Project 1

Haolin Sun (217805433)
sun0907@yorku.ca

November 14, 2020

Note: You need to work individually for this project. You must use this latex template to write up your report. Remember to fill in your information (name, student number, email) at above. Submit your codes/scripts (*.zip) and a project report (*.pdf) (maximum 8 pages) from eClass before the deadline. No late submission will be accepted. No handwriting is accepted. Direct your queries to Hui Jiang (hj@eecs.yorku.ca).

Pattern Classification: Features and Models

In this project, you will implement several discriminative models for pattern classification. You may choose to use any programming language for your own convenience. You are ONLY allowed to use libraries for linear algebra operations, such as matrix multiplication, matrix inversion, matrix factorization, and etc. However, You are NOT allowed to use any existing machine learning or statistics toolkits or libraries or any open-source codes for this project and you will have to implement the most parts of the model learning and testing algorithms yourself as a practice of various algorithms learned in class. This is the purpose of this project. If you have any questions, please consult the instructor for what you can or can not use for this project. You will use a popular data set, called MNIST, throughout the project. The MNIST database of handwritten digits contains 60,000 training images and 10,000 test images, and the images are 28-by-28 in size. The MNIST data set can be downloaded from http://yann.lecun.com/exdb/mnist/. In this project, for simplicity, you just use pixels as raw features for the following models.

Question 1

Feature Extraction and Data Visualization: Select three digits of '2', '3' and '5', study the effects of various dimensionality reduction methods.

(a) Use all training images of these three digits ('2', '3' and '5') to estimate principal component analysis (PCA) projection matrices, and plot the total distortion errors of these images as a function of the used PCA dimensions (such as 2, 10, 50, 100, 200, 300).

(b) Use all training images of these three digits ('2', '3' and '5') to estimate Linear Discriminant Analysis (LDA) projection matrices, what are the maximum LDA dimensions you may use in this case? Why?

(c) Use PCA and Linear Discriminant Analysis (LDA) to project all these images into 2-D space and plot them with three different colors for data visualization. Comparing these two linear methods with a popular nonlinear method, namely t-Distributed Stochastic Neighbor Embedding (t-SNE) (https://lvdmaaten.github.io/tsne/). You don't need to implement t-SNE and can directly download the t-SNE code from the website and run it on your data to compare with PCA and LDA. Based on your results, explain how these three methods differ in data visualization.

Question 2

Linear Regression and Logistic Regression: Use all training data of digits '3' and '5' to learn several binary classifiers using linear regression, logistic regression and MCE, and report the best possible classification performance in the held-out test images of these two digits. For logistic regression and MCE, you may choose to use any iterative optimization algorithm. Don't just call any off-the-shelf optimizer. You need to implement the optimizer yourself.

Question 3

Support Vector Machine: Use all training data of 10 digits to learn two 10-class classifiers (considering to use the one vs. one strategy) using linear support vector machine (SVM) and nonlinear SVM (with Gaussian RBF kernel), and report the best possible classification performance in the held-out test images. Don't just call any off-the-shelf optimizer. You need to implement the SVM optimizer yourself.

Question 4

Deep Neural Networks: Implement a general back-propagation (BP) algorithm for feed-forward fully-connected multi-layer deep neural networks. Use all training data to learn a 10-class classifier using your own BP implementation, investigate various network structures (such as different number of layers and nodes per layer), report the best possible classification performance in the heldout test images.

What to submit?

You need to submit all of your codes written for this project. Write ONE single script for all above steps for the instructor to quickly run your code and reproduce your results. Please provide a clear instruction on how to repeat your experiments in a separate readme file. You need to submit a project report (in pdf, maximum 8 pages) to summarize what you have done in terms of algorithm development and experimental fine-tuning, also report the best settings for each classifier and discuss your findings from this project.

Your Report (maximum 8 pages):

1 Background

1.1 Running environment and developing tools I used:

- Anaconda 1.10.0
- Spyder (Python3.8) 4.1.4
- Jupyter notebook 6.0.3

1.2 How to run the program:

There are several dependencies need to be installed in order to run the program, please check the **readme.md** in the folder and follow the instructions to install and see the results.

1.3 Results visualization:

The program will take time (53 mins on my local CPU) to perform a complete run, for the convenience, I also provided a Jupyter notebook file named **demo.ipynb** to let viewers quickly see the test results.

2 Observation and Discussion

2.1 Question 1:

2.1.1 PCA in MNIST:

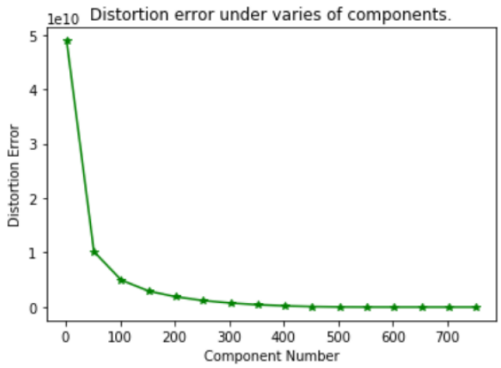

Figure 1: PCA

Figure 1 shows the distortion error in different dimensions. We can easily find that when the dimension is 100, we can restore the original image well properly.

2.1.2 LDA in MNIST:

In LDA, because we used three different digits, the maximum dimension of LDA is 2. The reason is that the between class scatter matrix S_b does not have full rank. It is derived from only K different class-dependent mean vectors and we can verify its rank cannot exceed K - 1. As a result, the rank of the matrix $S_{\overline{w}}1S_b$ does not exceed K - 1 as well, and this in LDA we can only derive at most K - 1 mutually orthogonal eigenvectors corresponding to K - 1 non-zero eigenvalues. (quote from the textbook)

2.1.3 t-SNE in MNIST:

Figures down below shows the projection results of PCA, LDA and t-SNE algorithms on the 2D plane. Compared with PCA, both LDA and t-SNE need to use label information during the dimension reduction. Therefore, the space after the dimension reduction of LDA and t-SNE using different types of data looks more scattered.

However, PCA uses the unsupervised dimension reduction, so the variance after the reduction is large, and the categories do not seem to be well scattered.

Also, it should be noted that due to the slow speed of the python version t-SNE, this project only randomly selected 500 samples of each digit, for a total of 1500 samples.

2.1.4 Projection results of PCA, LDA and t-SNE:

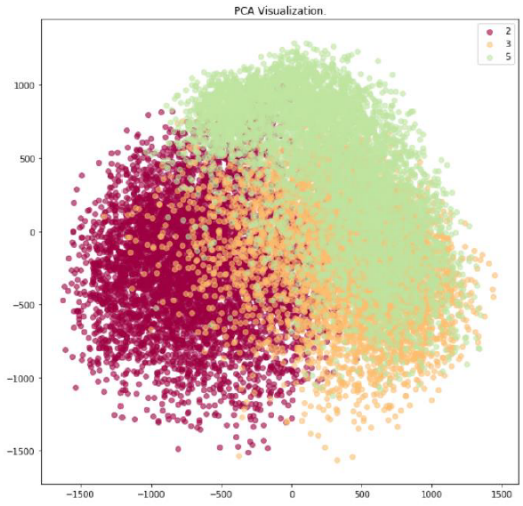

Figure 2: PCA

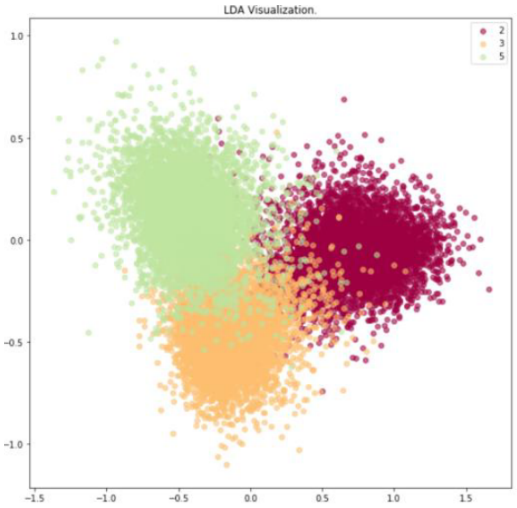

Figure 3: LDA

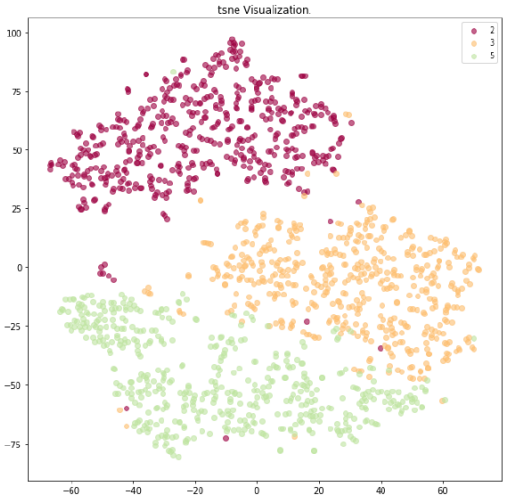

Figure 4: t-SNE

2.2 Question 2:

By running the linear regression equation, we found that linear regression has analytical solutions, while logistic regression and MCE need to be solved iteratively, so the running speed of linear regression is the fastest among the three algorithms.

The batch gradient descent algorithm is used to update the parameters in the experimental process of logistic regression and MCE. The learning rate of both is 0.00001, and the maximum number of iterations is 300.

Table 1 below shows the accuracy of the three algorithms on the test set. We can find that linear regression has the best accuracy and operating efficiency.

Table 1: Test results of three algorithms

Algorithm	Test Accuracy
Linear Regression	0.9632
Logic Regression	0.9548
MCE	0.9632

Figure 3 shows the training loss function values of logistic regression and MCE. It can be found that MCE is more stable in training, while logistic regression converges slightly faster.

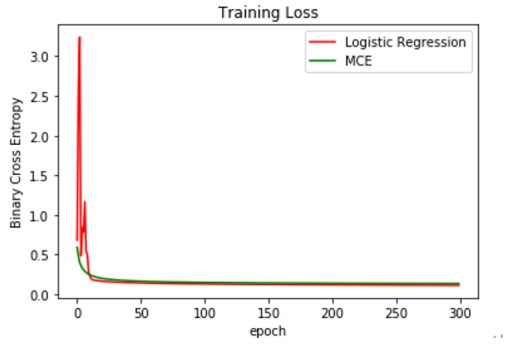

Figure 5: Training loss of LR and MCE

2.3 Question 3:

In order to verify whether the SVM we implemented can work properly, we manually generated the binary classification data. Figure 6 and Figure 7 show the decision boundary and the support vectors. Figure 6 uses the Gaussian RBF kernel, Figure 7 uses the linear kernel, and the dots indicate the support vectors. It is not difficult to see that the SVM using the RBF core and the linear core can both divide the data correctly.

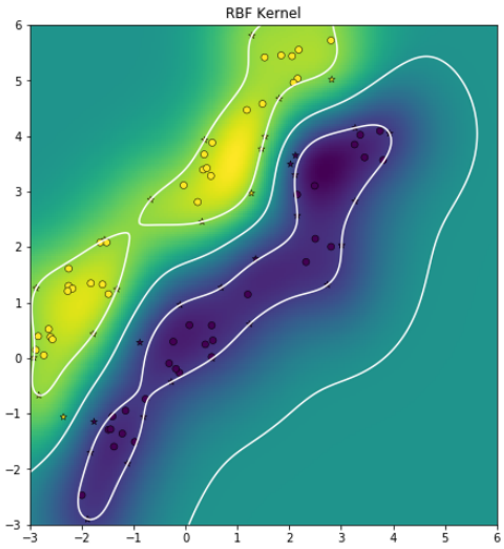

Figure 6: SVM with RBF kernel

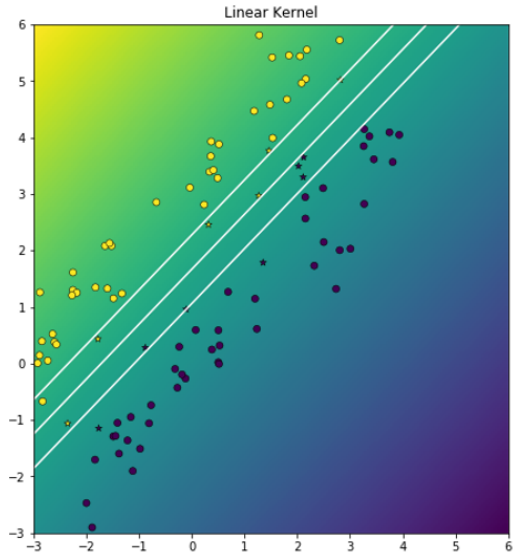

Figure 7: SVM with linear kernel

In the experiment, the SMO algorithm was used to solve the optimal solution of the quadratic convex optimization. The classification accuracy of a single RBF SVM on the numbers 3 and 5 was 0.9937, and the convergence time was about 17 minutes.

However, when only 1000 samples are used for each class, the accuracy on the test set is 0.9853, and the convergence time is about 10 seconds. We can see that the analysis accuracy is slightly reduced in the case of a small number of samples, but the speed is greatly improved.

Therefore, in order to improve the running speed, we only use 1000 samples for training. The penalty function C in RBF SVM and Linear SVM Both are set to 1, the sigma in RBF is calculated using $1/(N * var(X))$, where N represents the number of samples, and var(X) represents the variance of the entire sample.

Table 2 below shows the results of RBF SVM and Linear SVM on the multi-type test set. It can be found that RBF SVM has the highest accuracy.

Table 2: Test results of SVM

Algorithm	Test Accuracy
Gaussian RBF SVM	0.9625
Linear SVM	0.9111

2.4 Question 4:

In this question, we have implemented a total of three multi-layer perceptrons with activation functions, namely sigmoid, relu and tanh activation functions. We use the SGD optimizer to update the parameters.

We first tested the convergence of the three activation functions, the learning rate is 0.1, and the coefficient of the L2 penalty is 0.0001.

Figure 8 shows the loss function of the training process. Among them, the relu activation function was abnormal during the training process, and the gradient explosion appeared on the loss function.

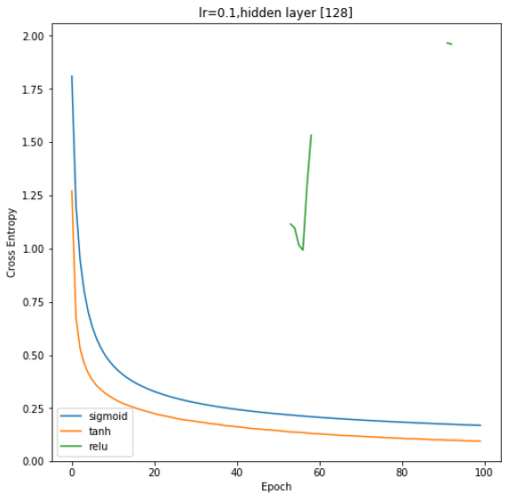

Figure 8: Loss function

Secondly, we compared the influence of different hidden layers on the network. Figure 9 shows the corresponding training process.Please see next page to see the influence of different hidden layers.

In addition, we also found that the learning rate used by different activation functions should not be the same. Base on many times of experiment, it is more appropriate to set the learning rate of sigmoid to 0.1, the accuracy of the test set is 0.9687. The learning rate of relu is set to 0.0001, and the accuracy 0.7331.

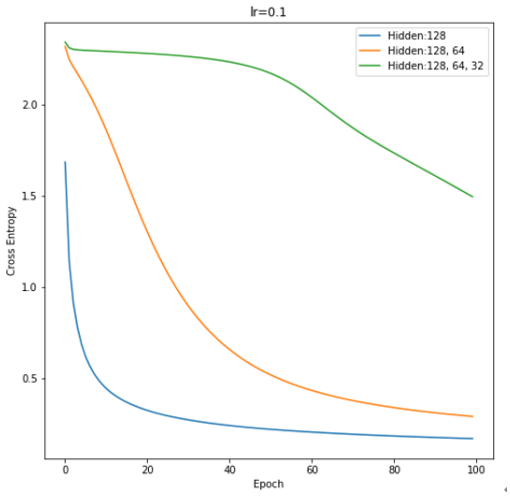

Figure 9: Different Hidden Layers