Assignment3_Release

March 9, 2019

1 Homework 3

1.1 Problem 1

perceptron and linear classifier

```
In [62]: #Load Data
         import numpy as np
         import matplotlib.pyplot as plt
         import pandas as pd
         from sklearn.utils import shuffle
         X = np.array([[-0.6,1],[-3,-4],[3,-2],[0.5,1]])
         Y = np.array([[-1],[-1],[1],[1]])
         print('X \n',X)
         print('Y \n',Y)
```

```
X
 [[-0.6  1. ]
 [-3.  -4. ]
 [ 3.  -2. ]
 [ 0.5  1. ]]
Y
 [[-1]
 [-1]
 [ 1]
 [ 1]]
```

Use: =1, . . . , :
If sign((_))_:
+_(_)
= _

```
In [104]: #perceptron
          w = np.zeros(shape=(5,2))
          t = np.zeros(shape=(5,1))
          Sign = np.zeros(shape=(5,1))
          w[0] = np.array([0,1])
```

```
            t[0] = 0
            #
            for i in range(len(t)-1):
                Sign[i] = np.sign(np.dot(w[i],X[i])-t[i])
                if Sign[i] != Y[i]:
                    w[i+1] = w[i] + Y[i]*X[i]
                    t[i+1] = t[i] - Y[i]
                else:
                    w[i+1] = w[i]
                    t[i+1] = t[i]
            print("w \n",w)
            print("t \n",t)
```

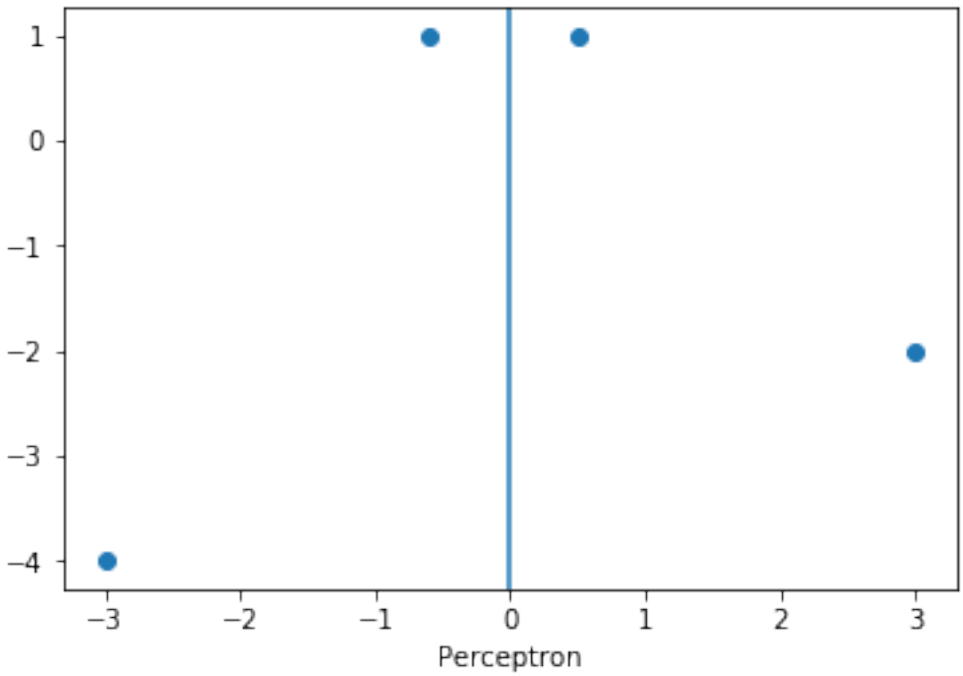
```
w
 [[0.  1. ]
 [0.6 0. ]
 [0.6 0. ]
 [0.6 0. ]
 [1.1 1. ]]
t
 [[0.]
 [1.]
 [1.]
 [1.]
 [0.]]
```

Reference: https://medium.com/@thomascountz/calculate-the-decision-boundary-of-a-single-perceptron-visualizing-linear-separability-c4d77099ef38

```
In [109]: a, b = X.T
          plt.scatter(a,b)
          slope = -(t[4] / w[4,0]) /  (-(t[4] /w[4,1]))
          # which is the y axis
          intercept = (-t[4] / w[4,1])
          #which is 0
          #c = np.linspace(-4,2,100)
          #d = slope*c+intercept
          plt.axvline(x=0)
          #plt.plot(c, d, '-r', label='classfier')
          print(slope,intercept)
          plt.xlabel('Perceptron')
          plt.show()
```

```
[nan] [-0.]


/Users/Lisa/anaconda3/lib/python3.7/site-packages/ipykernel_launcher.py:3: RuntimeWarning: inva
  This is separate from the ipykernel package so we can avoid doing imports until
```



1.2 Problem 2 & Problem 3

1.3 Problem 4

1.4 Wine Quality Classification

In this assignment, we will use logistic regression to judge the quality of wines. The dataset is taken from UCI machine learning repository. For description of the dataset, see here.

Attributes of the dataset are listed as following: 1. fixed acidity 2. volatile acidity 3. citric acid 4. residual sugar 5. chlorides 6. free sulfur dioxide 7. total sulfur dioxide 8. density 9. pH 10. sulphates 11. alcohol

Output variable (based on sensory data): 12. quality (score between 0 and 10)

The following code loads the dataset, and the dataset looks like the following:

```
In [84]: #train = np.genfromtxt('wine_training1.txt', delimiter=',')
         red = pd.read_csv('winequality-red.csv')
         white = pd.read_csv('winequality-white.csv')
         red = shuffle(red, random_state = 10)
         white = shuffle(white, random_state = 10)
         red.head(10)
         white.head(10)
```

```
Out[84]:       fixed acidity  volatile acidity  citric acid  residual sugar  chlorides  \
         4731            5.3              0.31         0.38            10.5      0.031
         937             6.1              0.36         0.58            15.0      0.044
```

```
1217            8.0              0.61         0.38            12.1      0.301
3296            6.6              0.28         0.42             8.2      0.044
4524            6.6              0.16         0.25             9.8      0.049
3640            6.8              0.19         0.33             4.9      0.047
785             7.6              0.30         0.27            10.6      0.039
393             7.3              0.24         0.43             2.0      0.021
562             7.7              0.34         0.27             8.8      0.063
1285            7.8              0.16         0.41             1.7      0.026

      free sulfur dioxide  total sulfur dioxide  density    pH  sulphates  \
4731                 53.0                 140.0  0.99321  3.34       0.46
937                  42.0                 115.0  0.99780  3.15       0.51
1217                 24.0                 220.0  0.99930  2.94       0.48
3296                 60.0                 196.0  0.99562  3.14       0.48
4524                 59.5                 137.0  0.99500  3.16       0.38
3640                 42.0                 130.0  0.99283  3.12       0.56
785                  31.0                 119.0  0.99815  3.27       0.30
393                  20.0                  69.0  0.99000  3.08       0.56
562                  39.0                 184.0  0.99690  3.09       0.63
1285                 29.0                 140.0  0.99100  3.02       0.78

      alcohol  quality
4731     11.7        6
937       9.0        5
1217      9.2        5
3296      9.4        5
4524     10.0        6
3640     11.0        6
785       9.3        6
393      12.2        6
562       9.2        6
1285     12.5        6
```

1.5 Data Splitting

To get this into a binary classification task. We split the quality into a binary feature *good* or *bad* depending on whether the quality is larger than 6 or not.

Next we randomly pick 70% of the data to be our training set and the remaining for testing for both red and white wines.

```
In [86]: from sklearn.model_selection import train_test_split
         X_red = red.iloc[:, :-1]
         y_red = red.iloc[:, -1] >= 6

         X_train_red, X_test_red, y_train_red, y_test_red = train_test_split(X_red, y_red, tes

         X_white = white.iloc[:, :-1]
         y_white = white.iloc[:, -1] >= 6
```

```
X_train_white, X_test_white, y_train_white, y_test_white = train_test_split(X_white, y

#y_red.head(10)
y_white.head(10)
```

```
Out[86]: 4731     True
         937     False
         1217    False
         3296    False
         4524     True
         3640     True
         785      True
         393      True
         562      True
         1285     True
         Name: quality, dtype: bool
```

1.6 Problem 1 Logistic Regression for Red Wine

Using scikit learn, train a Logistic Regression classifier using 'X_trn_red, y_trn_red'. Use the solver sag, which stands for Stochastic Average Gradient. Set max iteration to be 10000. Test the model on X_test_red. Output the testing error.

```
In [91]: #========Your code here ======
         from sklearn.linear_model import LogisticRegression
         from sklearn.metrics import hamming_loss
         clf = LogisticRegression(random_state=0, solver='sag',multi_class='multinomial', max_i
         y_red_pred = clf.predict(X_test_red)
         error_red = hamming_loss(y_test_red, y_red_pred)
         #========================
         print('The testing error for red wine is: ' + str(error_red) + '.')
```

```
The testing error for red wine is: 0.275.
```

1.7 Problem 2 Logistic Regression for White Wine

Using scikit learn, train a Logistic Regression classifier using 'X_trn_white, y_trn_white'. Use the solver sag, which stands for Stochastic Average Gradient. Set max iteration to be 10000. Test the model on X_test_white. Output the testing error.

```
In [93]: #========Your code here ======
         from sklearn.linear_model import LogisticRegression
         from sklearn.metrics import hamming_loss
         clf2 = LogisticRegression(random_state=0, solver='sag',multi_class='multinomial', max_
         y_white_pred = clf2.predict(X_test_white)
         error_white = hamming_loss(y_test_white, y_white_pred)
         #========================
         print('The testing error for white wine is: ' + str(error_white) + '.')
```

```
The testing error for white wine is: 0.2612244897959184.
```

1.8 Problem 3

Use the model you trained using 'X_trn_white, y_trn_white' to test on 'X_test_red' and use the model you trained on 'X_test_white'. Print out the errors and compare with previous results. Explain.

```
In [94]: #========Your code here ======
         y_red_pred_2 = clf2.predict(X_test_red)
         error_red2 = hamming_loss(y_test_red, y_red_pred_2)
         #
         y_white_pred = clf.predict(X_test_white)
         error_white2 = hamming_loss(y_test_white, y_white_pred)
         #========================
         print('The testing error for red wine using white wine training data is: ' + str(erro
         print('The testing error for white wine using red wine training data is: ' + str(erro
```

```
The testing error for red wine using white wine training data is: 0.36666666666666664.
The testing error for white wine using red wine training data is: 0.33129251700680273.
```

Comment:

The error here is much larger and growed about 0.1because we classified red wine and white wine into 2 different set, and they are Trained differently.

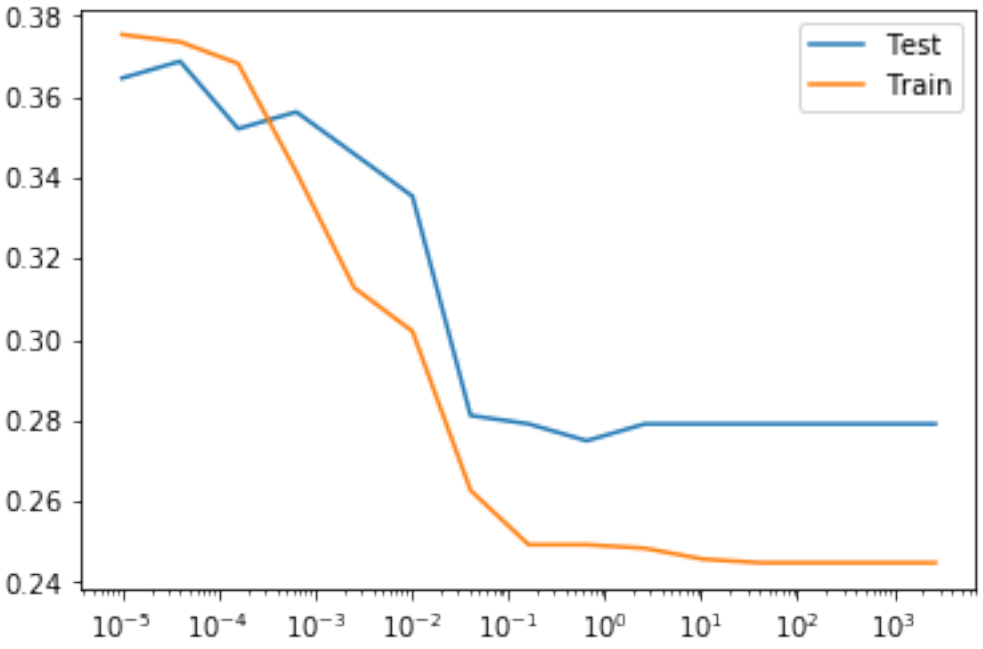
2 Problem 4 The effect of regularization

Using red wine dataset. Implement logistic regression in sklearn, using ℓ_2 regularization with regularizer value C in the set $\{0.00001 \times 4^i : i = 0, 1, 2, ..., 15\}$. (The regularization parameter is 'C' in scikit-learn, which is the inverse of λ we see in class). Plot the training error and test error with respect to the regularizer value. Explain what you get.

```
In [99]: N = np.array(range(0,15))
         alpha = 0.00001*(4**N)
         error_trn = np.zeros(15)
         error_tst = np.zeros(15)
         #========Your code here ======
         for i in range(len(N)):
             clf = LogisticRegression(random_state=0, C = alpha[i], solver='sag',multi_class='
             #training error
             y_red_pred_train = clf.predict(X_train_red)
             error_trn[i] = hamming_loss(y_train_red, y_red_pred_train)
             #testing error
             y_red_pred_test = clf.predict(X_test_red)
             error_tst[i] = hamming_loss(y_test_red, y_red_pred_test)
         #========================
```

```
plt.figure(1)
plt.semilogx(alpha, error_tst,label = 'Test')
plt.semilogx(alpha, error_trn, label = 'Train')
plt.legend()
```

```
Out[99]: <matplotlib.legend.Legend at 0x1a18bed668>
```




```
In [98]: alpha
```

```
Out[98]: array([1.00000000e-05, 4.00000000e-05, 1.60000000e-04, 6.40000000e-04,
                2.56000000e-03, 1.02400000e-02, 4.09600000e-02, 1.63840000e-01,
                6.55360000e-01, 2.62144000e+00, 1.04857600e+01, 4.19430400e+01,
                1.67772160e+02, 6.71088640e+02, 2.68435456e+03])
```

The traning error drop with increase of C, but the testing error remains similar since C = 0.1; Traning and also testing set error rate starts to grow when C = 1, it means lamda/C has limited effect on reducing the variance of our data set, choosing the right C will help to find the best model.