ECE586 MP2 Least-Squares Solutions Report

Libo Zhang (NetID – lz200)

Exercise 1

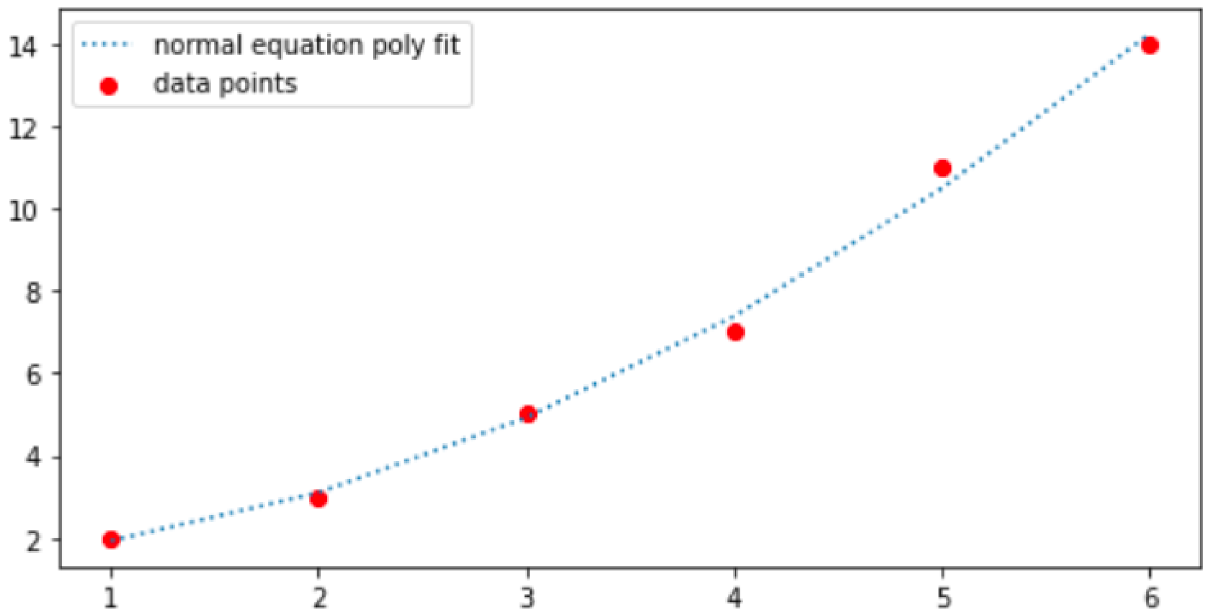
```
: x = np.arange(1, 10)
  create_vandermonde(x, 3)

: array([[  1.,   1.,   1.,   1.],
         [  1.,   2.,   4.,   8.],
         [  1.,   3.,   9.,  27.],
         [  1.,   4.,  16.,  64.],
         [  1.,   5.,  25., 125.],
         [  1.,   6.,  36., 216.],
         [  1.,   7.,  49., 343.],
         [  1.,   8.,  64., 512.],
         [  1.,   9.,  81., 729.]])
```

Exercise 2

```
poly1_expr = ' + '.join(['{0:.4f} x^{1}'.format(v, i) for i, v in enum
print('normal equation polynomial fit is {0}'.format(poly1_expr))
print('normal equation MSE is {0:.4f}'.format(mse))

normal equation polynomial fit is 0.3214 x^2 + 0.2071 x^1 + 1.4000
normal equation MSE is 0.0810
```



Exercise 3

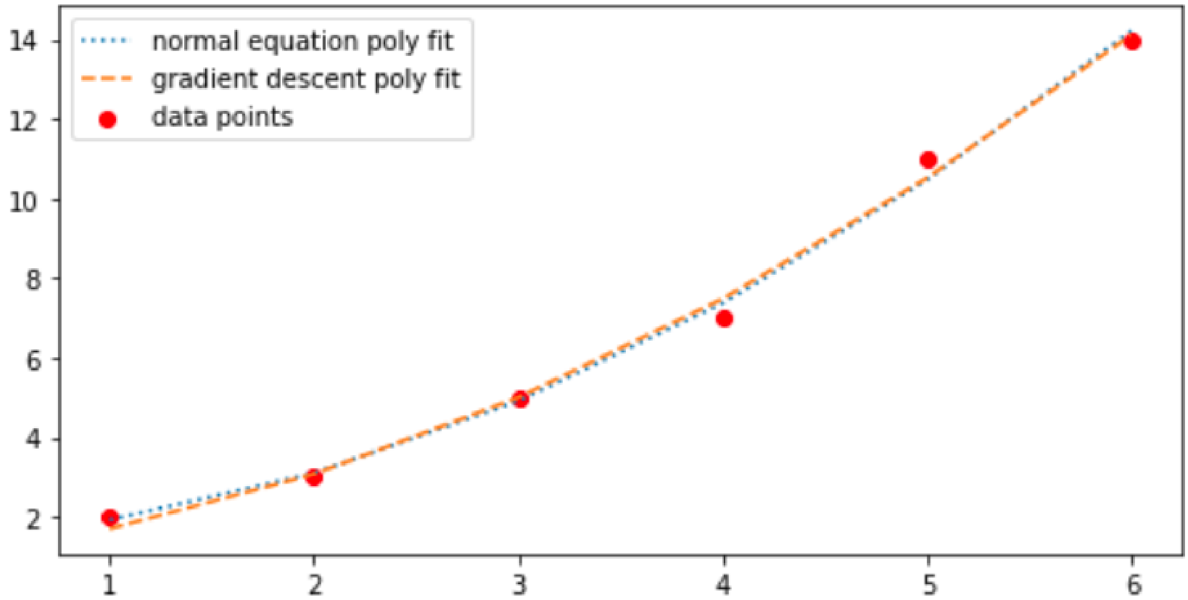
With the help of Least Squares function, I find the Mean Squared Error is 0.0810, so the gradient descent MSE at most 20% larger should be 0.0810 * 1.2 = 0.0971.

With the help of Least Squares Gradient Descent function (step size = 0.0002), I find that:

When T (number of iterations) is equal to 80000, the gradient descent MSE is 0.0953 < 0.0971.

When T = 500000, the gradient descent MSE is 0.0815, which is very close to the Least Squares MSE, 0.0810.

```
gradient descent polynomial fit is 0.2769 x^2 + 0.5426 x^1 + 0.8835
previous MSE is 0.0810
at most 20% larger than the previous MSE is 0.0971
Current T (number of iterations) is 80000
gradient descent MSE is 0.0953
```



Exercise 4

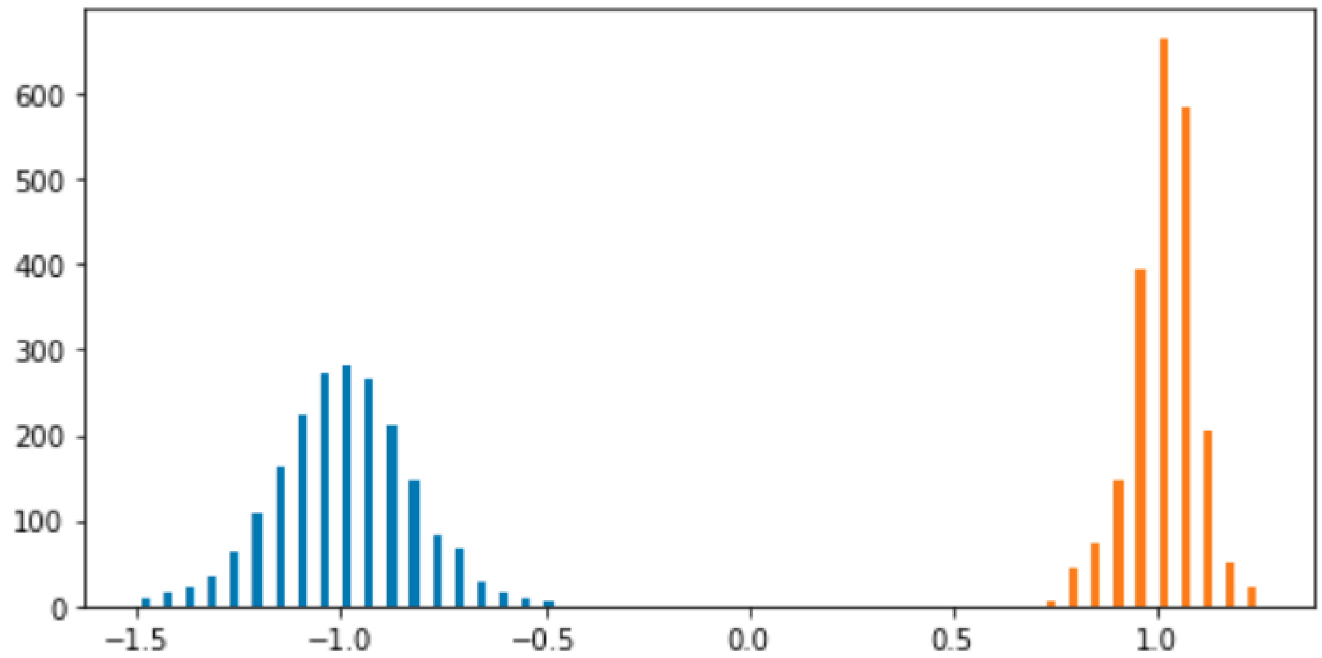
```
16]: # Pairwise experiment for LSQ to classify between 0 and 1
     mnist_pairwise_LS(df, 0, 1, verbose=True)
```

```
Pairwise experiment, mapping 0 to -1, mapping 1 to 1
training error = 0.43%, testing error = 0.79%
Confusion matrix for Training Set is:
 [[2064    2]
 [  17 2325]]
Confusion matrix for Test Set is:
 [[2047   19]
 [  16 2326]]
```

```
16]: array([0.00431034, 0.00794011])
```

For the test set, compute the histogram of the function output separately for each class and then plot the two histograms together.

```
array([0.00431034, 0.00794011])
```



Exercise 5

```python
print(np.round(err_matrix*100, 2))
```

```
<ipython-input-17-bea7d5aad970>:3: TqdmDeprecationWarning: This func
Please use `tqdm.notebook.tqdm` instead of `tqdm.tqdm_notebook`
  for a, b in tqdm(it.combinations(range(10), 2), total=45):
```

100% 45/45 [00:41<00:00, 1.08it/s]

```
[[0.   0.34 0.63 0.26 0.2  1.82 0.51 0.28 0.71 0.41]
 [1.23 0.   0.84 0.71 0.25 0.61 0.07 0.53 1.85 0.32]
 [2.19 2.71 0.   1.83 0.63 1.53 1.35 0.86 1.97 0.45]
 [1.65 2.12 4.29 0.   0.43 2.6  0.31 0.69 2.81 1.36]
 [0.93 1.39 2.52 1.71 0.   0.74 0.51 0.73 0.44 2.35]
 [3.58 1.91 3.79 5.72 1.91 0.   1.54 0.22 2.42 1.08]
 [2.27 1.59 2.67 1.93 1.36 3.5  0.   0.02 0.78 0.17]
 [0.84 1.56 2.61 2.15 2.31 1.54 1.03 0.   0.43 2.7 ]
 [1.68 3.77 4.27 5.68 1.23 4.96 2.29 2.36 0.   1.58]
 [1.51 1.24 2.37 2.83 5.11 3.33 0.98 5.61 3.54 0.  ]]
```

Exercise 6

Report both the overall classification error rate and confusion matrices for both the training and test sets.

```
-------------------------------------------------------------
training error = 13.56%, testing error = 15.28%
-------------------------------------------------------------
Confusion matrix for Training Set is:
 [[1986    1    8    4    7   17   20    1   23    1]
 [   0 2276    8    5    5    5    8    5   20    3]
 [  36   77 1703   42   32    1   72   34   67   11]
 [  13   56   56 1921   10   40   16   33   57   44]
 [   3   36   14    1 1826   15   16    6   11  110]
 [  60   33    6  172   42 1319   63   12   87   44]
 [  41   26   15    0   20   29 1890    0   13    0]
 [  21   72   15    9   53    3    3 1906    4  102]
 [  17  164   16   78   32   68   18    5 1606   46]
 [  29   21    3   48  128    2    0  153   25 1719]]
-------------------------------------------------------------
Confusion matrix for Test Set is:
 [[1972    2   10   10   10   17   22    3   14    4]
 [   1 2269   18    8   11    7    7    3   21    4]
 [  36   93 1653   55   60    5   89   35   65   11]
 [  12   56   65 1774   13   37   15   40   53   40]
 [   4   45   17    6 1784   25   12    7   26  108]
 [  65   46    9  184   57 1354   66   19  114   43]
 [  37   24   29    2   31   34 1927    1   13    5]
 [  14   76   21   17   66    1    2 1902    0  114]
 [  24  144   23   85   43   82   29    7 1526   50]
 [  25   25    4   37  122    9    2  190   16 1630]]
```

The following pages are Appendices, in which I showed the results of running all test cases in lsq_code_test.py. My running results matched the desired output indicated in the test code.

As for the training and testing errors are a bit different from those in the desired output, after consulting with the professor, this is because splitting training and testing samples is random. Sometimes I get a larger error such as 5.X% or 6.X%. Sometimes I get a smaller error such as 3.X% or 4.X%.

Appendices

```
Vandermonde Example 1:
 [[1. 1. 1.]
 [1. 3. 9.]
 [1. 2. 4.]]

Vandermonde Example 2:
 [[ 1. -2.  4. -8.]
 [ 1. -1.  1. -1.]
 [ 1.  0.  0.  0.]
 [ 1.  1.  1.  1.]
 [ 1.  2.  4.  8.]]

solve_linear_LS Example 1:
 [-4.   6.5 -1.5]

solve_linear_LS Example 2:
 [ 1.25714286  0.58333333  0.07142857 -0.08333333]

solve_linear_gd Example 1:
 [-3.92367303  6.41369966 -1.47965981]

solve_linear_gd Example 2:
 [ 1.24030161  0.51307179  0.07121913 -0.06784267]
```

```
#
# Desired output of this script
# ----------------------------
#
# Vandermonde Example 1:
#  [[1 1 1]
#  [1 3 9]
#  [1 2 4]]

# Vandermonde Example 2:
#  [[ 1 -2  4 -8]
#  [ 1 -1  1 -1]
#  [ 1  0  0  0]
#  [ 1  1  1  1]
#  [ 1  2  4  8]]

# solve_linear_LS Example 1:
#  [-4.   6.5 -1.5]

# solve_linear_LS Example 2:
#  [ 1.25714286  0.58333333  0.07142857 -0.08333333]

# solve_linear_gd Example 1:
#  [-3.92367303  6.41369966 -1.47965981]

# solve_linear_gd Example 2:
#  [ 1.24030161  0.51307179  0.07121913 -0.06784267]
```

```
mnist_pairwise_LS Example 0 :
Pairwise experiment, mapping 2 to -1, mapping 3 to 1
training error = 1.88%, testing error = 3.82%
Confusion matrix for Training Set is:
 [[2052   36]
 [  44 2131]]
Confusion matrix for Test Set is:
 [[2023   66]
 [  97 2079]]
results:  [0.01876613 0.03821805]

mnist_pairwise_LS Example 1 :
Pairwise experiment, mapping 2 to -1, mapping 3 to 1
training error = 1.97%, testing error = 4.62%
Confusion matrix for Training Set is:
 [[2058   30]
 [  54 2121]]
Confusion matrix for Test Set is:
 [[1989  100]
 [  97 2079]]
results:  [0.01970443 0.04618992]

mnist_pairwise_LS Example 2 :
Pairwise experiment, mapping 2 to -1, mapping 3 to 1
training error = 1.90%, testing error = 3.99%
Confusion matrix for Training Set is:
 [[2051   37]
 [  44 2131]]
Confusion matrix for Test Set is:
 [[2019   70]
 [ 100 2076]]
results:  [0.0190007  0.03985932]
```

```
# mnist_pairwise_LS Example 0 :
# Pairwise experiment, mapping 2 to -1, mapping 3 to 1
# training error = 1.85%, testing error = 4.20%
# Confusion matrix:
#  [[2013   75]
#  [ 104 2071]]
# results:  [0.01852286 0.04198921]

# mnist_pairwise_LS Example 1 :
# Pairwise experiment, mapping 2 to -1, mapping 3 to 1
# training error = 1.85%, testing error = 3.85%
# Confusion matrix:
#  [[2015   73]
#  [  91 2084]]
# results:  [0.01852286 0.03847056]

# mnist_pairwise_LS Example 2 :
# Pairwise experiment, mapping 2 to -1, mapping 3 to 1
# training error = 1.83%, testing error = 3.94%
# Confusion matrix:
#  [[2002   86]
#  [  82 2093]]
# results:  [0.01828839 0.03940887]
```