PROJECT 1

HAO LI

213312137

PART 1 INSTRUCTIONS FOR PROGRAM

PCA, LDA and TSNE

For PCA and LDA, please go into the "project/source" directory and **type**

python pca_lda.py

The program takes about 87 seconds to finish and output what the program is doing step by step, the related figure will be plotted one by one by matplot.

For TSNE, I do not recommend to run the program, it takes several hours to finish, the figure plotted by tsne will be pasted latter. If you want to run the program, please go into the "project/source" directory and **type**

python tsne.py

LINEAR REGRESSION

Go to the "project/source" directory, **type**

python Linear_regression.py

The solution computed by normal equation is pretty quick, can view the solution only in less a second.

LOGISTIC REGRESSION

Go to the "project/source" directory, **type**

python Logistic_regression.py --alpha 0.01 --epochs 10

You can define your own parameter set

SVM

Go to the "project/source" directory, **type**

python multiSMO.py --num_class 10 --kernel_type Gaussian_kernel --sample_data 1 This program also takes a long time and much memory to run.

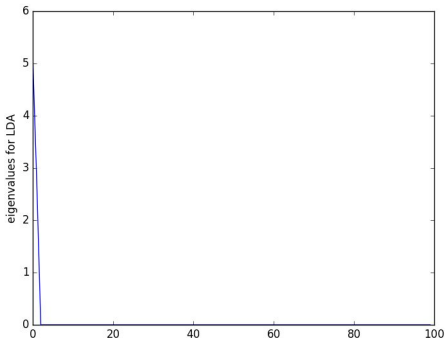
Neural Network

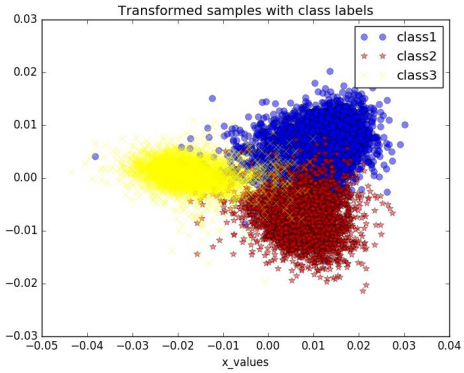
Go to the "project/source" directory, **type (**you can define your own parameter set **)**

python neural_network.py --size [784,100,10] --epoch 10 --mini_batch_size 10 --eta 0.01 --lmd 0.001

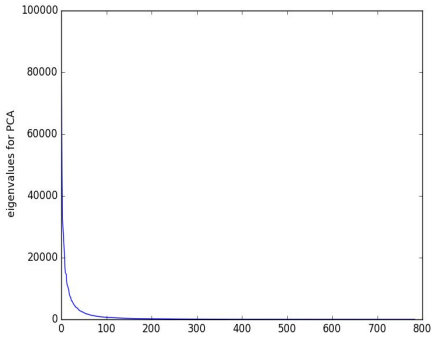
PART 2 ANALYSIS FROM MODELS

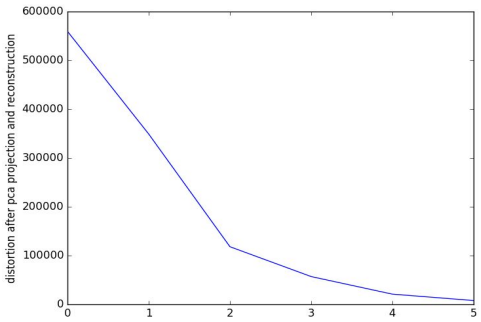
FIGURES FOR LDA (eigenvalues left, projected data right)
We can see LDA gives better classification compared to PCA.

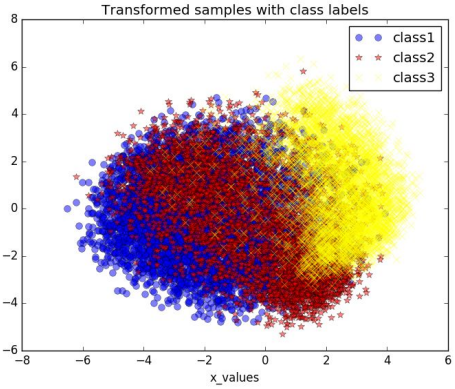


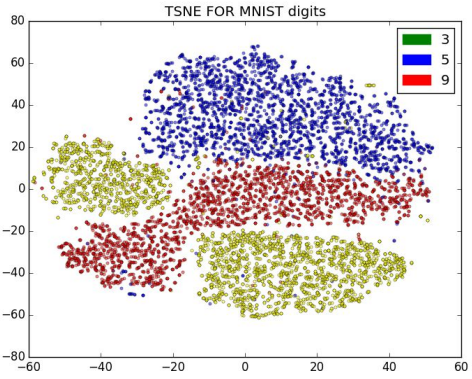


FIGURES FOR PCA and T-SNE (eigenvalues left up, distortion right up, PCA projected data left bottom, tsne right bottom)
As we can see in the distortion figure, as we keep more dimensions, the distortion is going down which indicates we kept more information of the original.









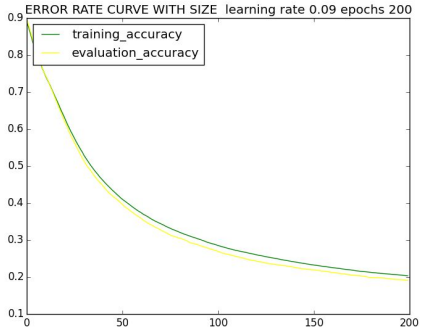
LINEAR REGRESSION

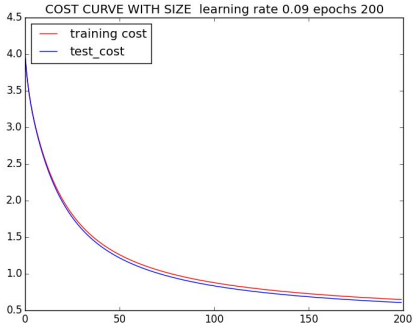
The accuracy computed by normal equation is 0.8606, in the experiment, 50000 sample data is selected and used in the normal equation $(X^T X)^{-1} X^T Y$ to compute the parameter Θ, and 10000 sample data is selected for testing the accuracy. Experiment with gradient descent is also done but normal equation is much faster.

LOGISTIC REGRESSION

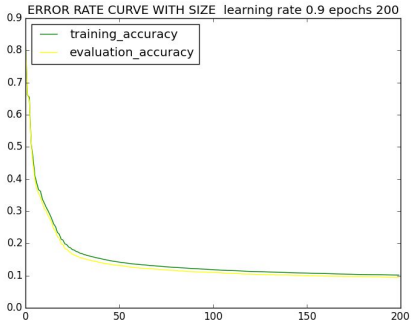
In this experiment, batch gradient descent is used to compute the gradient for each epoch, each time 50000 data samples is multiplied by the weight matrix and applied softmax followed by cross entropy to compute the loss. The model is experimented with three different learning rate, 1.0, 0.1 and 0.01. The following are the training error rate, test error rate, training cost and test cost. As we can observe in the figure, the error ratio for the model with learning rate 0.1 decrease slower than the model with learning rate 1.0 and only reaches about 80 % accuracy, on the other hand the error ratio for the model with learning rate 1.0 converges to 91% accuracy pretty fast. For the last model with learning rate 0.01, the error ratio hardly decreases, it goes to slow.

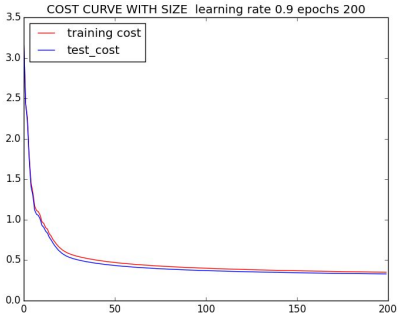
1. Learning rate 0.1



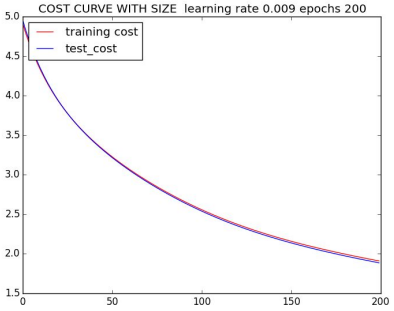


2. Learning rate 1.0





3. Learning rate 0.01



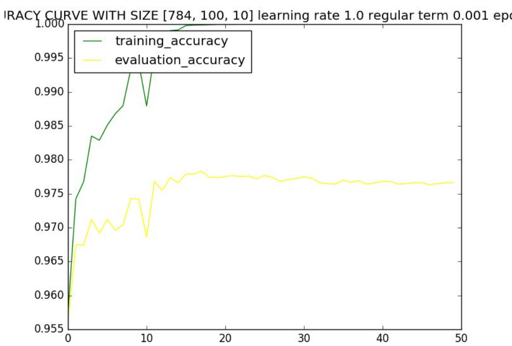
SVM

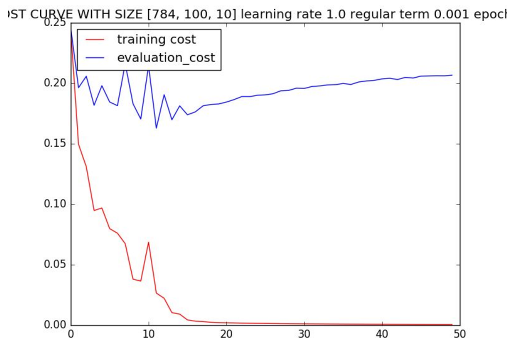
In SVM, sequential minimal optimization (SMO) is used for the purpose of optimizing the alphas. One vs One method is used for multi-class classification. In order to make computation fast, a catch is used in the smo model which stores all the Kernel values for each data, for example, when the program needs the value $x_i.dot(X)$, it can directly sum the ith row of the Kernel catch for that value instead compute the value many times, for 2 class classification, the catch can be used without no problem, since only about 10000 data sample is used, this leads to the 10000 x 10000 real numbers needs to be stored. It also works for up to 5 class classification where 12 classifier is trained therefore 12 x 10000 x 10000 real numbers need to be stored. However training more than 5 classifier uses memory beyond the capacity of the speech1, so only part of the data samples where 1000 samples is selected from each class is used for training. For 2 - 5 class classification, the svm gives above 99 % accuracy and for 10 class classification, it achieves around 95 % accuracy. If you have enough memory you can set the --sample_data to be false, pass 0 to argument --sample_data.

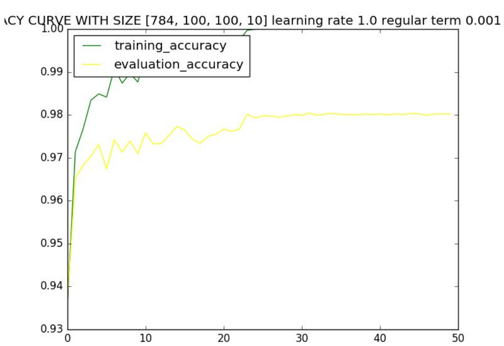
NEURAL NETWORK

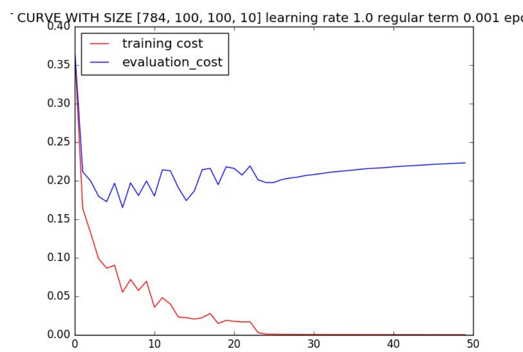
Different structures of network and sets of parameters are experimented.The three structures are [784,100,10], [784,100,100,10] and [784,100,100,100,10] and sets of the parameters are {eta:1.0,lmd=0.001},{eta:0.1,lmd=0.001} and {eta:0.1,lmd=0.0001}

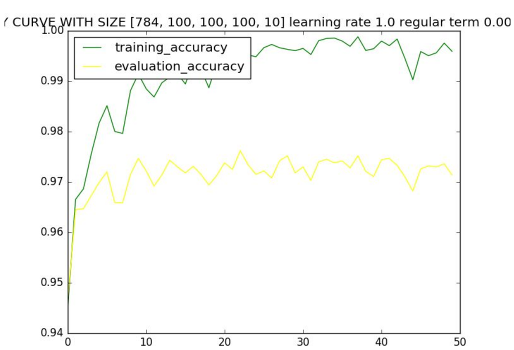
1. Figures from [784,100,10] and {eta:1.0,lmd:0.001}, [784,100,100,10] {eta:1.0,lmd:0.001} and [784,100,100,100,10] {eta:1.0,lmd:0.001} as we can see in the picture, overfitting happens in all the experiment and both accuracy and cost fluctuates severely. Another thing is as we use more and more layer the accuracy first becomes better and smoother but then becomes worse and less smoother.

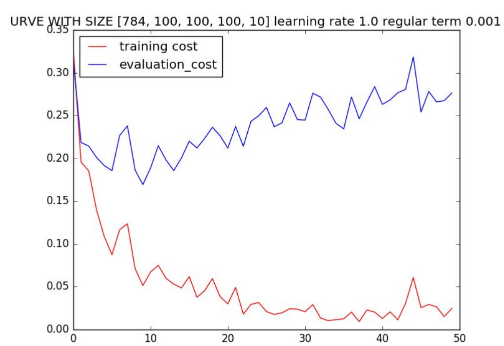




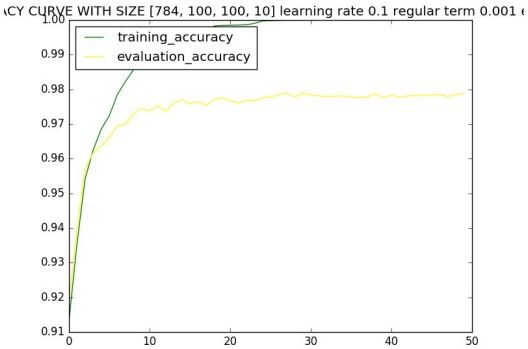


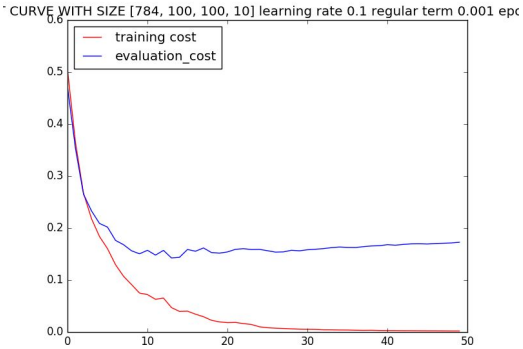






2. Figures from [784,100,100,10] {eta:0.1,lmd:0.001} as we can see in the picture, overfitting also happens here but the curve is smoother than the one with eta 1.0. Accuracy for the evaluation set is almost always increasing.





3. Figures from [784,100,100,10] {eta:0.1,lmd:0.0001} as we can see in the picture, the figure showed the overfitting is less severe than previous sets of parameters.