Report for EE5907 CA2 Programming Assignment

National University of Singapore

Name: Zhang Fei

Matric Number: A0117981X

Introduction: This assignment aims to classify handwritten digits with reduced dimensional features by Principal Component Analysis (PCA). The standardised MNIST database consists of 70,000 images, each of size 28 by 28 greyscale pixels. There is a standard set of 60,000 training images and a standard set of 10,000 test images. The variance among the train set is derived in terms of covariance matrix and the new eigen vectors of covariance matrix form the directions of new feature space. The main idea is to improve accuracy on test set when the images are projected to the new feature space. Different models are investigated and analysed to improve the accuracy on test set.

1. Calculating Eigenvalues and Eigenvectors for Features Extraction

1.1 Compute the mean of the training digits

In this assignment, Eigenvalues and Eigenvectors of image training set in MNIST Database are investigated for Features Extraction. I will apply Principal Component Analysis (PCA) on the train set for dimensionality reduction. Firstly, Compute the mean of the train set digits, and subtract the mean from each train set data point:

```
# mean of the training digits(mean of per feature)
meanVal = np.mean(x_train_raw,axis = 0)

# subtract the mean from each training data point
x_train = x_train_raw - meanVal
```

This is to make all the training data set Zero Centring. It helps the network model to learn faster since gradients act uniformly for each feature channel.

1.2 Compute the covariance matrix

Secondly, Compute the covariance matrix using all the train images. Perform singular value decomposition (SVD) on train set data X:

$$X_{d,n} = U_{d,d} D_{d,n} V_{d,d}^T$$

Then the covariance matrix can be written as:

$$S = XX^T = UD^2U^T$$

```
# Compute the covariance matrix using all the training images
covMat = np.cov(x_train ,rowvar = 0)
```

```
# get eigenvalue and eigenvector from covirance matrix
featValue,featVec = np.linalg.eig(covMat)
# sort the eigenvalue in ascending order and return index
index = np.argsort(featValue)
```

1.3 Visualize the Eigenvectors

A set of eigenvector digits can be produced through PCA on the train set. The eigenvectors can be considered as the feature set on the train set. In this assignment, the columns of $U_{d,p}$ form the eigenvector digits. Every sample of a hand-written digit x_i can be represented as a linear combination of these eigenvector plus an average digit $\bar{x}$.

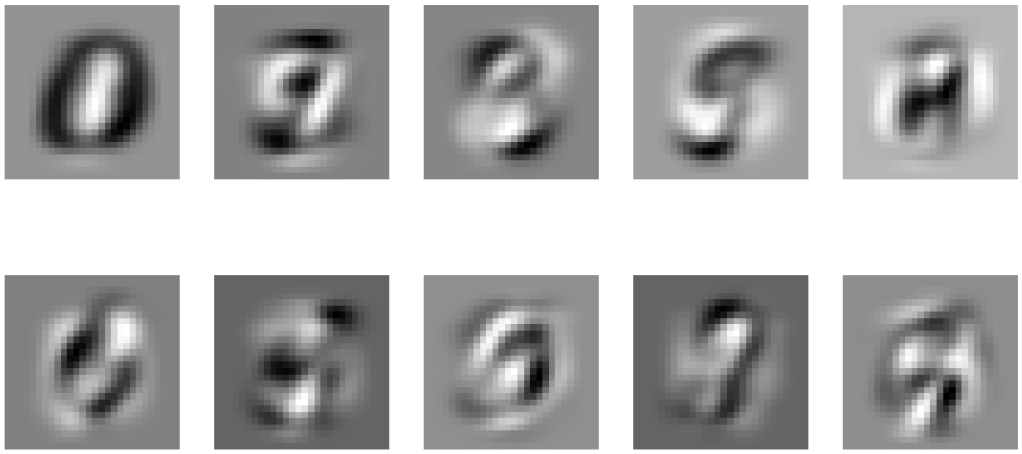


Figure 1 First 10 eigen-digits sampled from train set

Figure 1 shows the first 10 eigen-digits, corresponding to the 10 largest eigenvalues projected from train set. In other words, the first 10 columns of $U_{d,p}$ is visualized. Among those eigen-digits, the first eigen-digit looks like a digit most, as it extracts the most information and features of handwritten digits.

1.4 Handwritten digit image reconstruction

1.4.1 Original images from the training set

Figure 2 selected original images from the training set

Figure 2 shows the images corresponding to columns 2, 4, 6, 8, 10, 12, 14, 16, 18, 20 in the train set. It shows digits from 0 to 9.

1.4.2 Reconstructed images from the test set

Second row projected images:

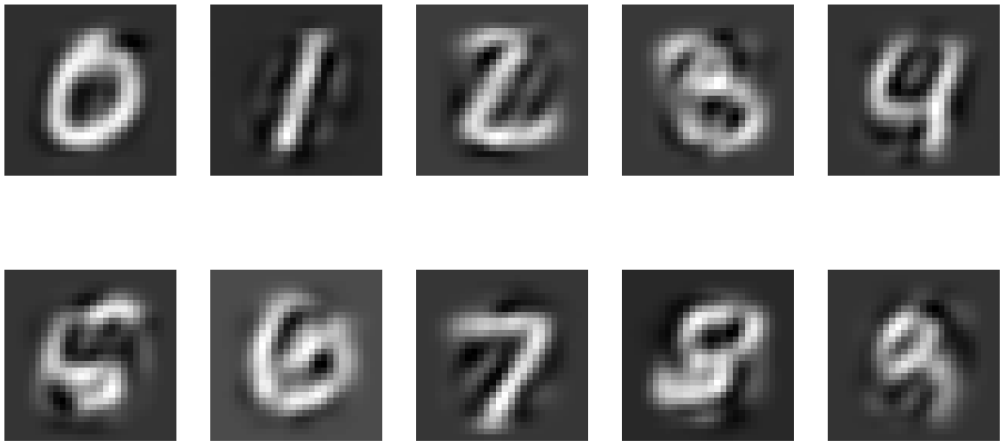


Figure 3 reconstructed images from selected the test set

Figure 3 shows the reconstructed images corresponding to columns 4, 3, 2, 19, 5, 9, 12, 1, 62, 8 in the test set. And we can see the visualized images show digits from 0 to 9. The reconstructed images show that the eigenvector has a good approximation on features extraction.

2. Using Eigen-digits to Classify each digit Image

2.1 Apply linear regression to learn the 30-dimensional data

2.1.1 MSE Using Indicator Matrix (Primal):

We define loss vector $\mathbf{e} = \mathbf{Ya} - \mathbf{b}$, to minimize the loss function is equivalent to minimize the sum of squared error criterion function:

$$J_s(a) = ||Ya - b||^2 = \sum_{i=1}^{n} (a^t y_i - b_i)^2$$

We differentiate the cost function and make the equation equal to zero, this will yield:

$$Y^t Ya = Y^t b$$

Since the new train and test set data features are all in 30 dimensions, Y^tY is non-singular, we can derive weight vector a = $(Y^tY)^{-1}Y^tb$

And the classification accuracy rate on test set is 83.32%.

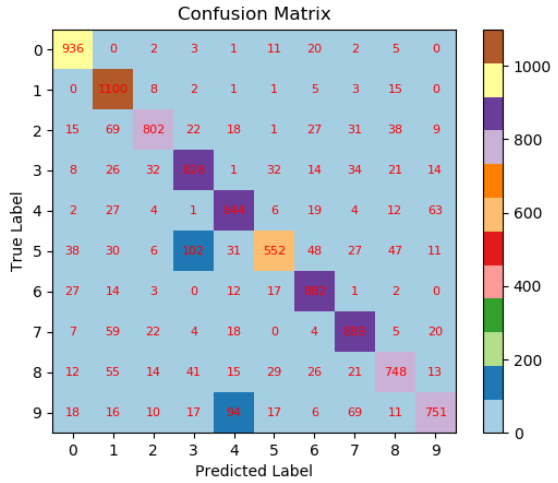


Figure 4 Confusion Matrix in linear regression

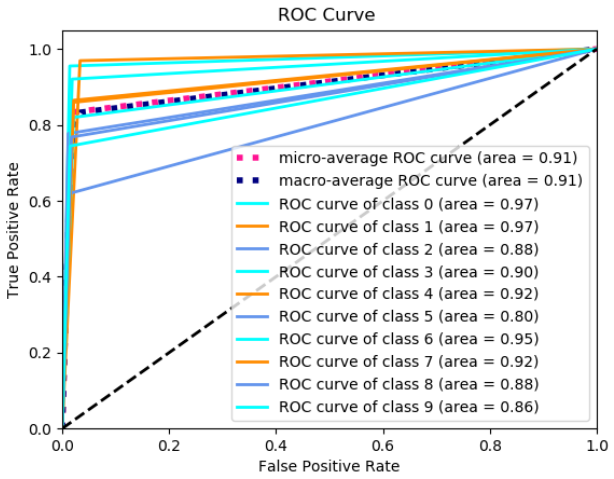


Figure 5 ROC Curve in linear regression

Figure 4 is the Confusion Matrix figure which shows the prediction results on test data. The number of correct and incorrect predictions are summarized with count values and broken down by each class. We can visualize that the error counts in prediction of digit 5 and digit 9 are 102 and 94 respectively, which is quite large. And we can see that the accuracy of the model is based on total number of correct predictions over total number of labels. High accuracy dose not guarantee good performance of the model unless the test set has similar distributions for all the classes of labels.

Figure 5 shows the ROC curve of the TPR vs. FPR at different classification thresholds. Areas under ROC measure of performance across all possible classification thresholds. we can see that most of AUG for different classes are between 0.86~0.92.

2.1.2 Stochastic Gradient Descent (SGD) method:

The linear discriminant function is:

$$g(x) = w_1 x_1 + \ldots + w_d x_d + w_0$$

And the updated weight can be expressed as such:

$$a \leftarrow a + \eta(k)(b_k - a^t y^k) y^k$$

The initial learning rate is set as 0.005. With 100 epochs of training, the classification accuracy on test set is 83.38%.

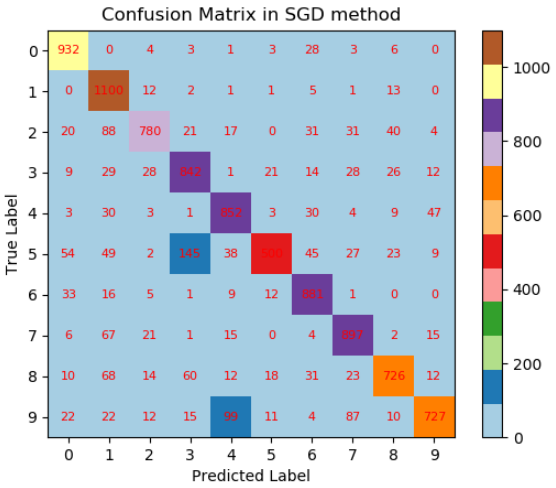


Figure 6 Confusion Matrix in SGD method

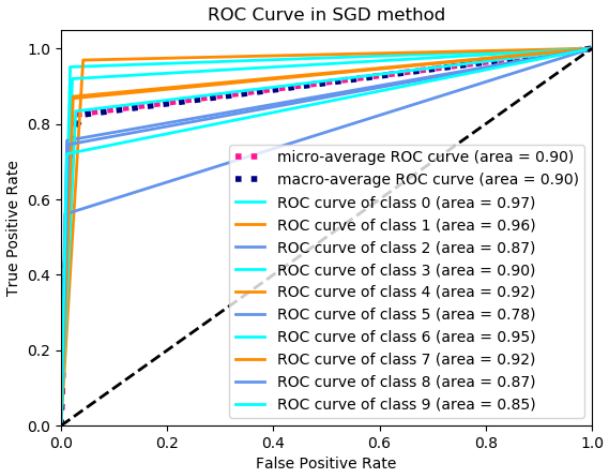


Figure 7 ROC Curve in SGD method

Figure 6 shows the Confusion Matrix obtained by using SGD method. The predicted results in matrix are quite similar with results obtained by pseudoinverse matrix in Figure 4. And the ROC curve in SGD method is also similar with ROC curve obtained by pseudoinverse matrix.

We can see that both accuracies obtained from MSE Without Flipping (Primal) Method and Gradient descent are quite close. Therefore, for high dimensions of features, we can apply PCA to reduce dimension complexities, then we can compute weights directly if there is no singular matrix occurred. Otherwise, we can apply other methods such as gradient descent to find out optimal weight vector.

2.2 Apply polynomial regression to learn the 30-dimensional data

2.2.1 MSE Using Indicator Matrix (Primal):

To get the multivariate polynomial terms, I imported PolynomialFeatures in scikit learn package. I applied 2rd order on the features to retrieve 2rd order polynomial regression features. Therefore, every single datapoint dimension is expanded from 1x30 into 1x496 dimension space. When apply MSE Using Indicator Matrix (Primal), we can get weight vectors by equation a = $(Y^tY)^{-1}Y^tb$.

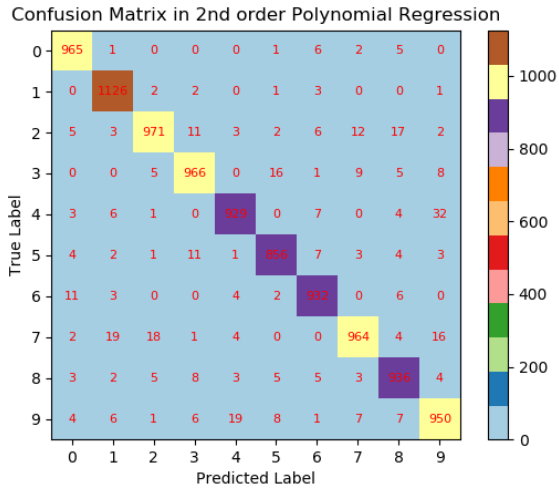


Figure 7 Confusion Matrix

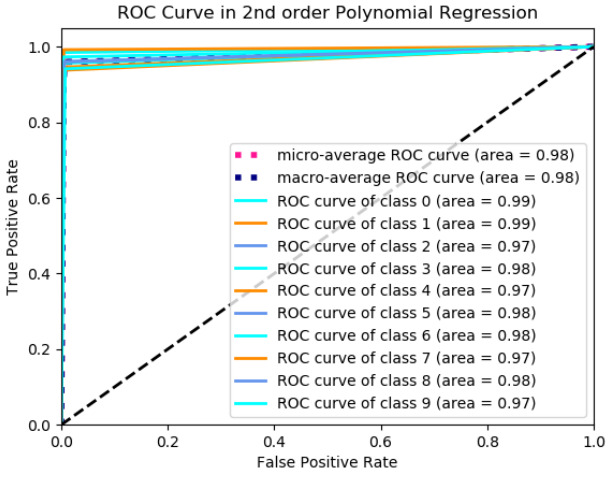


Figure 8 ROC Curve

Figure 7 and Figure 8 show the Confusion Matrix and ROC curve of 2nd order polynomial regression model obtained by pseudoinverse matrix method. The classification accuracy rate on test set is 95.95%. This resulting model has a drastic improvement on accuracy than the accuracy obtained by linear regression model. The AUG score in ROC curve for different classed are all increased. Therefore, this 2rd order polynomial regression model fits a wide range of curvature. The new features now can be clearly explained and identified.

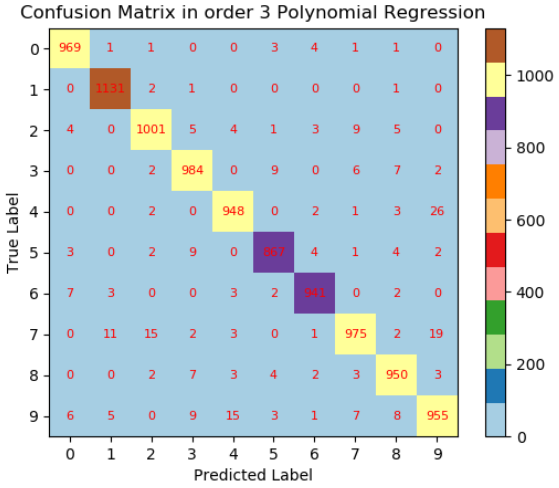


Figure 9 Confusion Matrix

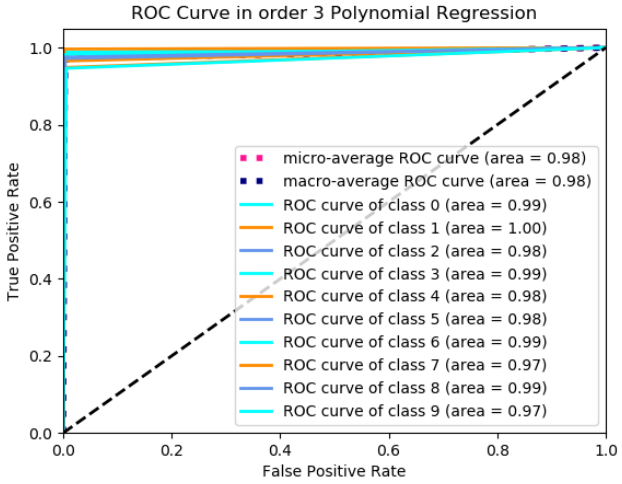
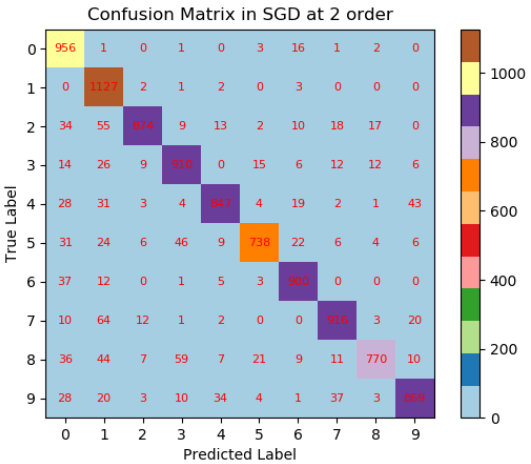


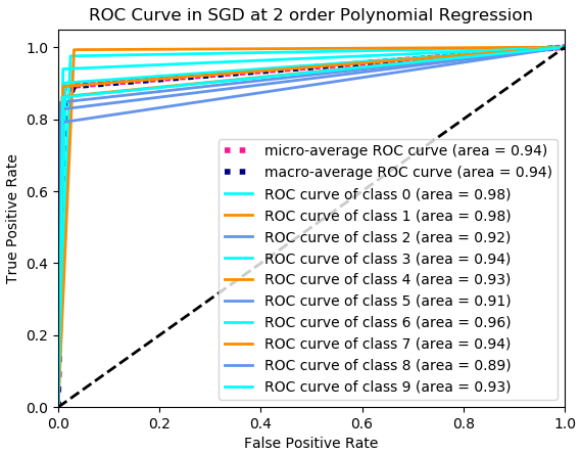
Figure 10 ROC Curve

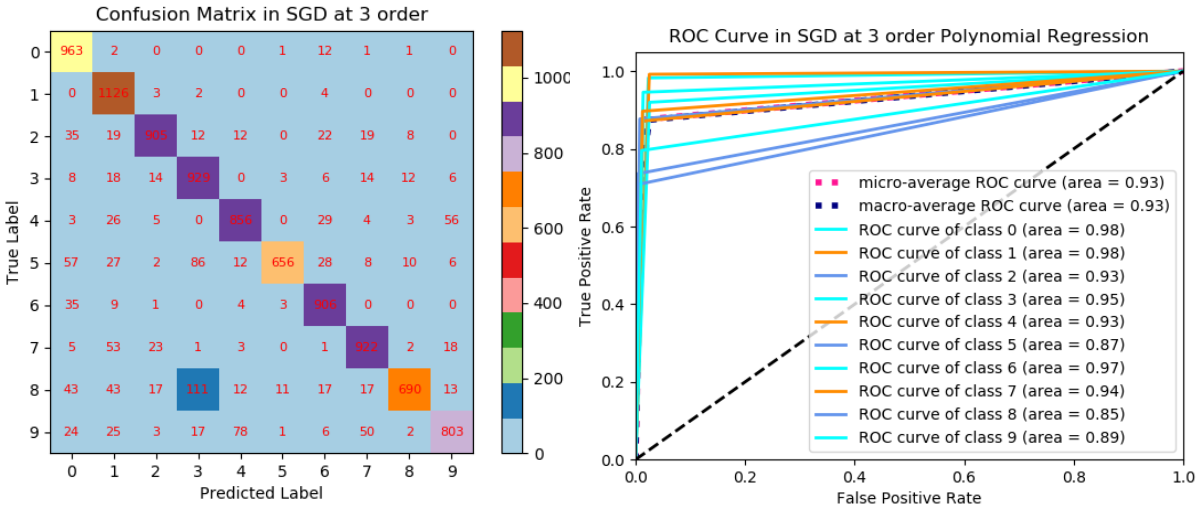
Figure 9 and Figure 10 show the Confusion Matrix and ROC curve of 3rd order polynomial regression model obtained by pseudoinverse matrix method. With the 3rd order term in polynomial regression (the eigen vector dimension set 20 due to computer memory limitation), the model accuracy increases to 97.21%, while maintaining a similarly high AUG score 0.97~0.99. we can see that projected features from 2nd order to 3rd order polynomial regression model fit better. However, we cannot guarantee higher orders of polynomial regression model will always result high accuracy due to an explosion in number of features and lead to the curse of dimensionality.

2.2.2 Stochastic Gradient Descent method

With the same gradient descent model as in linear regression, the classification accuracy of polynomial with 2nd order on test set is 89.08%. the classification accuracy of polynomial with 3rd order on test set is 87.61%.







The figures above show the Confusion Matrix and ROC Curve obtained by SGD method with 2nd and 3rd order polynomial model. Both results are better than results obtained by SGD in linear model. And in the SGD method, the solution is not unique which means we can train the model with more epochs or smaller learn rate to improve performance. This is different from pseudoinverse method which we can only get one unique solution. And when encountering singular matrix, we have to use SGD or other methods instead.

3. Improvement on test accuracy

3.1 Increase number of eigen vectors in MSE Using Indicator Matrix (Primal)

Table 3-1 Accuracy of different Eigen vectors

No. of Eigen Vectors	30 Eigen Vectors	35 Eigen Vectors	40 Eigen Vectors	50 Eigen Vectors	60 Eigen Vectors	80 Eigen Vectors	100 Eigen Vectors	150 Eigen Vectors
Accuracy on Linear Regression	83.33%	84.18%	84.82%	85.57%	85.81%	86.25%	86.28%	86.24%
Accuracy on Polynomial (2nd order) Regression	95.95%	96.33%	96.67%	97.28%	97.38%	97.59%	x	x

Table 3-1 show the accuracies in test set increase in both Linear Regression model and Polynomial Regression model as the number of eigen vectors increase from 30 to 150. Though eigen vectors above 100 cannot be computed out in 2rd order polynomial regression due to computer RAM limitation, the accuracy still can reach up to 97.59% in 80 dimensions of eigen vectors.

3.2 L_2 regularization on Ridge Regression:

Ridge Regression is to add L2 regularization on loss function. The expression is:

$$J_s(a) = ||Ya - b||^2 + \lambda \mathrm{a}^{\mathrm{T}}\mathrm{a}$$

Differentiate the loss function respect to a, and let equation equals to 0, we get:

$$a = (Y^tY + \lambda I)^{-1}Y^tb, \text{ where I is identity matrix.}$$

I apply L_2 regularization Ridge Regression on 2nd order polynomial regression model.

Table 3-2 Accuracy on Ridge model with different L_2 factor

Regularization factor	0.001	0.005	0.01	0.05	0.1	0.5	1
Accuracy	95.96%	95.94%	95.89%	95.76%	95.67%	94.83%	94.32%

Table 3-2 shows Accuracy on Ridge model with different L_2 factor. From the table we can see that as regularization factor increases, the accuracy decreases. The Ridge regression model does not help improve accuracy rate.

3.3 Implementation of classifiers

Four classifiers are implemented and investigated to evaluate the accuracy on test set.

3.3.1 Logistic Regression Classifier

Logistic regression is a machine learning classifier used to predict binary classification. The difference with linear regression is that the decision function is logistic function (Sigmoid). The method One-vs-all technique helps extend the binary classification to multi-class classification, which involves training N distinct binary classifiers, each designed for recognizing a particular label class. Regularization factor is added in the model.

The accuracy obtained in linear regression feature is 88.74%, the accuracy obtained in polynomial regression feature is 97.94%

3.3.2 MLP Classifier

MLP Classifier is Multi-Layer Perceptron which replies on neuron network to perform classification. Different layers with neurons MLP models are implemented. From the table we can see that the overall accuracy is 97%~98%. And MLP with 1 layer could perform better than more layers. The highest accuracy obtained is 98.18% in one hidden layer with 100 neuron nodes.

Table 3-3-2 MLP classifier with different layers and nodes

Model	50	100	400	100-50	100-50-10
Linear	97.22%	97.78%	98.14%	97.71%	97.56%
Polynomial	97.98%	98.16%	98.15%	98.0%	97.57%

3.3.3 KNN Classifier

Table 3-3-3 KNN classification accuracies at different K values.

K	1	3	5	7	9	11	13	15	17	19
Linear Feature	97.18%	97.41%	97.55%	97.51%	97.40%	97.46%	97.21%	97.22%	97.18%	97.21%
Polynomial Feature	96.97%	97.41%	97.35%	97.34%	97.18%	97.27%	97.16%	97.07%	96.99%	96.86%

We can see that the KNN classifier has stable prediction results. All the accuracies are around 97% at different K values. For KNN model, although it is simple to implement, the computation cost is quite high because we need to compute the distance of each data instance to all training samples when training sample is huge. The highest accuracy obtained is 97.55% in linear order feature dataset.

3.3.4 SVM Classifier

Table 3-3-4 Radial basis function (RBF) Kernel - SVM classification results

C(penalty)	0.01	0.05	0.1	0.5	1
Linear Feature	93.67%	95.95%	96.67%	97.81%	98.03%
Polynomial Feature	90.88%	95.96%	96.85%	98.04%	98.31%

Table 4 shows the SVM classification accuracies using RBF kernel for different penalty C value on linear and polynomial features. As C values increase, both accuracies on linear and polynomial features dataset increase. However, with the complexities of features in polynomial terms, training and prediction of model take much longer time than those in linear features dataset. The highest accuracy obtained is 98.31% in polynomial regression feature dataset when penalty C is 1.

Conclusion:

In this assignment, pattern recognition problem is investigated on MNIST handwritten digits. The Eigen features in low dimensions are extracted by Principal Component Analysis (PCA) method. Linear Regression and Polynomial Regression models are applied on Eigen features to solve classification problem. Eigen features contain information of all the handwritten digits in train set. Increase number of eigen features can result a better accuracy result. And higher order of polynomial regression has higher accuracy. However, higher order of polynomial regression will easily lead to the curse of dimensionality. A proper polynomial order is important to achieve optimal accuracy.

Other classifiers like KNN, MLP, Logistic Regression and SVM are implemented and investigated to improve the accuracy on test set. And all of them can achieve good accuracy near 97% to 98% on test set. Compare the complexity of all the models applied in this assignment, the polynomial regression model with pseudoinverse matrix method can achieve similar result with simple construction than other models. However, for all the model tested in this assignment, it is difficult to achieve better result above 99%.