EECS 6327
Project One Report
Winter 2018

Farzaneh Heidari

March 13, 2018

Instructor: Professor Hui Jiang

1 Introduction

The purpose of this project is implementation of basic classification and dimensionality reduction algorithms in machine learning. In question one, Principal Component Analysis (PCA) and Linear Discriminant Analysis (LDA), in question two, Linear regression and Logistic Regression, in question three, Linear and non-linear Support Vector Machine (SVM) and in question four simple one layer Neural Network have been implemented.
The project's code is written in Python and algebra operations are performed using Numpy library.

1.1 Dataset

In this Project, all the classification tasks are performed on the modified National Institute of Standards and Technology (MNIST) dataset of handwritten images which is highly used for image processing tasks. This dataset contains 60000 training images and 10000 testing images. Images are $28-\text{by}-28$ in size. First step in this project is reading this dataset. In all of our classification and dimensionality reduction tasks, we read this dataset in a way that each image will be represented by a one dimensional vector with size of 784. Also, for each entry, there is a label array which refers to the true label (handwritten digit).

2 Feature Extraction and Data Visualization

In this question, we study effect of various dimensionality reduction tasks on a subset of the dataset consist of three digits '4', '7' and '8'.

2.1 PCA

The goal of the Principal Component Analysis is to find the most prominent dimensions of a data and re-express the complex data in its most meaningful basis. PCA uses an orthogonal linear transformation and maps the data to a new coordinate system which maximum variance of the data lies on the first component (Principal component) and second greatest variance on the second coordinate and so on.Shlens (2014)

2.1.1 PCA Results

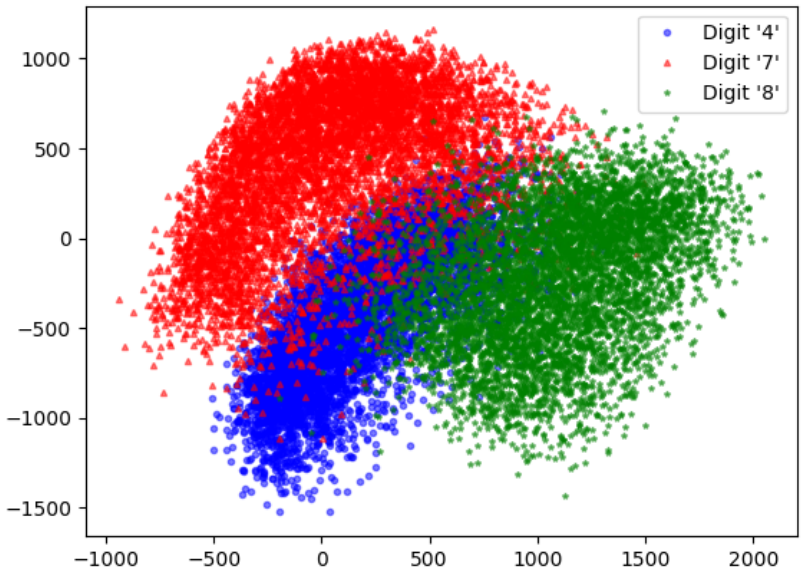

Figure 1: PCA projection on three digits 4, 7 and 8. Even though PCA is a linear method, it almost separates three classes of data finely.

MNIST images have 784 dimensions. We project these images to their two first principal components. In figure 2.1.1 we see that PCA by projecting maximum variance of the data on the x-axis, is almost able to separate these three digits. PCA is a linear method and can only diminish effect of linear correlation of the observations by mapping data to linearly uncorrelated coordinates. As it is obvious in 2.1.1, data is not completely separated because of the nonlinear correlations between the classes. Total distortion error of the PCA as function of its dimensions reduces as dimensions get larger 2.1.1. We expect these results because larger dimensions preserve more structure of the data thus result in

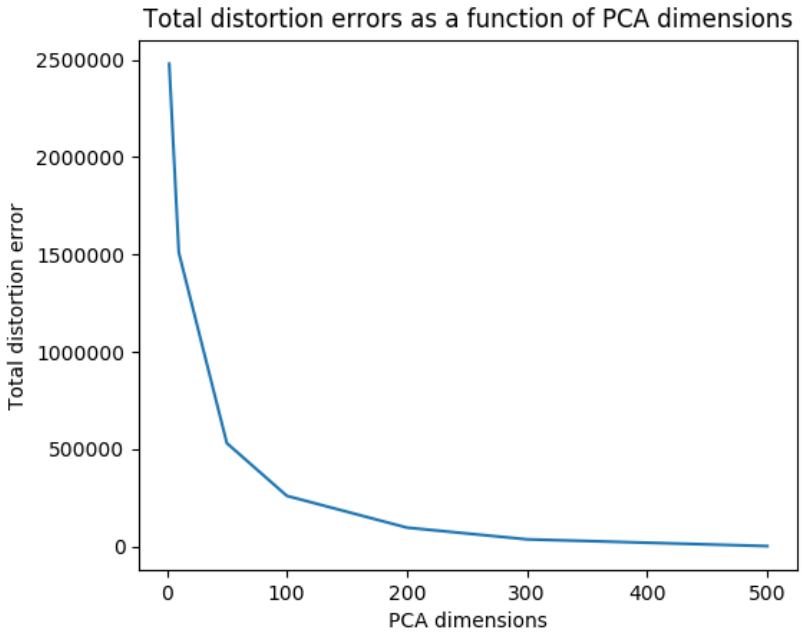

Figure 2: PCA total distortion error of projection on three digits 4, 7 and 8 as a function of PCA dimensions.

lower distortion errors. reconstruction or distortion error of PCA is defined as :

$$\sum_{i=1}^{n} ||x_i - x_i'||_2^2$$

where x_i' is inverse mapping of x_i to the original dimension. We can simplify the expression above in a few step and get to:

$$Distortion_k = n \sum_{j=K+1}^{m} \lambda_j$$

where λ_j is the eigenvalue of unselected eigenvectors as axises in PCA.

2.2 LDA

LDA is a supervised dimensionality reduction that maximizes the class separation. LDA explicitly attempts to model the difference between the classes of data.

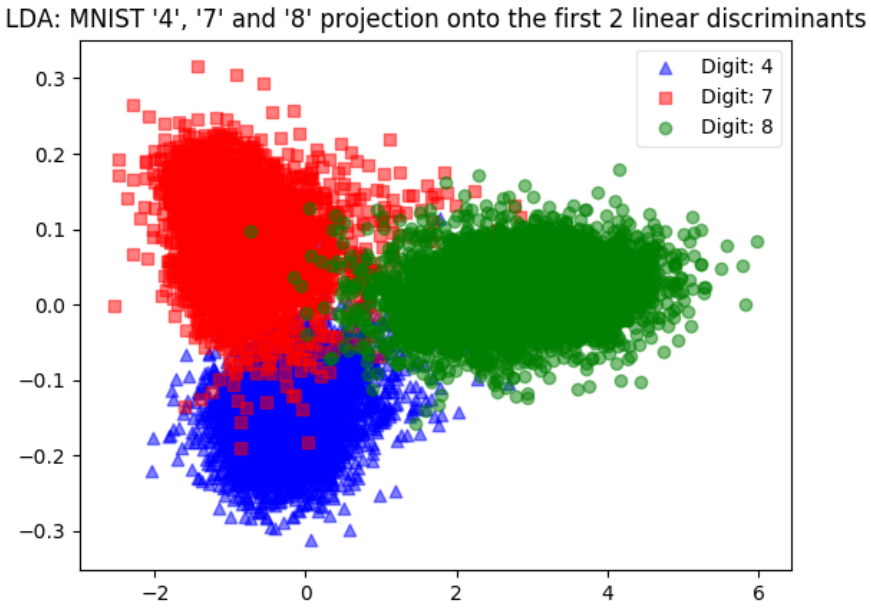

Figure 3: LDA projection on three digits 4, 7 and 8. LDA separates data slightly better than PCA,

2.3 t-SNE

t-Distributed Stochastic Neighbor Embedding (t-SNE) is a nonlinear for dimensionality reduction which is a stochastic local mapping method.

2.4 Comparison

In 2.1.1, 2.2 and 2.3 we see figures of three different dimensionality reduction techniques (PCA, LDA, t-SNE). As we see in the figures t-SNE has the best performance in separating and visualizing the data according to its nonlinearity and LDA and PCA are almost the same.

3 Linear and Logistic Regression

In this section, we train two 10-class classifier using Logistic and Linear Regression and perform it on the held-out MNIST test images. To extend, Regressions to multi-class classifiers, we use one-vs-rest strategy which involves training a single classifier per class, with the samples of that class as positive samples and all other samples as negatives. Accuracy results are summarized in table6. In

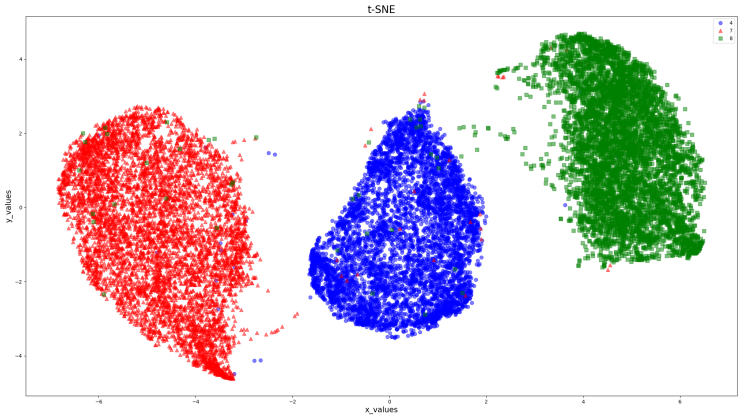

Figure 4: t-SNE visualization of three digits of MNIST dataset

the Logistic Regression, number of iterations for gradient ascent is 3000 and learning rate is $5(10^{-5})$.

3.1 Linear Regression

Assume X is a 60000×784 matrix of our observations. Each image is a 784 one-dimensional horizontal vector. Using one-vs-rest strategy each time we will construct a label matrix Y that is a one-dimensional vertical matrix and only entries of one specific label (digit) is one and rest of the entries zero. From two class Linear Regression we have:

$$Y = Xw^T$$

and

$$w^* = \arg\ min_w \sum_i (x_i w^T - y_i)^2$$

$$w^* = (X^T X))^{-1} X^T Y$$

Linear Regression has a closed form solution.

3.2 one-vs-rest

We found w^* for any digit (10 digits) and inner product each observation two these ten w^* classifiers and find a $score_i$. The w_i^* with maximum score is that observations defines the label (i) of that observation. Matrix Inversion in Linear Regression is time consuming.

3.3 Logistic Regression

Like Linear Regression, Logistic Regression is a binary classifier that maximizes the likelihood of an observation. Logistic Regression measure the relationship between dependent variable and independent variable using a logistic function. Here, we have used , sigmoid function as the link function:

$$\sigma(x) = \frac{1}{1 + e^{-t}}$$

Unlike Linear Regression, Logistic Regression doesn't have a closed form solution thus we use a gradient ascent algorithm to maximize the log-likelihood function.

$$LogLikelihood = \sum_{i=1}^{N} y_i \beta^T x_i - \log(1 + e^{\beta^T x_i})$$

and gradient step is to maximize log-likelihood is:

$$\Delta ll = X^T(Y - Predictions)$$

where X is training data and Y is the labels.

4 SVM

The goal of SVM is finding a a decision boundary H, that is as far away from the data of both classes as possible. The decision boundary can be found by solving he following constrained optimization problem.

$$\min \frac{1}{2}||w||^2$$

$$||w||^2 = w^T w$$

subject to: $y_i(w^T x_i + b) \geq 1 \ \forall i$. The dual problem converts to:

$$\max W(\alpha) = \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j f(x_i)^T f(x_j)$$

subject to $0 \leq \alpha_i \leq C$ and $\sum_{i=1}^{n} \alpha_i y_i = 0$

$$\Phi(x_i, x_j) = f(x_i)^T f(x_j)$$

where:

$$\Phi(x_i, x_j) = \exp(\gamma ||x_i - x_j||^2)$$

For implementing SVM, in this project Pegasos algorithm is used Shalev-Shwartz et al. (2011). In this algorithm for mini-batch gradient descent updates is used and sub-gradients of the approximate objective is given by:

$$\Delta_t = \lambda w_t - \frac{1}{k} \sum_{i \in A_t} \mathbf{1}[y_i \langle w_t, x_t \rangle < 1] y_i x_i$$

In the kernalized pegasos algorithm, only kernel evaluations without direct access to the feature vectors $\Phi(x)$ or explicit access to the weight vector w. The algorithms and proof is stated in Shalev-Shwartz et al. (2011).

$$w_{t+1} = \frac{1}{\lambda t} \sum_{i=1}^{t} \mathbf{1}[y_i \langle w_t, \Phi(x_{i_t}) \rangle < 1] y_i \Phi(x_{i_t})$$

Like Regression, as mentioned in the previous section, multi-class classifier is a one-vs-rest multiple binary classifiers. For Linear SVM, mini-batch of size $k = 600$ and $iterations = 3000$ and learning rate $\frac{1}{60000}$ is used. Results are summarized in the table 6.

5 Neural Network

For 10-class classification of MNIST dataset, in this project, a Neural Network with one hidden layer of 800 nodes is used. Input layer consists of 784 nodes, hidden layer, 800 nodes with sigmoid activation function , and the output layer 10 nodes with Softmax activations. As we are classifying a 10-class dataset, Cross-Entropy loss is used. Best performance is in learning rate of 0.01 and 2000 epochs. For optmization Stochastic Gradient Descent (SGD) is used.

6 Classification Accuracy

Below is the classification accuracy on for different classifiers.

Linear Regression	Logistic Regression	Linear SVM	RBF SVM	Neural Network
0.86	0.89	0.80	0.88	0.92

References

Shalev-Shwartz, S., Singer, Y., Srebro, N., and Cotter, A. (2011). Pegasos: primal estimated sub-gradient solver for svm. *Mathematical Programming*, 127(1):3–30.

Shlens, J. (2014). A tutorial on principal component analysis. *CoRR*, abs/1404.1100.