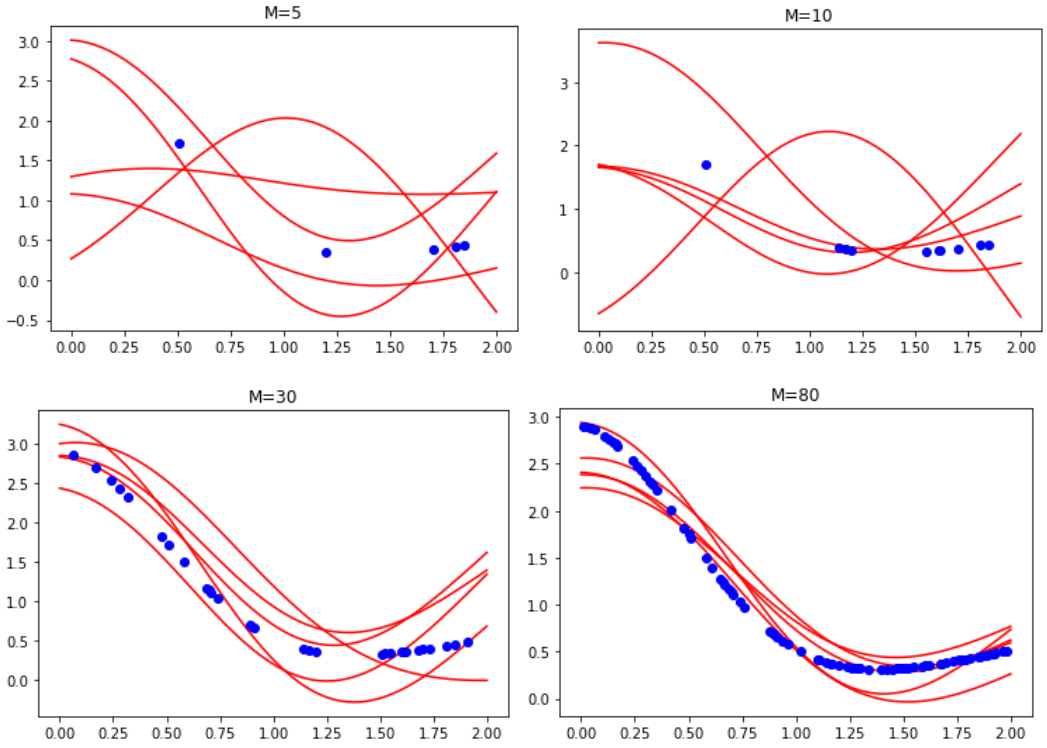
HW2 Report

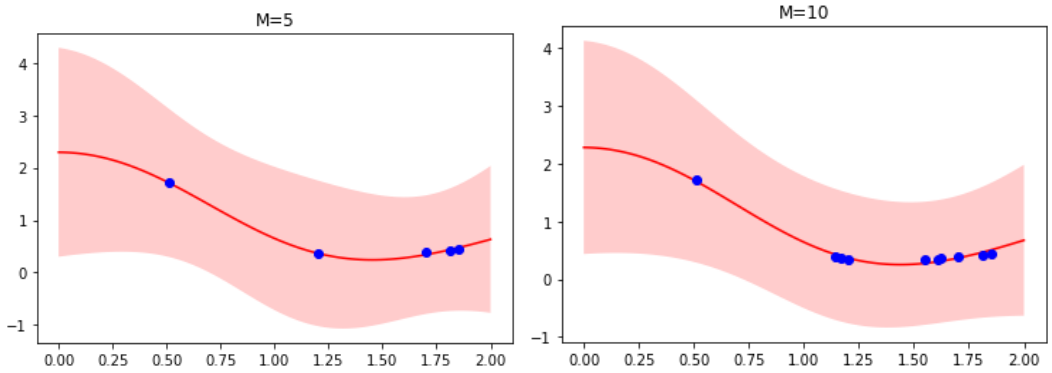
309555025 羅文笙

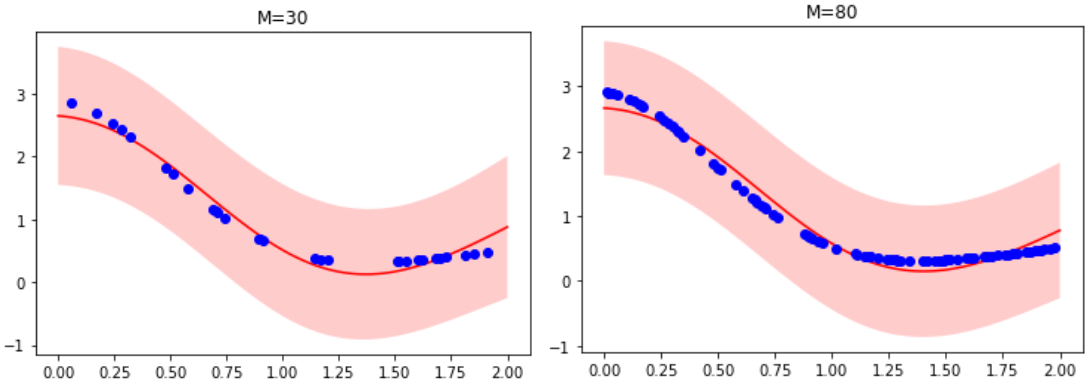
1. Sequential Bayesian Learning

1-1

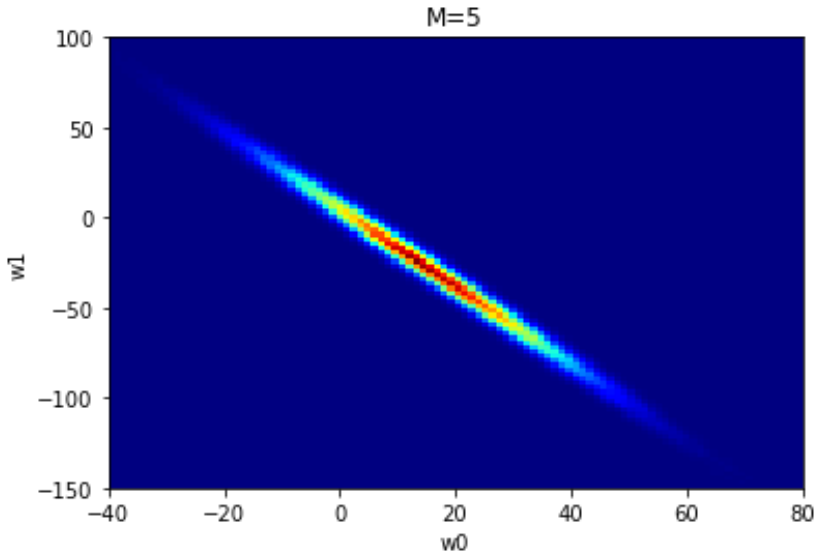


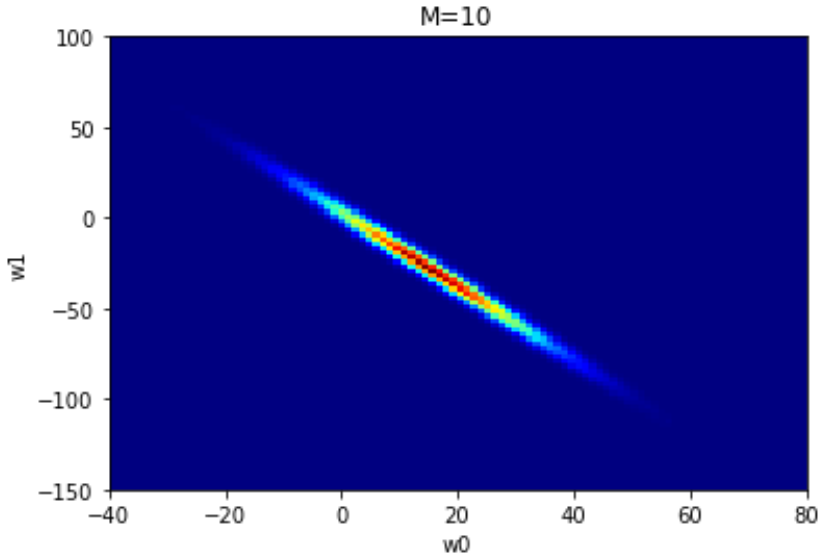
1-2

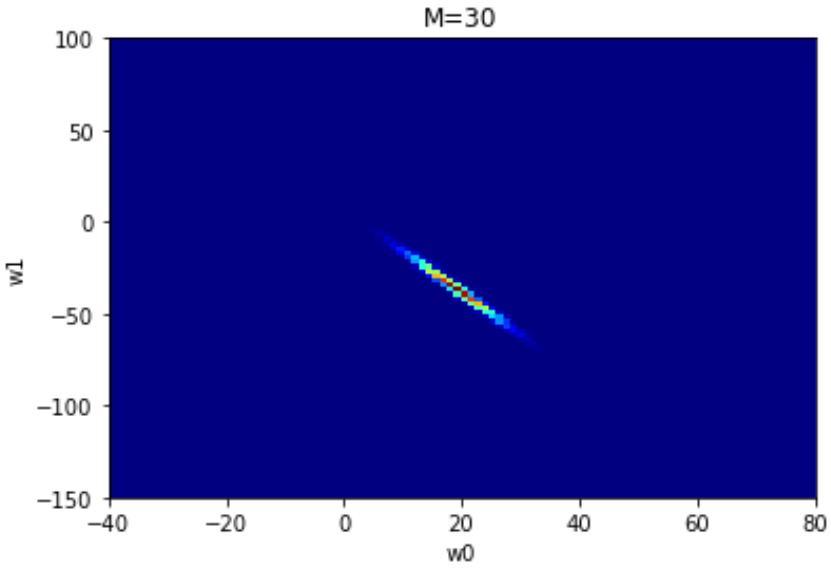


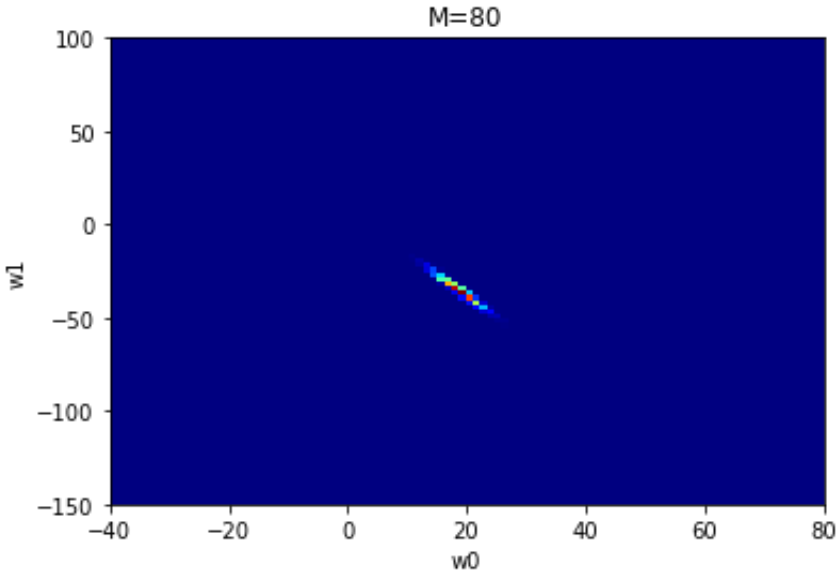

M=30
M=80
3
2
1
0
−1
0.00
0.25
0.50
0.75
1.00
1.25
1.50
1.75
2.00

1-3


M=5
100
50
0
−50
−100
−150
w1
−40
−20
0
20
40
60
80
w0


M=10
100
50
0
−50
−100
−150
w1
−40
−20
0
20
40
60
80
w0



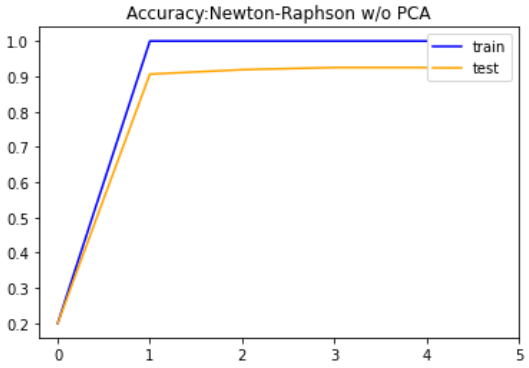


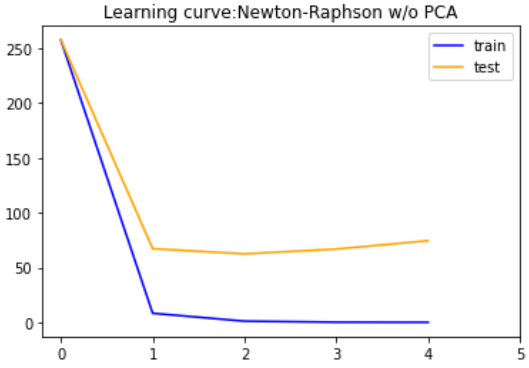
1-4 Discussion

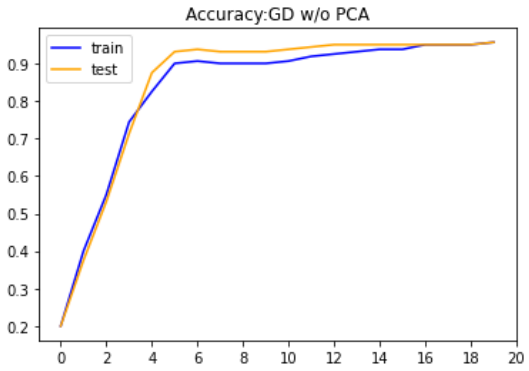
While increasing N, we can notice that our model becomes more and more fitting to the true distribution as the result shown in question 1-1, and has smaller variance as the result shown in question 1-2. Also, in question 1-3, it shows that the region of sampling 'w' gets smaller and smaller. In conclusion, all these results indicate Bayesian learning method really learn from the training data sequentially by increasing N.

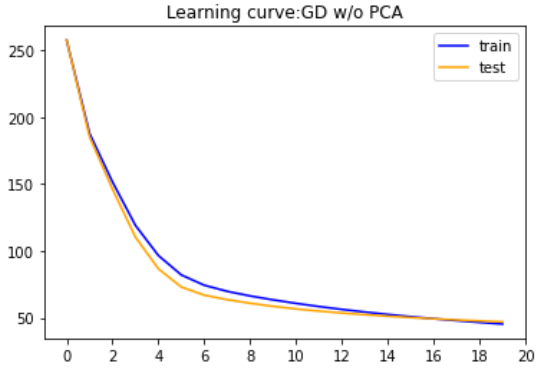
2. Logistic Regression

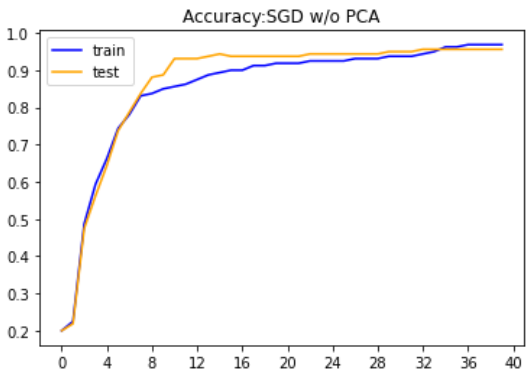
2-1 (a)

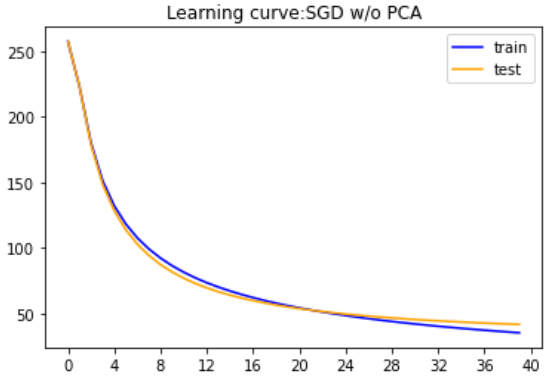


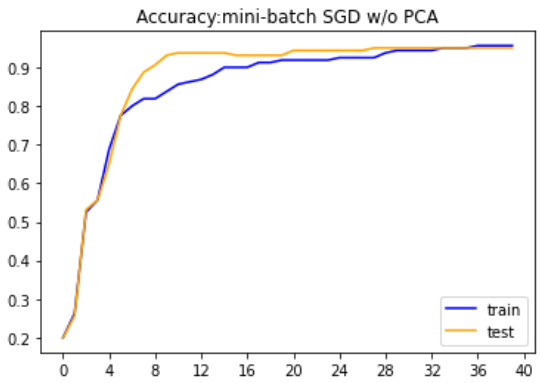


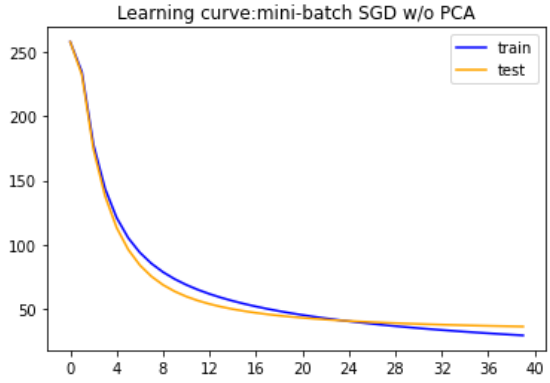












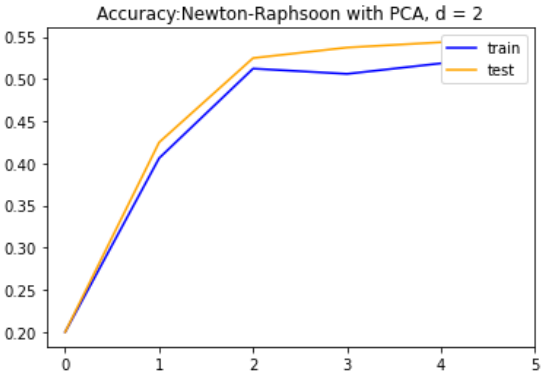
2-1(b)

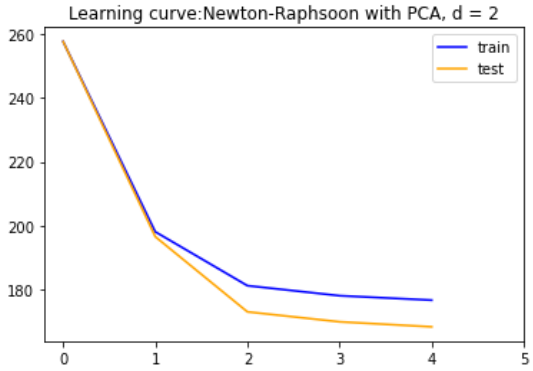
	Newton	GD	SGD	Mini-batch
Training accuracy	1.0	0.9526	0.96875	0.95625
Test accuracy	0.925	0.9526	0.95625	0.95

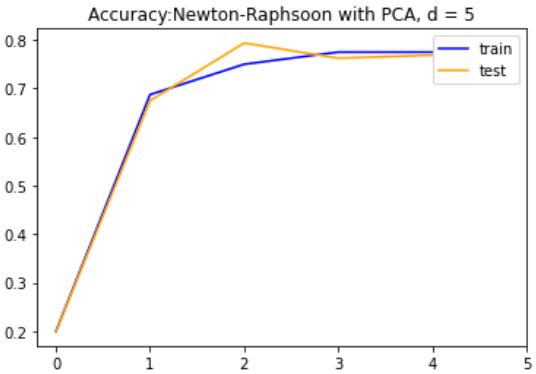
2-2(a)

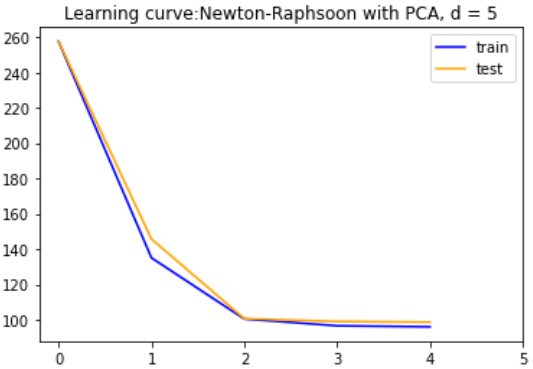
Training	Newton	GD	SGD	Mini-batch SGD
d = 2	0.5375	0.4875	0.4937	0.4875
d = 5	0.7625	0.7812	0.7625	0.6812
d = 10	0.9812	0.8562	0.8562	0.7625

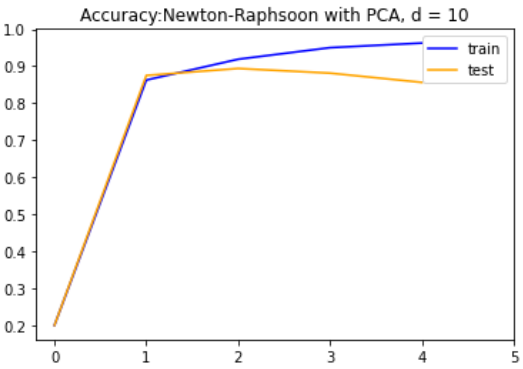
Test	Newton	GD	SGD	Mini-batch SGD
d = 2	0.5312	0.5125	0.4812	0.4937
d = 5	0.7625	0.7437	0.7625	0.7062
d = 10	0.8	0.8875	0.8812	0.8062

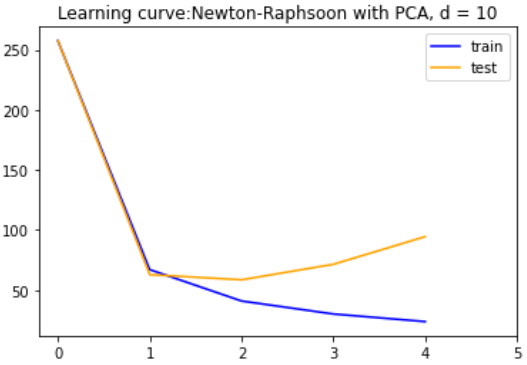

Accuracy:Newton-Raphsoon with PCA, d = 2
train
test

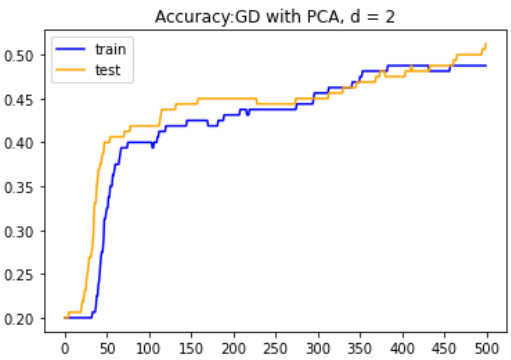

Learning curve:Newton-Raphsoon with PCA, d = 2
train
test

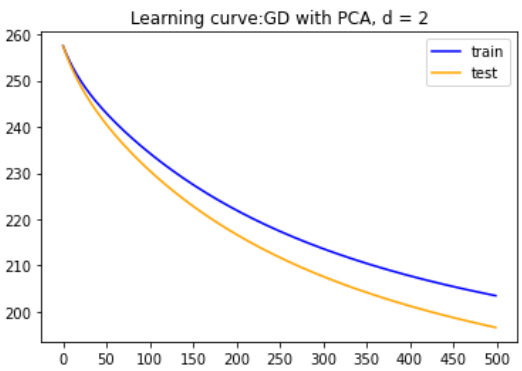

Accuracy:Newton-Raphsoon with PCA, d = 5
train
test

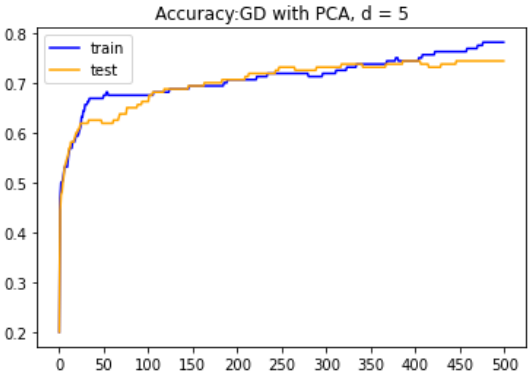

Learning curve:Newton-Raphsoon with PCA, d = 5
train
test

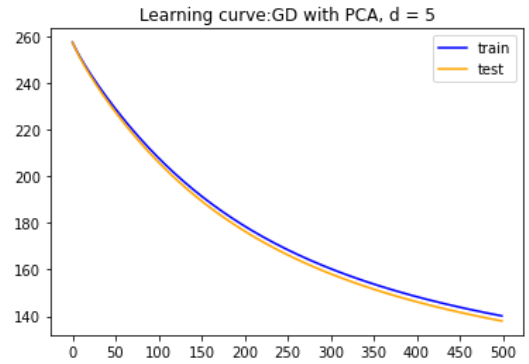

Accuracy:Newton-Raphsoon with PCA, d = 10
train
test

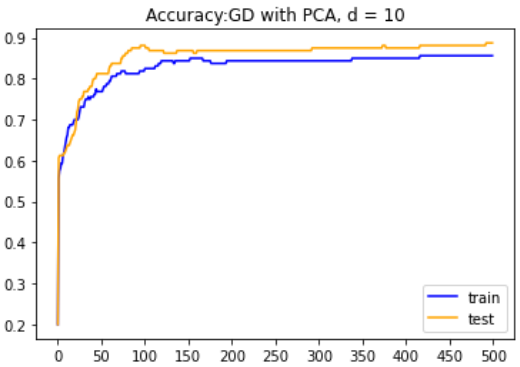

Learning curve:Newton-Raphsoon with PCA, d = 10
train
test

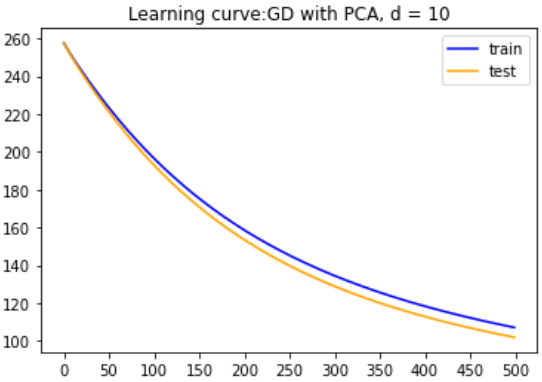

Accuracy:GD with PCA, d = 2
train
test

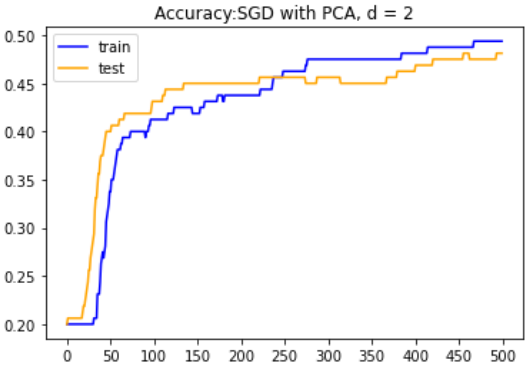

Learning curve:GD with PCA, d = 2
train
test

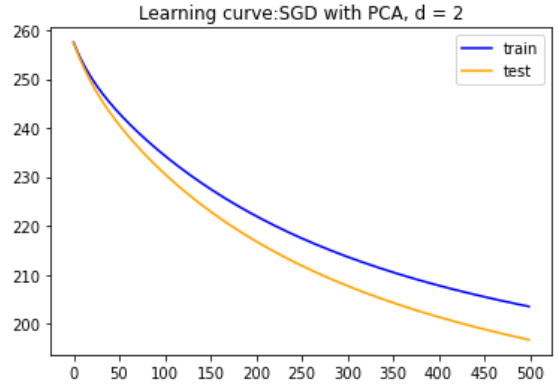

Accuracy:GD with PCA, d = 5
train
test

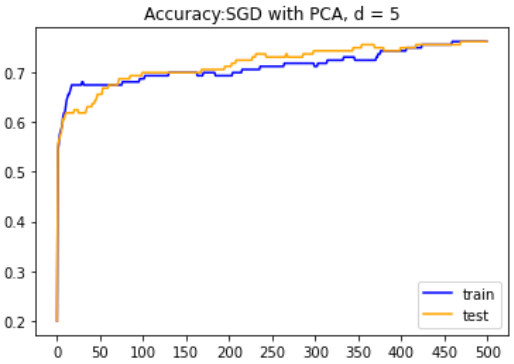

Learning curve:GD with PCA, d = 5
train
test

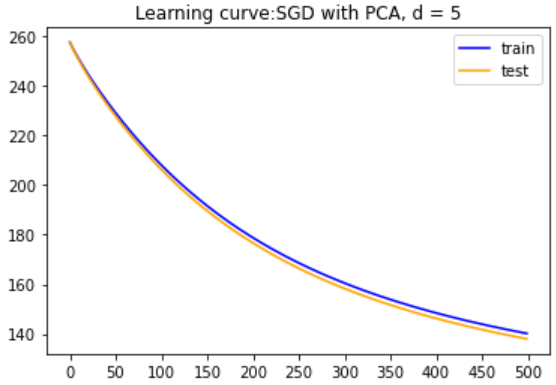

Accuracy:GD with PCA, d = 10
train
test

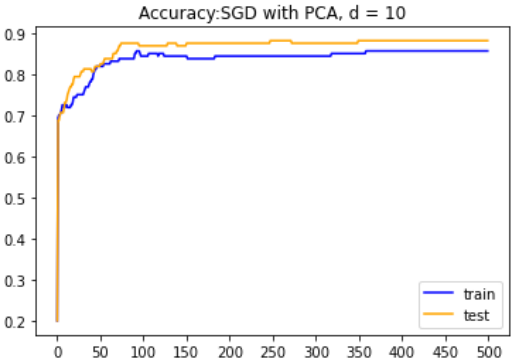

Learning curve:GD with PCA, d = 10
train
test

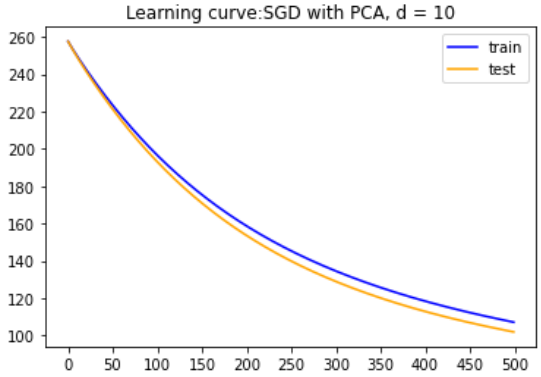

Accuracy:SGD with PCA, d = 2
train
test

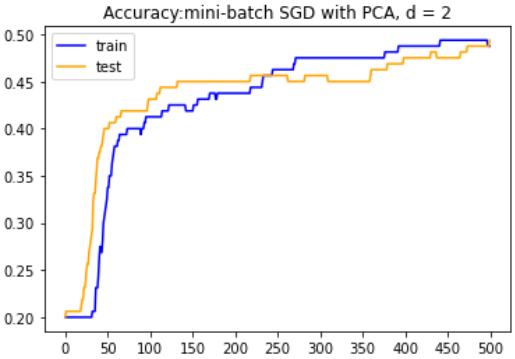

Learning curve:SGD with PCA, d = 2
train
test

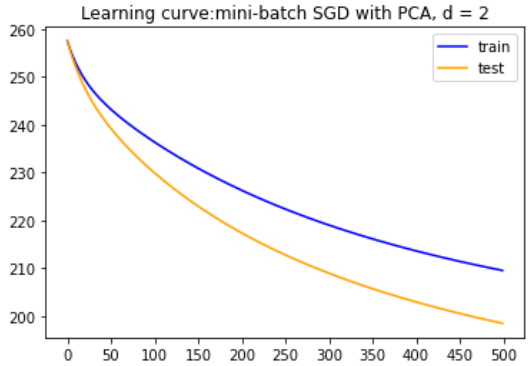

Accuracy:SGD with PCA, d = 5
train
test

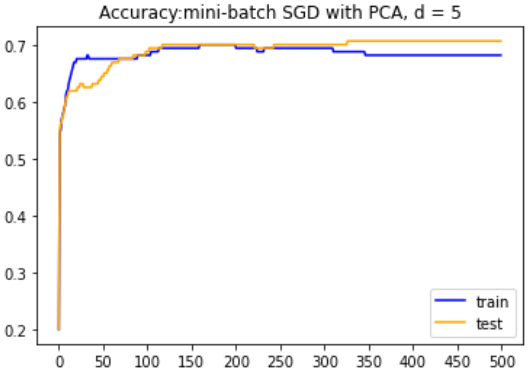

Learning curve:SGD with PCA, d = 5
train
test

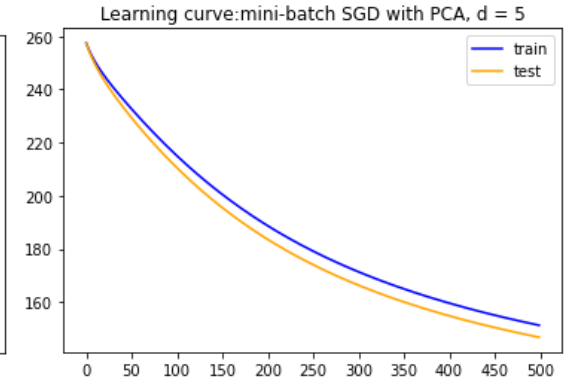

Accuracy:SGD with PCA, d = 10
train
test

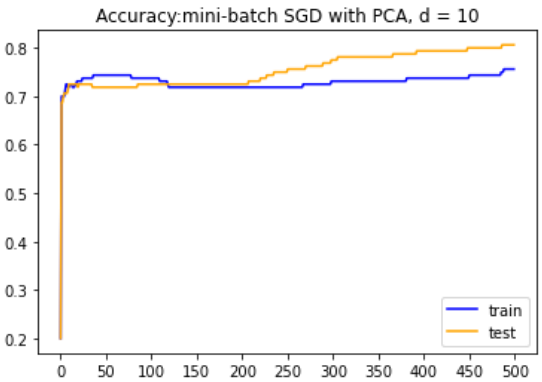

Learning curve:SGD with PCA, d = 10
train
test

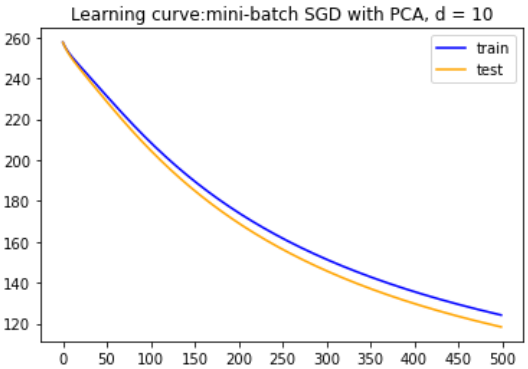

Accuracy:mini-batch SGD with PCA, d = 2
train
test


Learning curve:mini-batch SGD with PCA, d = 2
train
test


Accuracy:mini-batch SGD with PCA, d = 5
train
test

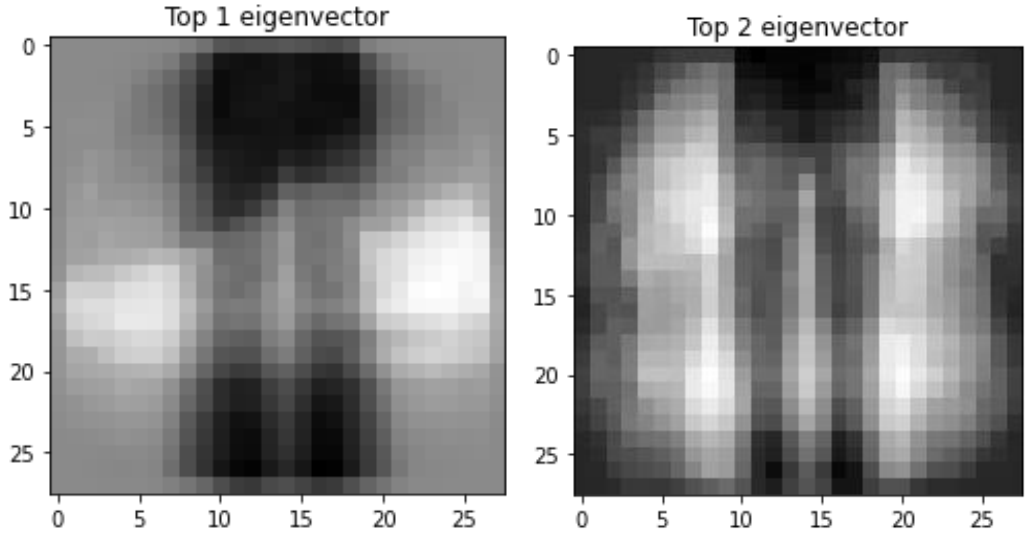

Learning curve:mini-batch SGD with PCA, d = 5
train
test

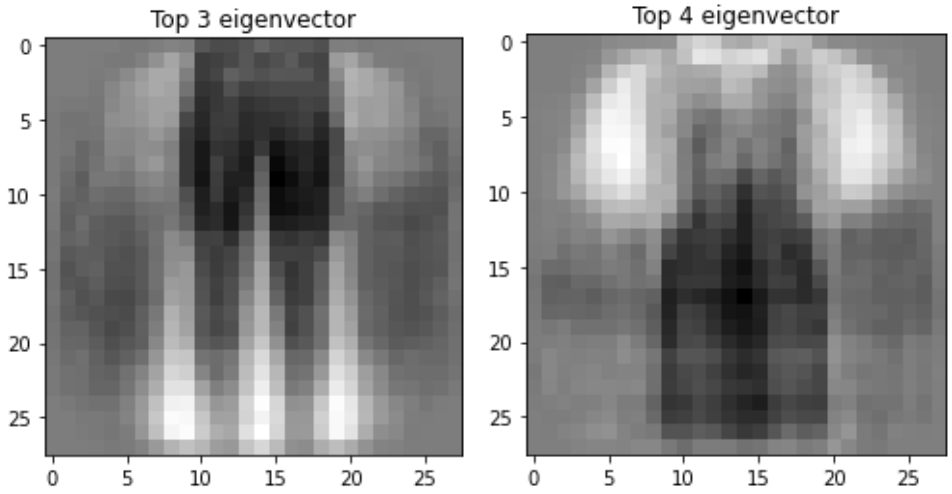

Accuracy:mini-batch SGD with PCA, d = 10
train
test

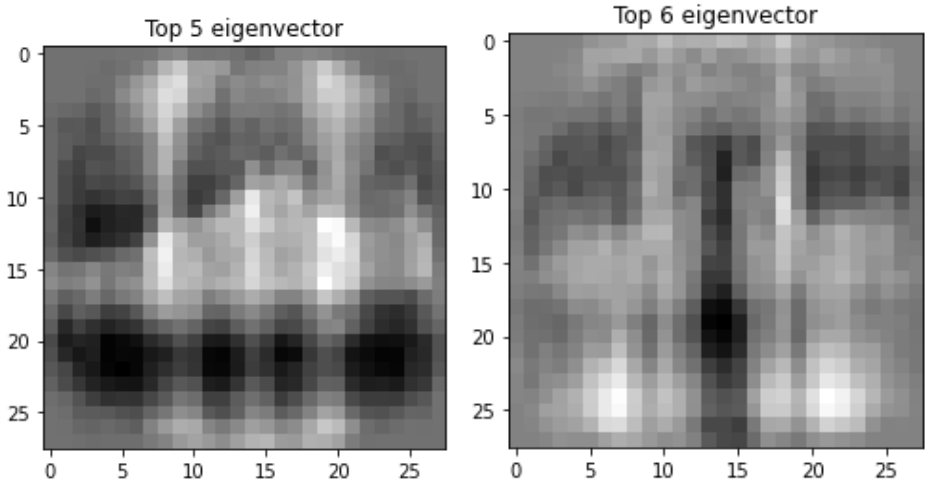

Learning curve:mini-batch SGD with PCA, d = 10
train
test

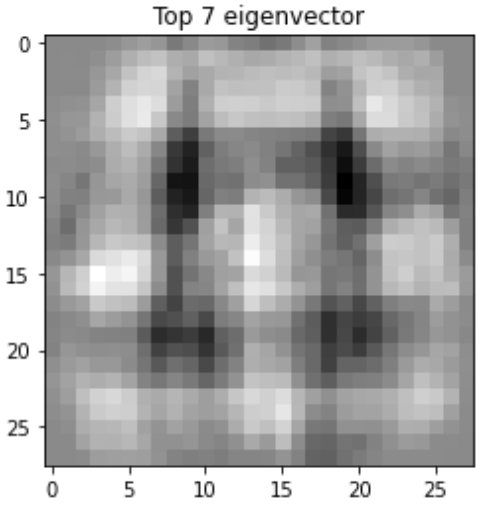
2-2(b)

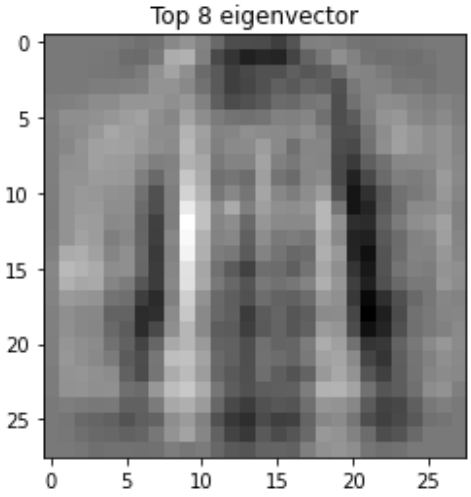
Use eigenvector in **SVD** of training data's covariance matrix.

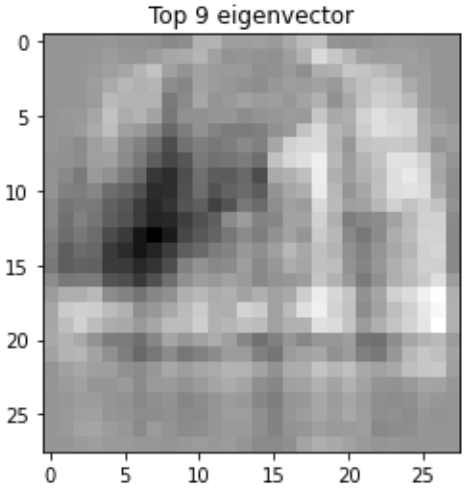


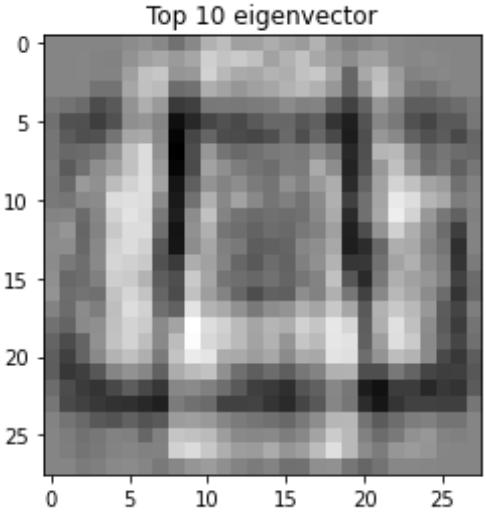




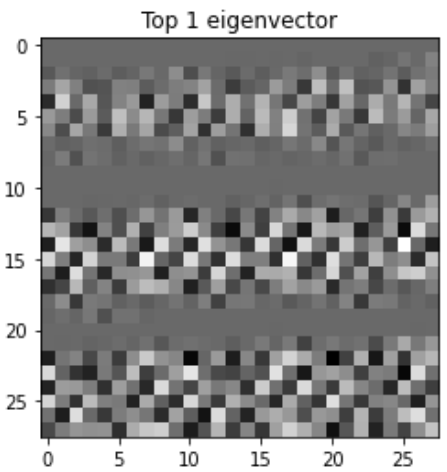


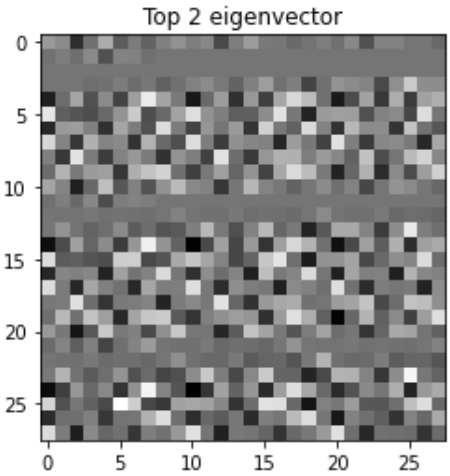


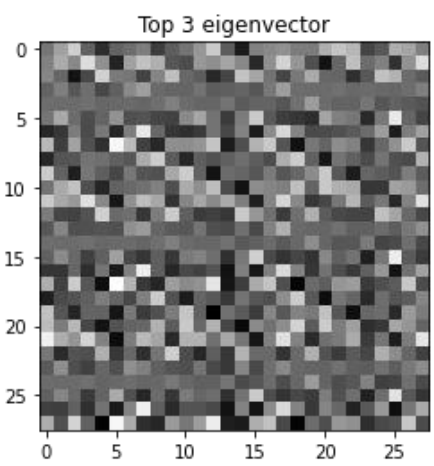


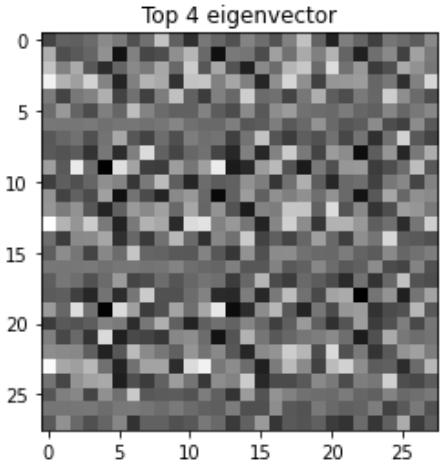


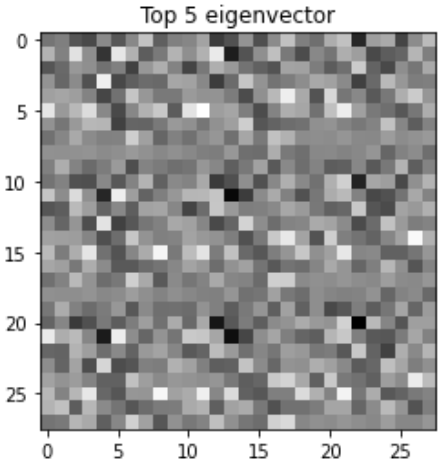
Use eigenvector of training data’s covariance matrix.

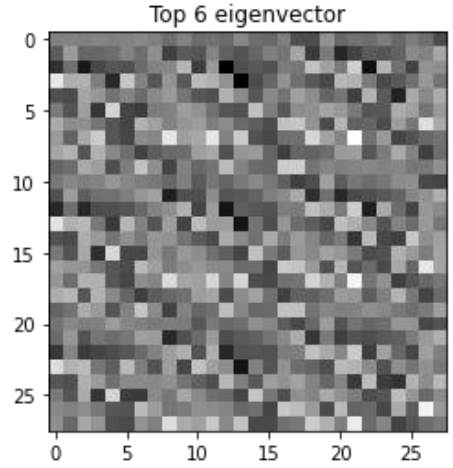


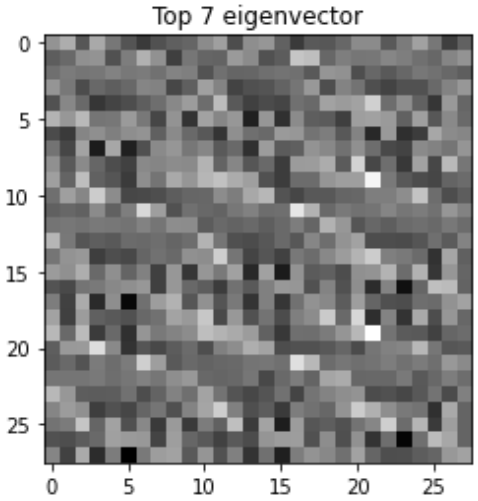


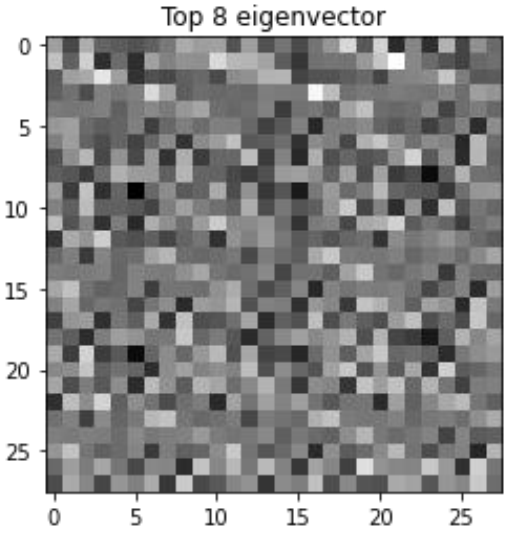

Top 3 eigenvector
0
5
10
15
20
25
0
5
10
15
20
25

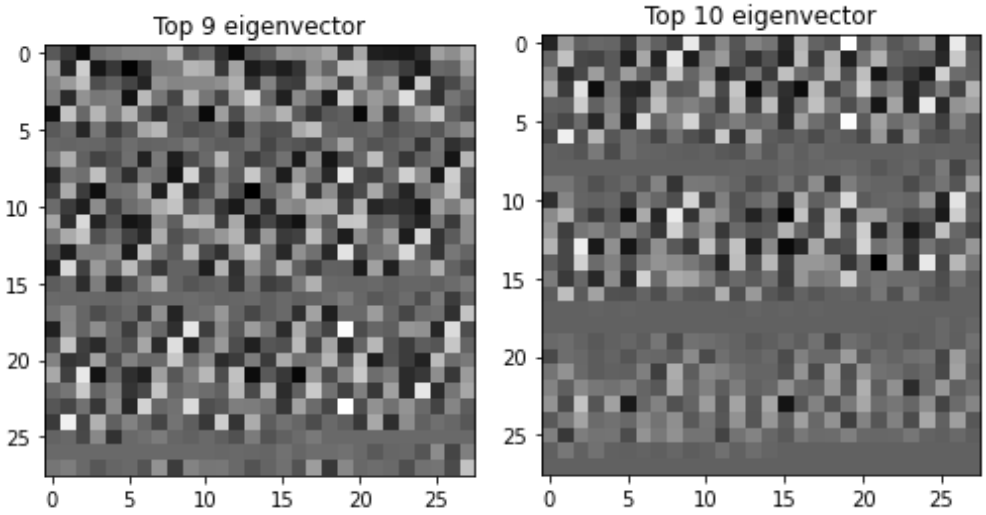

Top 4 eigenvector
0
5
10
15
20
25
0
5
10
15
20
25


Top 5 eigenvector
0
5
10
15
20
25
0
5
10
15
20
25


Top 6 eigenvector
0
5
10
15
20
25
0
5
10
15
20
25

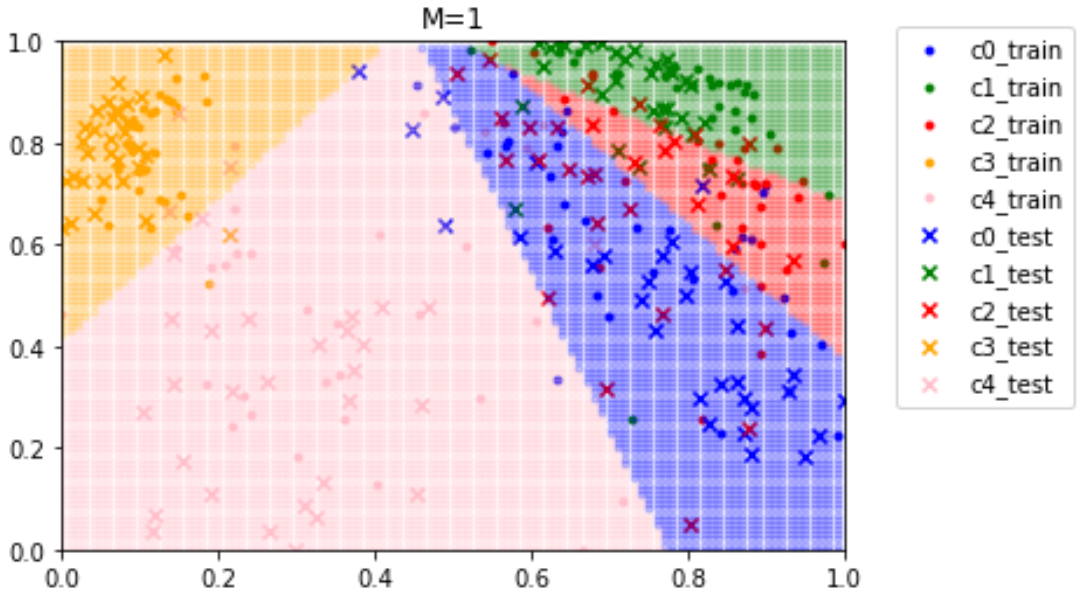

Top 7 eigenvector
0
5
10
15
20
25
0
5
10
15
20
25

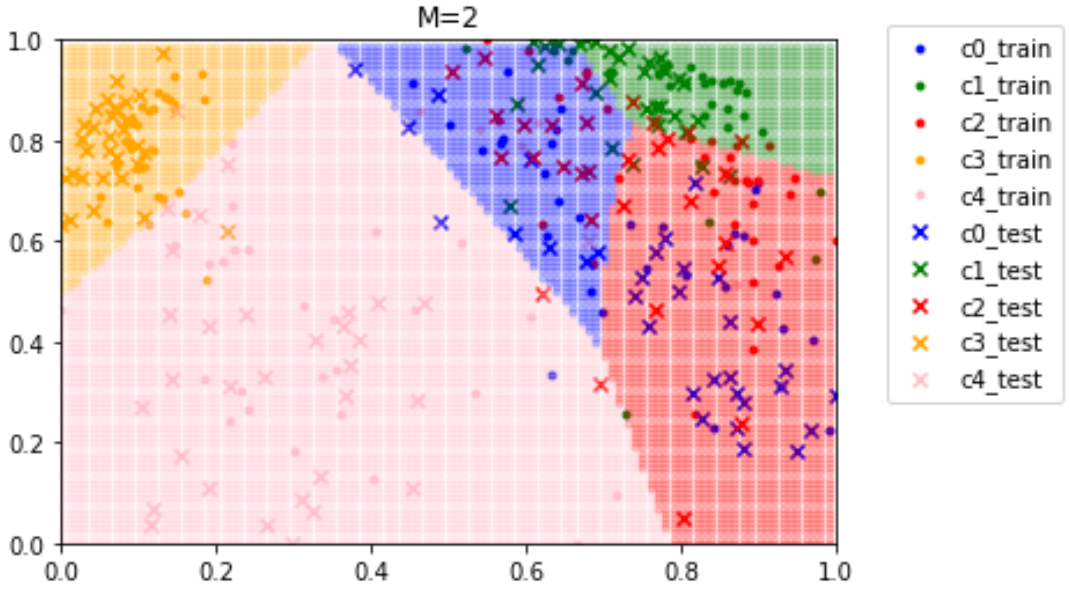

Top 8 eigenvector
0
5
10
15
20
25
0
5
10
15
20
25



2-3 (a)

Use **Newton method** to draw the graph, since it has better test accuracy among these 4 methods.





2-4 Discussion

In 2-1, we get a steep and sharp learning curve from Newton-Raphson compared to other gradient descent methods. The reason is the nature of Newton method which means its root-finding process tends to converge faster than gradient descent methods. With my experiment results, it's clear to see that Newton method only needs one iteration to find the best weight while gradient descent methods need at least 20 iterations to get decent results. On the other hand, Newton method has the highest training among these methods. However, its test accuracy isn’t as good as gradient descent methods, and it is because Newton method changes weight too much in one step so that it cannot find a more accurate weight like gradient method. The drawback of Newton method is the time. It takes much more time to finish one iteration than those gradient descent methods.

I also want to mention the differences between different gradient descent methods. First, the normal gradient descent method needs the less iterations to get 95% accuracy in test data. Second, the learning curves of SGD method and mini-batch SGD method both oscillate during the learning process, which shows SGD-family's root finding process is not as direct as the normal GD method.

In 2-2(a), as the dimension of images increases, the training and test accuracy is getting higher, and we can almost get 90% test accuracy with 10 eigenvectors. The results imply that we actually only need about 10 features to do a good classification in much less time compared to the original model.

In 2-2(b), let's call the covariance matrix of original 784 features as C The weirdest part is if I use eigenvectors of C to draw graph, the graph will be a disorganized one. However, if I used the eigenvector of C^TC to draw the graphs, they start to show the contour of pants ,clothes and shoes. I am not sure about which results are the desired one, so I show both of them.

Last but not least, In 2-3, I choose to use Newton method to train the model, since it has better test accuracy among 4 method. It is cool to know that we can use a lot of points to draw the decision regions. As the result shown above, we can easily distinguish class 1, class 3 and class 4 with only two features and one constant. Nonetheless, we have a bad result on distinguishing between class 0 and class 2. I think the reason is that class 0 consists a lot of shirts and class 2 consists a lot of short-sleeved dresses. Since shirt and short-sleeved dress are pretty similar on the top, it results we cannot draw a good decision boundary between these two classes. This problem cannot be solved even with M = 2, which is understandable because we are still only using top two features' combination to do the job. As a result, I think it means that we should take more features into consideration to do a better classification just like the conclusion in 2-1.