EE559 Week7 HW 1139093874 Pin-Hsuan Lee

1.(a) I use scikit-learn and Python.

(b) The normalizing factors should be calculated from the training data only and then applied to both the training data and test data because most of the time, we will not get the whole test dataset to calculate the mean and standard deviation. Besides, the training dataset has the same distribution as the testing data and it's the reason that why we can apply the calculation from the training data to both the training data and testing data.

The mean of each feature in original training data is

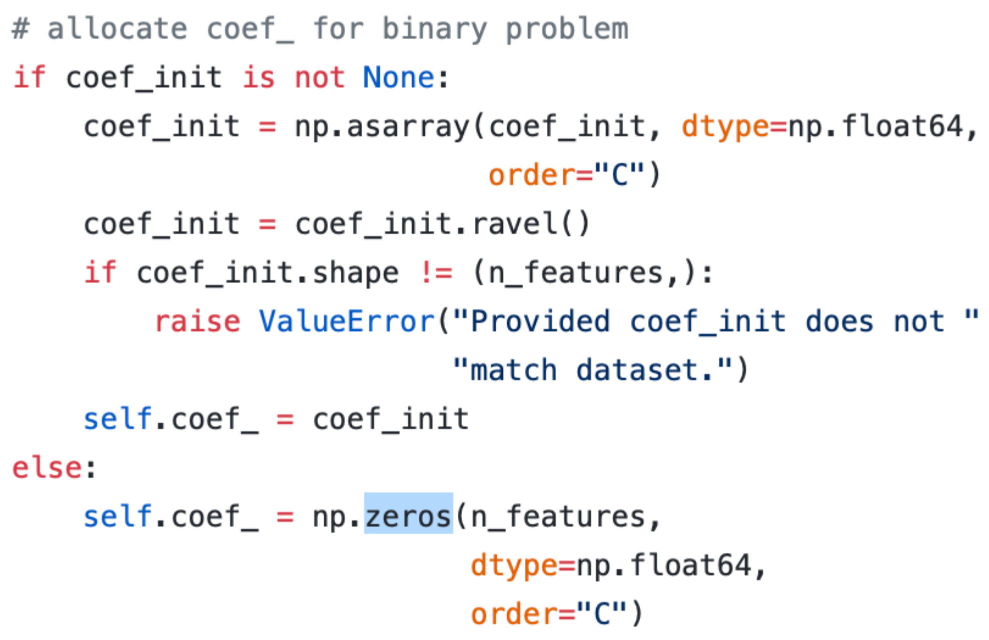
```
[1.29653933e+01 2.27000000e+00 2.37629213e+00 1.96494382e+01
 9.89101124e+01 2.27235955e+00 2.02943820e+00 3.60674157e-01
 1.57617978e+00 5.09123596e+00 9.53213483e-01 2.56033708e+00
 7.29707865e+02]
```

The standard deviation of each feature in original training data is

```
 [8.19560112e-01 1.10306417e+00 2.73004021e-01 3.46517583e+00
 1.15589748e+01 6.14548382e-01 9.19718276e-01 1.20241309e-01
 5.41470129e-01 2.40437795e+00 2.29907577e-01 7.23461482e-01
 3.07221225e+02]
```

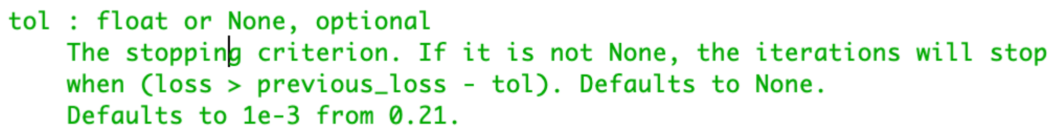
(c)

(i)The initial weight vector is a zero vector as we can see from the source code.



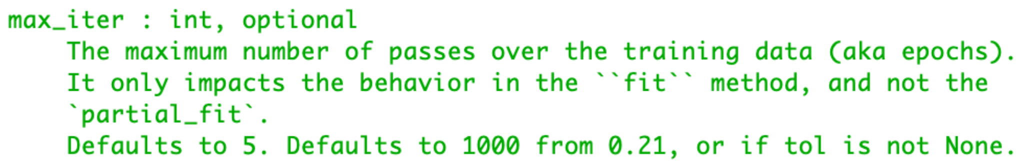
```
# allocate coef_ for binary problem
if coef_init is not None:
    coef_init = np.asarray(coef_init, dtype=np.float64,
                           order="C")
    coef_init = coef_init.ravel()
    if coef_init.shape != (n_features,):
        raise ValueError("Provided coef_init does not "
                         "match dataset.")
    self.coef_ = coef_init
else:
    self.coef_ = np.zeros(n_features,
                          dtype=np.float64,
                          order="C")
```

(ii)The halting condition is the variable in fit called ”tol”. When the gain of the accuracy is less than 0.001, it will halt.



```
tol : float or None, optional
    The stopping criterion. If it is not None, the iterations will stop
    when (loss > previous_loss - tol). Defaults to None.
    Defaults to 1e-3 from 0.21.
```

If the solution weight vector (which would correctly classify all training data points) is not reached, the backup halting condition is the number of epochs.Because the sklearn version installed in my computer is 0.20, the default epoch is 5.



```
max_iter : int, optional
    The maximum number of passes over the training data (aka epochs).
    It only impacts the behavior in the ``fit`` method, and not the
    `partial_fit`.
    Defaults to 5. Defaults to 1000 from 0.21, or if tol is not None.
```

(d)

When using only the first 2 features:

The accuracy for the first two features in training data is 0.7640449438202247

The accuracy for the first two features in testing data is 0.7528089887640449

The resulting 3 weight vectors for using the first two features are

```
[[ 2.05646082  0.29916664]
 [-2.19630015 -1.61368672]
 [ 1.28213477  3.23643909]]
```

When using only 13 features:

The accuracy for all features in training data is 1.0

The accuracy for all features in testing data is 0.9438202247191011

The resulting 3 weight vectors for using all features are

```
[[ 4.86476371  1.15133828  3.30200511 -1.22729962  1.8644008  -0.81068027
    3.56325529 -1.46895617  0.8142626  -1.8565385  -0.4955582   3.09855528
    4.76705126]
[-4.79991664 -4.34244909 -3.04436393  2.2071938  -3.57035669  2.14316511
    0.68315986  3.50979604 -0.55674479 -4.84794504  1.39029976  0.69251957
  -5.9952772 ]
[ 1.16203746  2.20295434  1.71335502  0.95135987  0.81847001  0.23457438
  -2.90672792 -3.37616962 -2.5724805   4.36595908 -3.83075284 -2.45759805
    1.84638021]]
```

(e)

For 2 features:

The best accuracy for two features in training data in 100 times run with different initial w vector is 0.8202247191011236 .
And the paired final vectors are

```
[[ 1.69958351 -0.86130147]
 [-1.99283307 -1.55294373]
 [ 0.64561537  0.9994913 ]]
```

The best accuracy result in training data happened in 59 th time and the paired accuracy for two features in testing data is 0.7865168539325843

For 13 features:
The best accuracy for all features in training data in 100 times run with different initial w vector is 1.
And the paired final vectors are

```
[[ 2.28013382e+00  1.94556950e+00  1.20422206e+00 -4.03854284e+00
   2.48717862e+00 -1.68206546e+00  2.89709697e+00 -1.25999091e+00
   6.30737674e-01 -1.58359746e+00  1.24337316e+00  3.27361095e+00
   5.79285365e+00]
 [-6.19454064e+00 -2.87820449e+00 -3.80220581e+00  1.21545183e+00
  -1.03234189e+00  1.57220676e+00  4.36696111e-01  2.04950607e+00
   9.07520860e-01 -5.88505804e+00  2.46972351e+00 -8.24718924e-01
  -5.79114857e+00]
 [ 2.37257264e+00  2.04570770e-01  8.02496365e-01 -2.05141183e-01
   1.35026099e+00  3.47654629e-01 -1.64706302e+00 -6.68140792e-01
   4.64146499e-03  4.17874906e+00 -2.31595428e+00 -1.75197396e+00
  -2.07251341e+00]]
```

The paired accuracy for all features in testing data is 0.9662921348314607 happened in 98 th time run.
(The best accuracy "1" happened for many times, therefore, we choose an index randomly to see the paired final vectors and the accuracy in testing data)

(f)

	(d)Accuracy for running once		**(e)Accuracy for running 100 times**	
	Training data	Testing data	Training data	Testing data
2 features	76.4%	75.3%	82%	78.7%
13 features	100%	94.4%	100%	96.6%

When compared 2 features and 13 features in training data and testing data, we can find that the accuracy in 13 features, no matter in training data or testing data, is better than 2 features. Because the classification is decided by many features rather than only one or two features, the more features can cause the better results. Also, we can find out the same situation when we compare two features to 13 features in accuracy for running 100 times. We can get better accuracy results if we have more features.

When we compared 2 features in running once to running 100 times, we can find that the accuracy runs a little bit better in running 100 times. Because we randomly generate the initial weight vectors from -1 to 1 which is very close to the initial default weight vector 0, these two accuracy results can be very close. When we compared 13 features in running once to running 100 times, we can find that it's the same situation as what happened in 2 features.

(g) The MSE(pseudo-inverse version) classification accuracy for the first 2 features on original training data is 0.7640449438202247
The MSE(pseudo-inverse version) classification accuracy for the first 2 features on original testing data is 0.7528089887640449
The MSE(pseudo-inverse version) classification accuracy for 13 features on original training data is 0.9887640449438202
The MSE(pseudo-inverse version) classification accuracy for 13 features on original testing data is 0.9775280898876404

(h) The MSE(pseudo-inverse version) classification accuracy for the first 2 features on standardized training data is 0.7640449438202247
The MSE(pseudo-inverse version) classification accuracy for the first 2 features on standardized testing data is 0.7528089887640449
The MSE(pseudo-inverse version) classification accuracy for 13 features on standardized training data is 0.9887640449438202
The MSE(pseudo-inverse version) classification accuracy for 13 features on standardized testing data is 0.9775280898876404

(i)

USE MSE	**(h)Accuracy in original data**		**(h)Accuracy in normalized data**	
	Training data	Testing data	Training data	Testing data
2 features	76.4%	75.2%	76.4%	75.2%
13 features	98.9%	97.8%	98.9%	97.8%

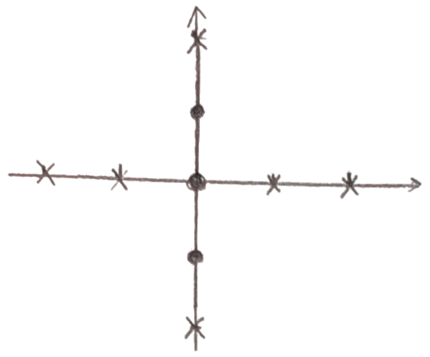
Compare the test accuracy results of (g) and (h), we can find that the accuracy getting from the original data will be the same as that getting from normalized data when using MSE. Because that MSE method works on the distribution of the data and that the original dataset has the same distribution as the normalized dataset, we can get the identical accuracy results.

(j)

	(h)Accuracy from using MSE to run		**(e) Accuracy from using perceptron to run 100 times**	
	Training data	Testing data	Training data	Testing data
2 features	76.4%	75.3%	82%	78.7%
13 features	98.9%	97.8%	100%	96.6%

Compare 2 features in using MSE to perceptron, we can find that these two results are both very high and close to each other.

2.



● class 1

X class 2

(a) From the plot, we can see that the two classes are not linearly separable.

(b)

$$\underline{\phi} = \begin{bmatrix} 1 \\ x_1 \\ x_2 \\ x_1x_1 \\ x_1x_2 \\ x_2x_2 \end{bmatrix} \qquad \underline{W} = \begin{bmatrix} W_{00} \\ W_{01} \\ W_{02} \\ W_{11} \\ W_{12} \\ W_{22} \end{bmatrix}$$

S_1:

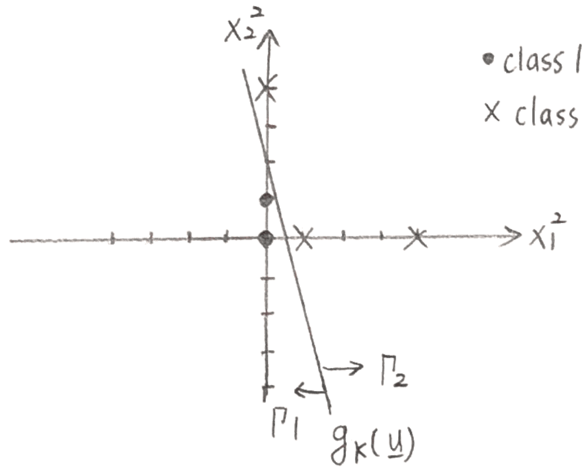
$$(0,0)^T \rightarrow \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \qquad (0,1)^T \rightarrow \begin{bmatrix} 1 \\ 0 \\ 1 \\ 0 \\ 0 \\ 1 \end{bmatrix} \qquad (0,-1)^T \rightarrow \begin{bmatrix} 0 \\ 0 \\ -1 \\ 0 \\ 0 \\ 1 \end{bmatrix}$$

S_2:

$$(-2,0)^T \rightarrow \begin{bmatrix} 1 \\ -2 \\ 0 \\ 4 \\ 0 \\ 0 \end{bmatrix} \qquad (-1,0)^T \rightarrow \begin{bmatrix} 1 \\ -1 \\ 0 \\ 1 \\ 0 \\ 0 \end{bmatrix} \qquad (0,2)^T \rightarrow \begin{bmatrix} 1 \\ 0 \\ 2 \\ 0 \\ 0 \\ 4 \end{bmatrix}$$

$$(0,-2) \rightarrow \begin{bmatrix} 1 \\ 0 \\ -2 \\ 0 \\ 0 \\ 4 \end{bmatrix} \qquad (1,0)^T \rightarrow \begin{bmatrix} 1 \\ 1 \\ 0 \\ 1 \\ 0 \\ 0 \end{bmatrix} \qquad (2,0)^T \rightarrow \begin{bmatrix} 1 \\ 2 \\ 0 \\ 4 \\ 0 \\ 0 \end{bmatrix}$$

(C)



● class 1

X class 2

$$\frac{x_1^2}{\frac{1}{2}} + \frac{x_2^2}{2} = 1 \Rightarrow 2x_1^2 + \frac{x_2^2}{2} = 1$$

$$g_K(u): 2x_1^2 + \frac{1}{2}x_2^2 - 1$$

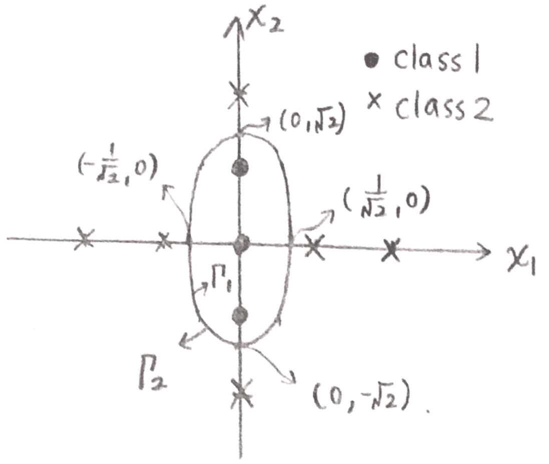
$g_K(\underline{u}) < 0 : \Gamma_1$

$g_K(\underline{u}) > 0 : \Gamma_2$

$$\underline{W}' = \begin{bmatrix} -1 \\ 0 \\ 0 \\ 2 \\ 0 \\ 1/2 \end{bmatrix}$$

$$g_K(u_i): \underline{W}'^T \underline{\phi} \begin{array}{l} < 0 : \Gamma_1 \\ > 0 : \Gamma_2 \end{array}$$

(d)



in expanded feature space : $g_k(\underline{u}) : 2x_1^2 + \frac{1}{2}x_2^2 - 1$

decision boundary : $g(\underline{x}) : 2x_1^2 + \frac{1}{2}x_2^2 - 1$ $< 0 : \Gamma_1$
(In original feature space) $> 0 : \Gamma_2$