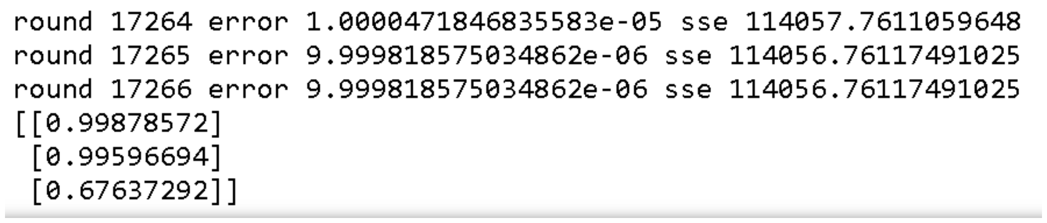
Machine Learning Submission Assignment #2

徐玉山
0851915

April 12, 2020

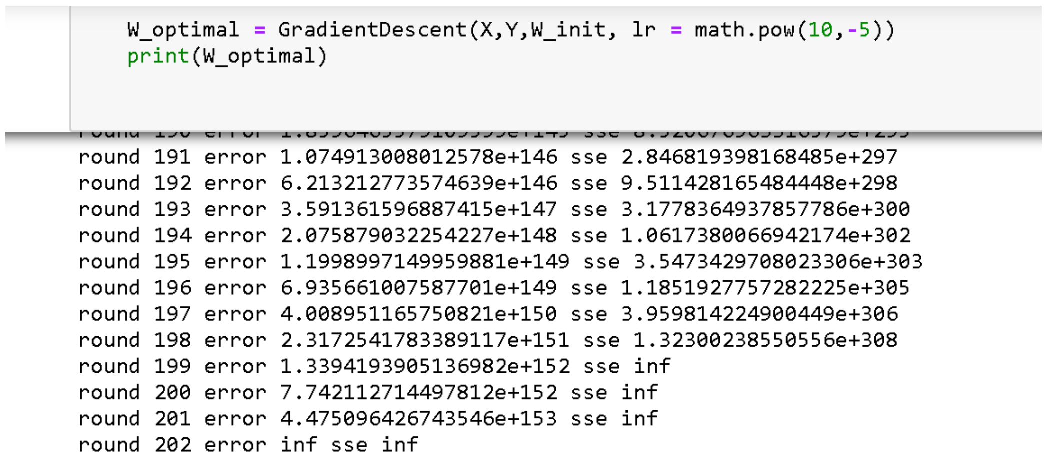
1 Problem 1. Unbiasedness of OLS

(a) 會收斂



```
round 17264 error 1.0000471846835583e-05 sse 114057.7611059648
round 17265 error 9.999818575034862e-06 sse 114056.76117491025
round 17266 error 9.999818575034862e-06 sse 114056.76117491025
[[0.99878572]
 [0.99596694]
 [0.67637292]]
```

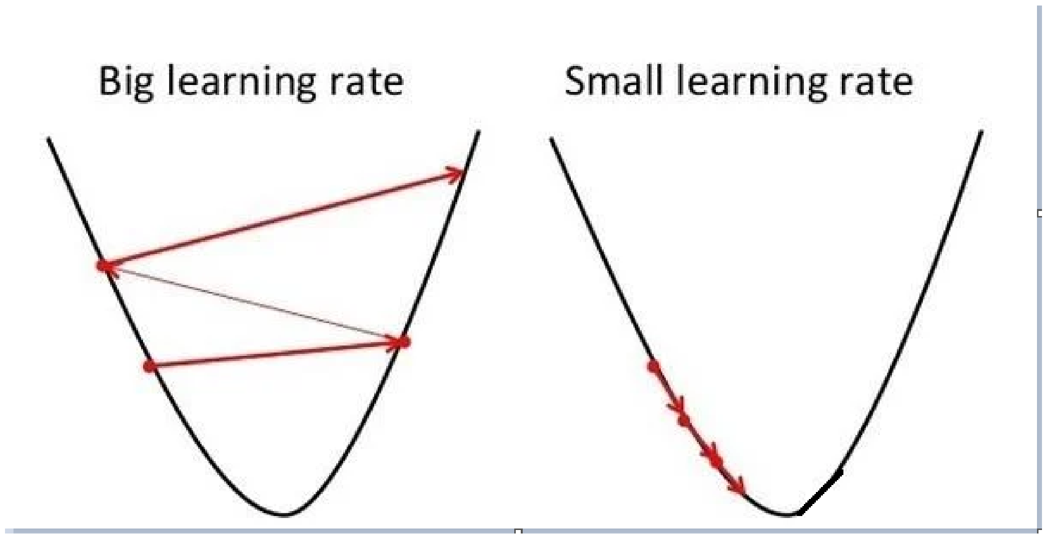
(b) 不收斂



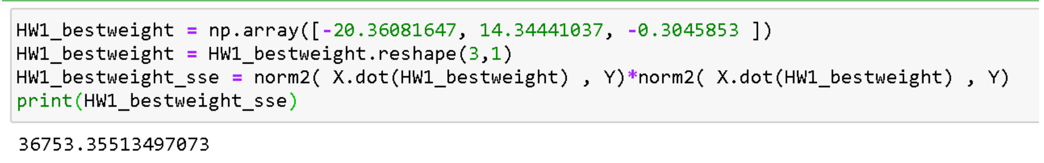
```
W_optimal = GradientDescent(X,Y,W_init, lr = math.pow(10,-5))
print(W_optimal)
```

```
round 191 error 1.074913008012578e+146 sse 2.846819398168485e+297
round 192 error 6.213212773574639e+146 sse 9.511428165484448e+298
round 193 error 3.591361596887415e+147 sse 3.1778364937857786e+300
round 194 error 2.075879032254227e+148 sse 1.0617380066942174e+302
round 195 error 1.1998997149959881e+149 sse 3.5473429708023306e+303
round 196 error 6.935661007587701e+149 sse 1.1851927757282225e+305
round 197 error 4.008951165750821e+150 sse 3.959814224900449e+306
round 198 error 2.3172541783389117e+151 sse 1.32300238550556e+308
round 199 error 1.3394193905136982e+152 sse inf
round 200 error 7.742112714497812e+152 sse inf
round 201 error 4.475096426743546e+153 sse inf
round 202 error inf sse inf
```

(c) b可能取的learning rate太大了，導致會像是下圖



(d) 上次作業算出來的sse比較小，用gradient descent算出來的sse比較大。
有可能gradient descent卡在某個區域極值
可能要換初始值W看看能不能達到比較好的結果

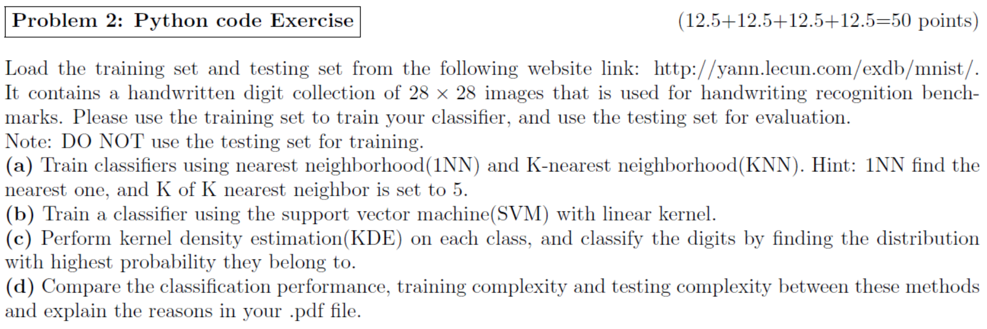


```
HW1_bestweight = np.array([-20.36081647, 14.34441037, -0.3045853 ])
HW1_bestweight = HW1_bestweight.reshape(3,1)
HW1_bestweight_sse = norm2( X.dot(HW1_bestweight) , Y)*norm2( X.dot(HW1_bestweight) , Y)
print(HW1_bestweight_sse)
```

```
36753.35513497073
```

2 Problem 2. Python code Exercise

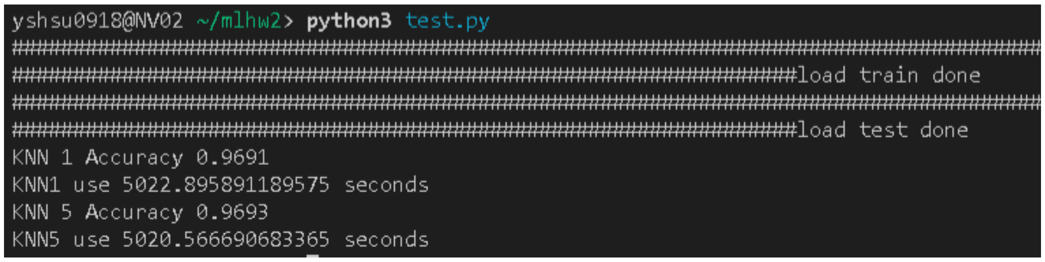
題目



Problem 2: Python code Exercise (12.5+12.5+12.5+12.5=50 points)

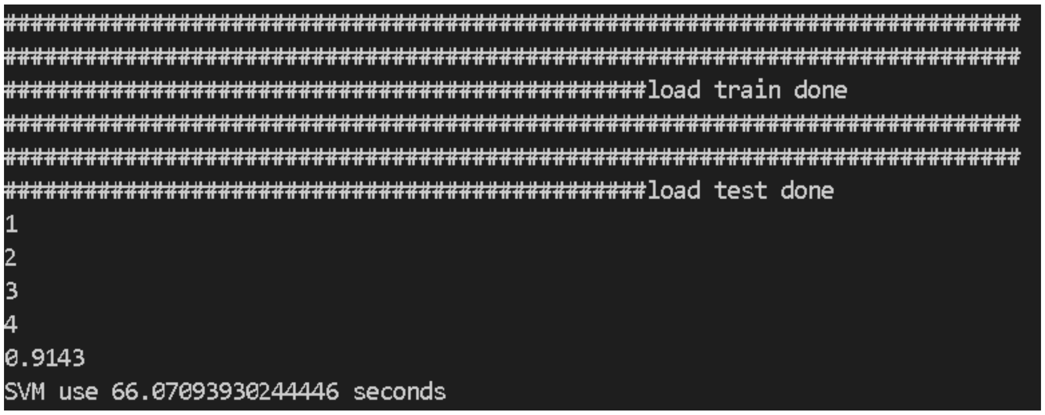
Load the training set and testing set from the following website link: http://yann.lecun.com/exdb/mnist/. It contains a handwritten digit collection of 28 × 28 images that is used for handwriting recognition benchmarks. Please use the training set to train your classifier, and use the testing set for evaluation.
Note: DO NOT use the testing set for training.
(a) Train classifiers using nearest neighborhood(1NN) and K-nearest neighborhood(KNN). Hint: 1NN find the nearest one, and K of K nearest neighbor is set to 5.
(b) Train a classifier using the support vector machine(SVM) with linear kernel.
(c) Perform kernel density estimation(KDE) on each class, and classify the digits by finding the distribution with highest probability they belong to.
(d) Compare the classification performance, training complexity and testing complexity between these methods and explain the reasons in your .pdf file.

(a) KNN



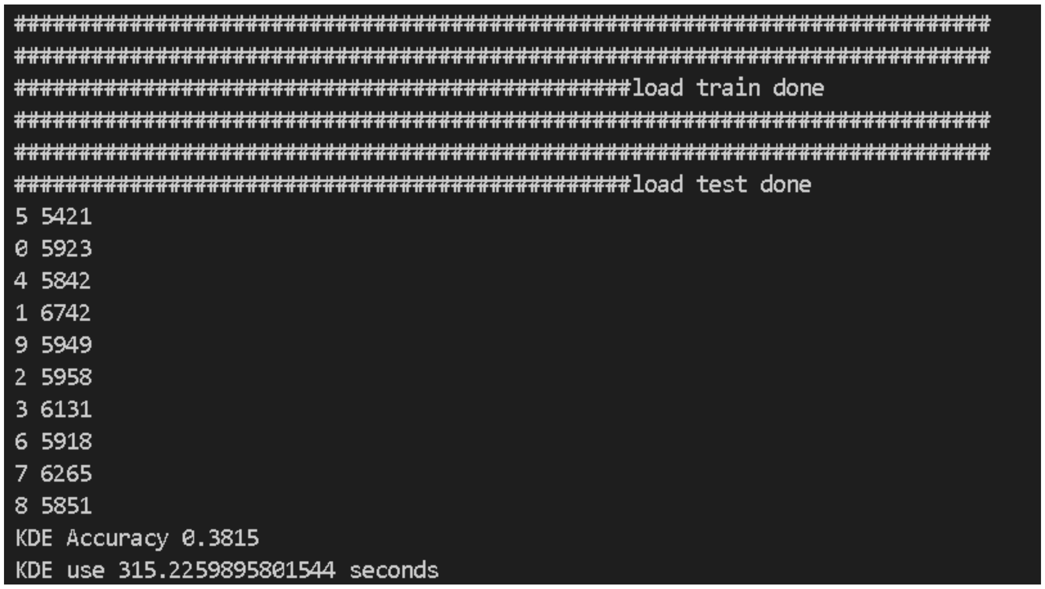
```
yshsu0918@NV02 ~/mlhw2> python3 test.py
####################################################################################
################################################################load train done
####################################################################################
################################################################load test done
KNN 1 Accuracy 0.9691
KNN1 use 5022.895891189575 seconds
KNN 5 Accuracy 0.9693
KNN5 use 5020.566690683365 seconds
```

(b) SVM



```
#############################################################################
#############################################################################
################################################load train done
#############################################################################
#############################################################################
################################################load test done
1
2
3
4
0.9143
SVM use 66.07093930244446 seconds
```

(c) KDE



```
############################################################################
############################################################################
################################################load train done
############################################################################
############################################################################
################################################load test done
5 5421
0 5923
4 5842
1 6742
9 5949
2 5958
3 6131
6 5918
7 6265
8 5851
KDE Accuracy 0.3815
KDE use 315.2259895801544 seconds
```

(d) Summary

d = dimension = 28*28
n = number of training/testing sample

none	KNN	SVM	KDE
training complexity	0	n^3	d
testingccomplexity	nd	nd	nd
classication performance	0.96	0.91	0.38