggregating. [7]

1.6. Principal Component Analysis

Principal Component Analysis (PCA) is a statistical technique which reduces the dimensions of the data and help us understand, plot the data with lesser dimension compared to original data. As the name says PCA helps us compute the Principal components in data. Principal components are basically vectors that are linearly uncorrelated and have a variance within data. From the principal components top p is picked which have the most variance. [8]

2. TRAINING

The following hyperparameters of the models are set by just observation of test accuracy and training & prediction speed in a few trials, which are different than default values in Scikit-Learn Library.

Pt: Learning rate = 0.2

KNN: Algorithm = Ball Tree, number of neighbors = 50, and leaf size = 10.

SVM: Default parameters are used.

DT: The minimum number of samples required to be at a leaf node = 10, and the number of features to consider when looking for the best split = 'auto', which is equal to $\sqrt{N}$ where N is the number of observations.

RF: The minimum number of samples required to be at a leaf node = 10, and number of decision trees = 20.

Before training the models, data sets are reshaped to 2-D array, so 784 features at total, and pixels are scaled between 0 and 1.

For SVM training with all 784 features, a random sample at size 10000 from training data set is used because its training time takes much longer when N increases.

3. DIMENSIONALITY REDUCTION

Firstly, a PCA model is fitted to the training dataset using all features, i.e., 784 components. This model gives how much variance is explained by each component. 5 variance ratios are experimented in this report, which are 0.25, 0.50, 0.75, 0.9, and 0.95. Based on these thresholds, 5 different number of components (n) are determined, which are 1, 3, 14, 84, and 188 where each explains %29, %53, %75, %90, and %95 variance of the training data, respectively. A PCA model is fitted to training data for each n and 5 reduced datasets are generated.

4. RESULTS

Table 1. Test Accuracy by Each Model Trained with N Components and Respective Explained Variance Ratios

EV Ratio	0.29	0.53	0.75	0.90	0.95	1.00
#Components	1	3	14	84	188	784
Pt	0.11	0.45	0.68	0.77	0.79	0.80
KNN	0.30	0.65	0.82	0.84	0.84	0.83
SVM	0.31	0.64	0.84	0.88	0.89	0.85
DT	0.26	0.61	0.76	0.74	0.68	0.79
RF	0.28	0.65	0.82	0.84	0.84	0.86

Test accuracy by each model trained on original data set and reduced datasets are presented in Table 1. Using all features, KNN, SVM, and RF models achieved approximately same accuracy level on the test data while Pt and DT slightly underperformed.

Using 1 and 3 components, Pt achieves too low accuracy compared to others but gains four times better accuracy with 3 components to 1 component. Using 3 components accuracy is doubled by KNN, SVM, DT, and RF models.

Using 14 components, there is a still substantial improvement in test accuracy by each model.

Using 84 components which explains 90% variance of training data is the most efficient dimensionality reduction of the dataset for these 5 models compared to using 188 and 784 components. Also, it is figured out that SVM performs better when not using all components of Fashion MNIST.

Pt	0	1	2	3	4	5	6	7	8	9	TR
0	734	4	31	17	0	3	117	0	8	0	0.80
1	6	943	3	16	0	0	1	0	0	0	0.97
2	39	7	634	39	49	0	155	0	6	0	0.68
3	60	23	10	780	12	1	42	0	4	0	0.84
4	67	18	301	114	927	2	432	0	17	0	0.49
5	1	0	0	0	0	855	0	19	4	1	0.97
6	76	1	11	20	10	0	230	0	4	1	0.65
7	0	0	1	0	0	68	0	944	5	40	0.89
8	17	3	8	8	2	16	23	1	951	0	0.92
9	0	1	1	6	0	55	0	36	1	958	0.91

KNN	0	1	2	3	4	5	6	7	8	9	TR
0	842	14	21	65	1	2	236	0	2	0	0.71
1	0	926	0	11	1	0	1	0	1	0	0.99
2	35	16	686	10	111	0	153	0	43	1	0.65
3	43	32	6	802	32	3	23	0	16	1	0.84
4	11	7	144	62	704	0	84	0	9	0	0.69
5	1	0	0	0	0	696	0	2	1	7	0.98
6	56	4	137	47	146	17	484	1	31	3	0.52
7	1	0	0	0	0	180	1	909	19	44	0.79
8	11	1	6	3	5	4	18	0	876	0	0.95
9	0	0	0	0	0	98	0	88	2	944	0.83

SVM	0	1	2	3	4	5	6	7	8	9	TR
0	852	4	17	31	1	0	193	0	0	0	0.78
1	0	950	0	5	0	0	1	0	0	0	0.99
2	6	9	775	8	87	0	122	0	6	0	0.77
3	45	30	12	881	34	3	35	0	6	0	0.84
4	5	3	116	34	787	0	91	0	1	0	0.76
5	2	0	0	0	0	924	0	33	3	19	0.94
6	74	3	74	37	85	0	532	0	6	0	0.66
7	0	0	0	0	0	48	0	918	5	40	0.91
8	16	1	6	4	6	2	26	0	973	1	0.94
9	0	0	0	0	0	23	0	49	0	940	0.93

DT	0	1	2	3	4	5	6	7	8	9	TR
0	771	8	18	40	8	0	171	0	8	1	0.75
1	5	921	1	22	3	0	6	1	1	0	0.96
2	22	6	686	19	152	0	130	0	12	0	0.67
3	52	39	19	825	50	10	40	0	13	0	0.79
4	15	7	141	43	676	1	112	0	11	1	0.67
5	3	1	1	0	0	874	1	53	18	43	0.88
6	118	12	122	43	100	0	513	0	15	0	0.56
7	0	1	0	0	0	62	0	874	11	60	0.87
8	14	5	12	7	10	16	27	5	908	4	0.90
9	0	0	0	1	1	37	0	67	3	891	0.89

RF	0	1	2	3	4	5	6	7	8	9	TR
0	834	5	11	21	0	0	147	0	0	0	0.82
1	0	953	0	2	1	0	0	0	1	0	1.00
2	18	7	788	15	104	0	125	0	6	1	0.74
3	36	27	11	894	44	1	37	0	4	0	0.85
4	5	3	129	38	796	0	105	0	7	0	0.73
5	1	0	0	0	1	942	0	21	2	11	0.96
6	93	4	55	28	52	0	564	0	8	0	0.70
7	0	0	0	0	0	40	0	922	4	42	0.91
8	13	1	6	2	2	2	22	0	966	3	0.95
9	0	0	0	0	0	15	0	57	2	943	0.93

Figure 1. Confusion Matrices for Test Data Set by Pt, KNN, SVM, DT, and RF Models Respectively

Figure 1 presents the confusion matrices of prediction of the test data by each model trained with all components. True prediction rates of each class are also presented at the right of confusion matrix.

The image with label 6 is the most confused class with labels 0, 2 and 4. It is interpretable because these are similar shaped clothes which are t-shirt, sweat, jacket, and shirt. Similarly, class 5 is confused with class 7 by each model.

REFERENCES

[1] https://machinelearningmastery.com/perceptron-algorithm-for-classification-in-python/

[2] https://machinelearningmastery.com/one-vs-rest-and-one-vs-one-for-multi-class-classification/

[3]https://www.tutorialspoint.com/machine_learning_with_python/machine_learning_with_python_knn_algorithm_finding_nearest_neighbors.htm

[4] https://towardsdatascience.com/support-vector-machine-introduction-to-machine-learning-algorithms-934a444fca47

[5] https://scikit-learn.org/stable/modules/svm.html

[6] https://www.javatpoint.com/machine-learning-decision-tree-classification-algorithm

[7] https://towardsdatascience.com/an-implementation-and-explanation-of-the-random-forest-in-python-77bf308a9b76

[8] https://medium.com/@raghavan99o/principal-component-analysis-pca-explained-and-implemented-eeab7cb73b72