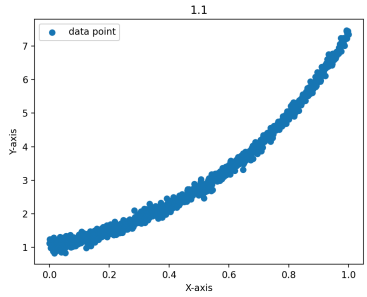
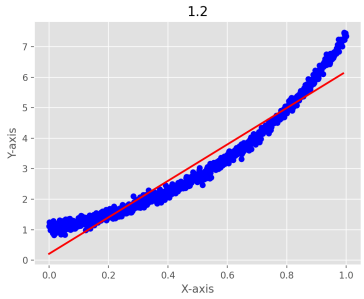
The code for HW1 is separated into question_1.py、question_2.py、question_3.py and question_4.py to represent each question in HW1. I have implement the machine learning models by myself in model.py for question_1.py、question_2.py and question_3.py to use . I write the pca calculation of Question_4 directly in question_4.py

1

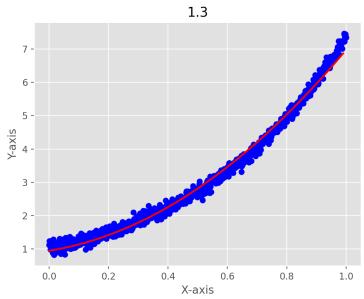
1.1 plot the data from 'data.mat'



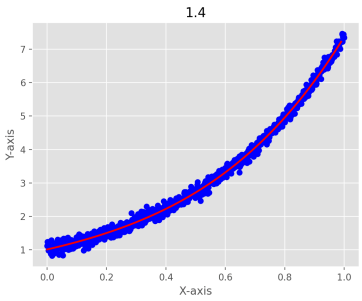
1.2 line y = $\theta_0 + x\theta_1$ overlay the line on the given data.



1.3 parabola y = $\theta_0 + x\theta_1 + x^2\theta_2$ overlay the parabola on the given data.



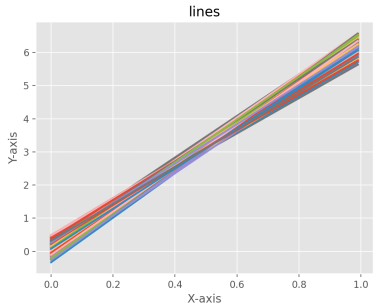
1.4 quartic curve y = $\theta_0 + x\theta_1 + x^2\theta_2 + x^3\theta_3 + x^4\theta_4$ overlay the quartic curve on the given data.



1.5 The MSE for the linear model is 0.102, for the parabolic model (quadratic) is 0.012, and for the quartic model (cubic curve) is 0.005. Therefore, the quartic model is more suitable for this dataset. This could be attributed to the dataset's distribution, which exhibits a curved pattern, and the quartic model allows for the most flexibility to capture this curvature.

2

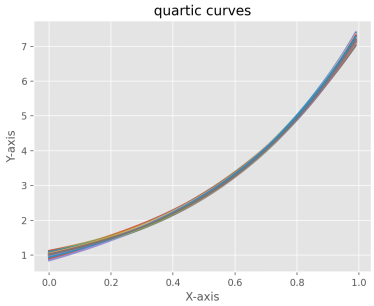
Randomly select 30 data samples for 200 times and plot these 200 lines



Line variance: 0.017394656591636345

Line bias : 0.3670494073395266

Randomly select 30 data samples for 200 times and plot these 200 quartic curves



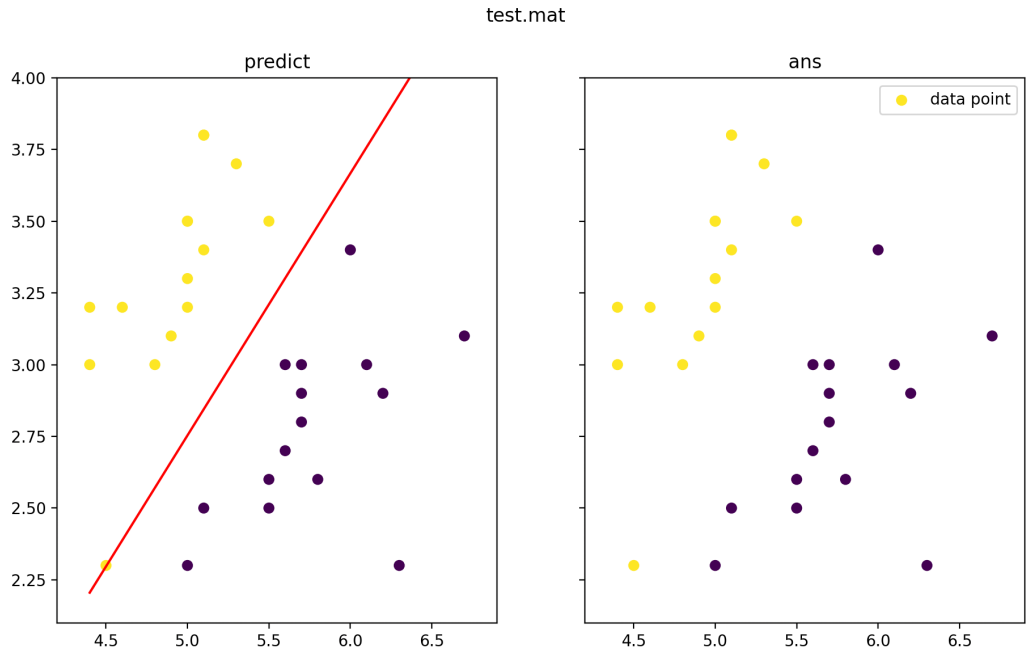
Quartic curves variance: 0.0009222344947764466

Quartic curves bias: 0.07709141158208305

The variance of line models is 0.0174, the bias is 0.367, and the variance of quartic models is 0.0009, while the bias is 0.077. It can be seen from the visually drawn pictures that the lines drawn by quartic models are closer than the line models, and it can also be seen from the data The variance of quartic curves is lower than the variance of Line. Quartic models have lower bias than line models because they are flexible and can fit complex data patterns

3

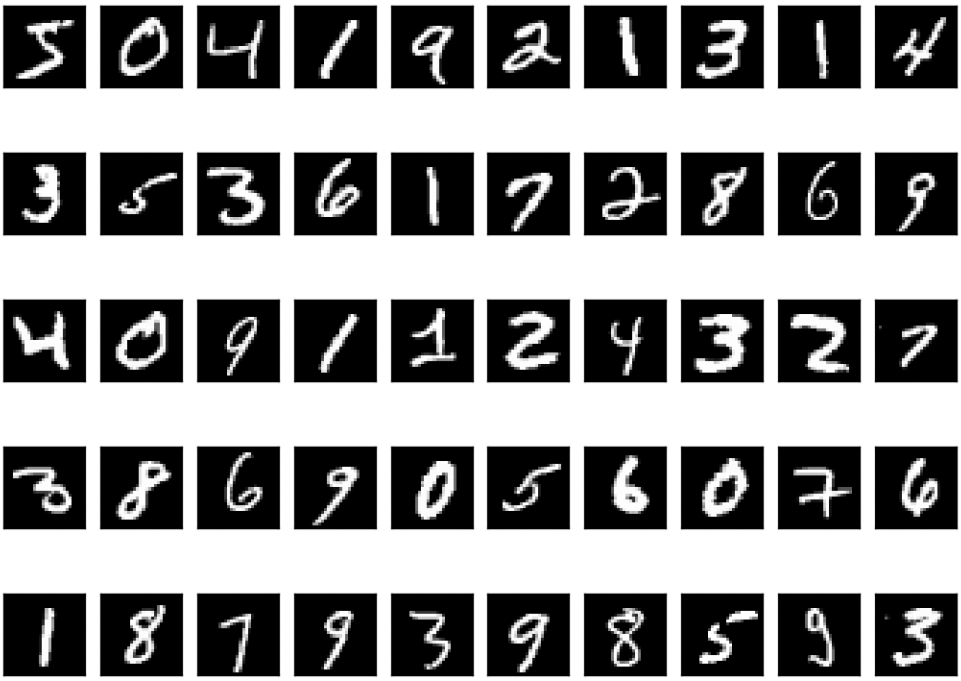
Use logistic regression to find the decision boundary



Test Error: 0.00%

4

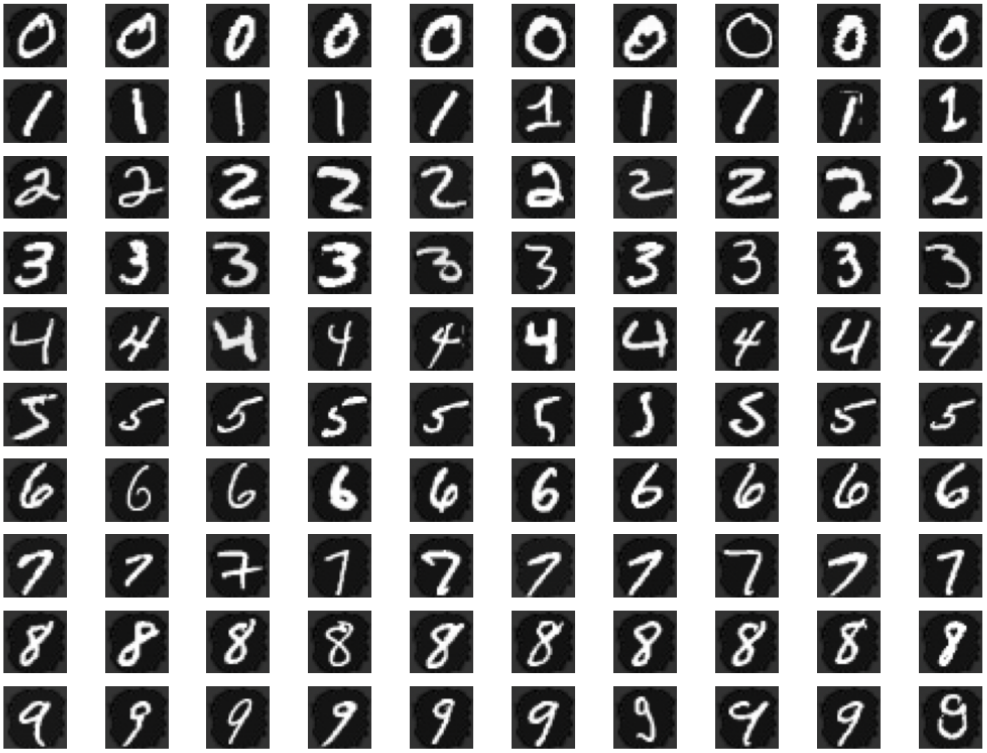
4.1 Use the following code to show 50 images in your own dataset.



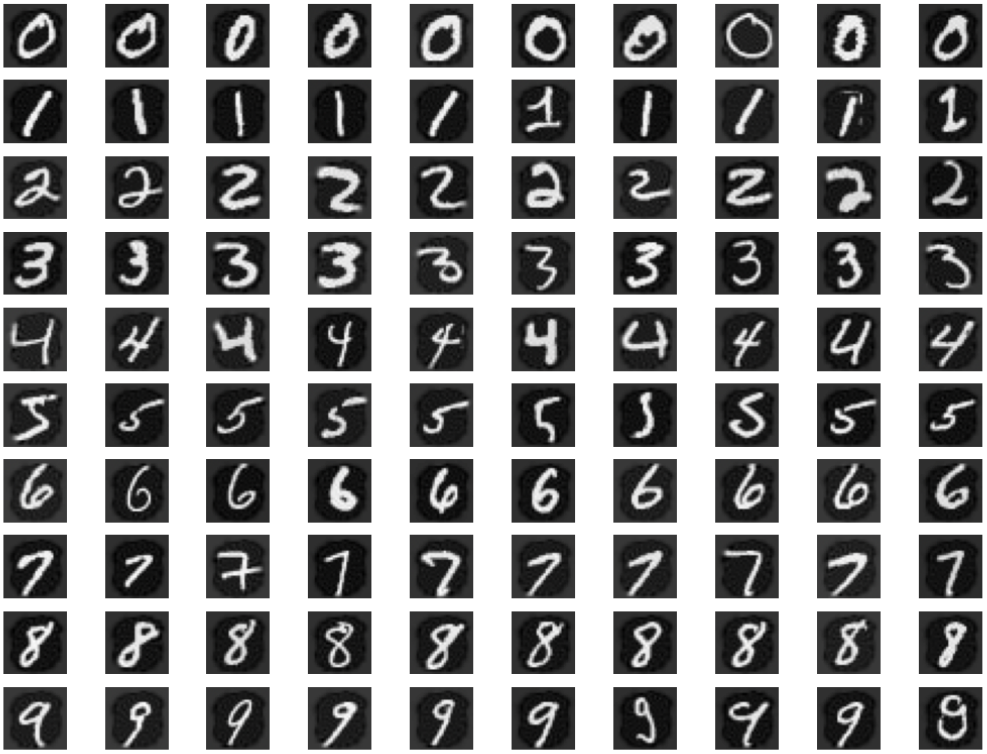
4.2 I have calculated the eigenpairs for the covariance of the data, sorted in a descending order based on eigenvalues, which is documented in question_4.py. However, due to the large amount of data, it was not possible to capture a complete screenshot. The subsequent PCA in section 4.3 is based on the computed results from section 4.2 eigenvectors.

4.3 Use PCA to reduce the 784 dimensional data to that with 500, 300, 100, and 50 dimensions, and then show 10 decoding results for each digit, respectively.

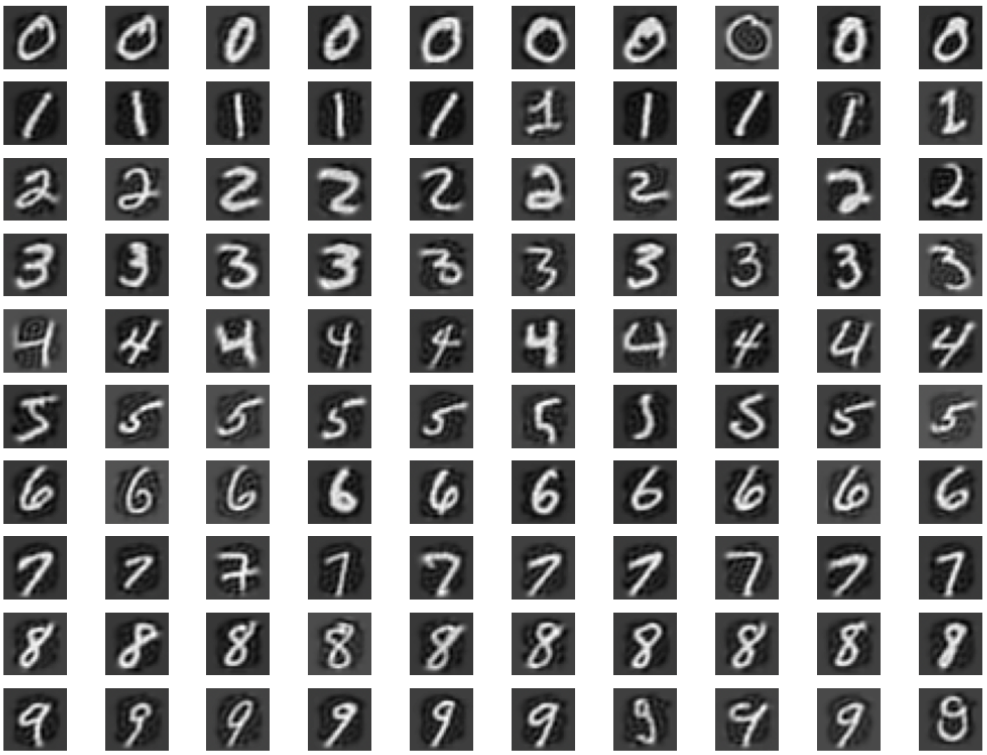
500 Dimensions



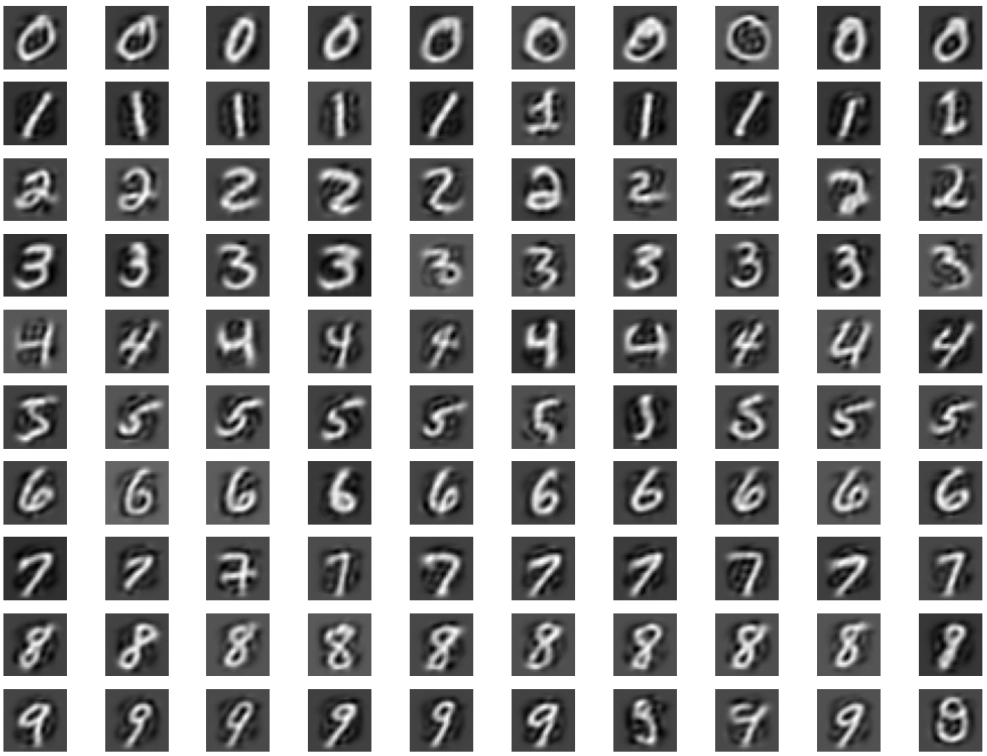
300 Dimensions



100 Dimensions



50 Dimensions



From the above images, it can be observed that while the overall integrity of the images is preserved, it is also evident from the visuals that as more dimensionality reduction is applied through PCA, more information is lost. As a result, the decoded images appear relatively more blurred in the visualization.