ECE1513-Introduction to Machine Learning

<u>Assignment 2</u>

Zihao Jiao (1002213428)

Date: 2020.02.03

Problem 1 Let (w, b) ∈ R^d × R and x ∈ R^d. Assume that $||w||_2$= 1. Let us consider the point v defined by $v = x - (w \cdot \mathrm{x} + \mathrm{b})\mathrm{w}$.

1. Show that w · v + b = 0.
   Solution:
   Known that v = x − (w · x + b) w, substitute v into w · v + b = 0:
   $$\mathrm{w} \cdot \mathrm{v} + \mathrm{b} = \mathrm{w}[\mathrm{x} - (\mathrm{wx} + \mathrm{b})\mathrm{w}] + \mathrm{b}$$
   $$= w[x - (w^2x + wb)] + b]$$
   $$= wx - w^2(wx + b) + b$$
   Where $w^2 = ||w||_2^2 = 1$, Thus:
   $$\mathrm{w} \cdot \mathrm{v} + \mathrm{b} = wx - wx - b + b = 0$$

2. Using the previous question, prove that the following holds:
   $$\min_{\mathrm{u} \in R^d, s.t.\ w \cdot u + b = 0} ||x - u||_2 \leq |wx + b|$$
   Solution:
   Expand it: $||x - u||_2 = \sqrt{x^2 - 2xu - u^2} = f(x, u)$.
   In order to minimize it, we need to calculate $\frac{\delta f(x,u)}{\delta u} = 0$,
   Thus, $\frac{\delta f}{\delta u} = \frac{1}{2}(x^2 - 2xu - u^2)^{-0.5}(2u - 2x) = \frac{1}{2}((x - u)^2)^{-0.5}(2u - 2x) = 0$
   when $x = u$, above equation satisfied.
   The minimum value of $l1$ norm $|wx{+}b|$is 0, which its values all greater or equal than 0.
   Thus, $||x - u||_2 \leq |wx + b|$, when x$\neq u$.

3. Let u ∈ R^d such that $w \cdot u + b = 0$. Show that $||x - u||_2 \geq ||x - v||_2$.
   Solution:
   Known that v = x − (w · x + b) w and $||w||_2$= 1, rearrange it becomes:
   $$x - v = (wx + b)w.$$
   Take l_2 norm of above equation: $||x - v||_2 = |(wx + b)| \cdot ||w||_2$
   Since $w \cdot u + b = 0$, therefore, $wx + b - (wu + b) = wx + b$
   $$wx - wu = wx + b$$
   $$w(x - u) = wx + b \quad (1)$$
   Recall Cauchy-Schwartz Inequality: $|x^Ty| \leq ||x||_2 \cdot ||y||_2$
   Apply Cauchy-Schwartz Inequality into (1):
   $$|w^T(x - u)| \leq ||w||_2 \cdot ||x - u||_2$$
   $$|w^T(x - u)| \leq ||x - u||_2$$
   Thus, from above mathematical arranging:
   $$|w^T(x - u)| = |(wx + b)| = ||x - v||_2$$
   Therefore, $||x - u||_2 \geq ||x - v||_2$.

4. Use the above results to find the analytical expression for the l_2 distance between a point x and the hyperplane defined by w · u + b = 0 for all u ∈ R^d.
   Solution:
   Known that $w \cdot u + b = 0$, $w \cdot v + b = 0$; thus, $w \cdot u + b - (w \cdot v + b) = 0$
   $$so,\ \ w(v - u) = 0$$
   Therefore, w is an orthogonal vector to the plane $w \cdot u + b = 0$. So, ux is parallel to w.

$$|w \cdot \mathrm{ux}| = |w \cdot (\mathrm{u} - \mathrm{x})| \cos 0 = |w||xu| = ||w||_2$$
$$|w \cdot \mathrm{ux}| = |w \cdot (u - x)| = |wu - wx| = |-(b + wx)| = |b + wx|$$

Thus, l_2 distance from the point to the plane is:

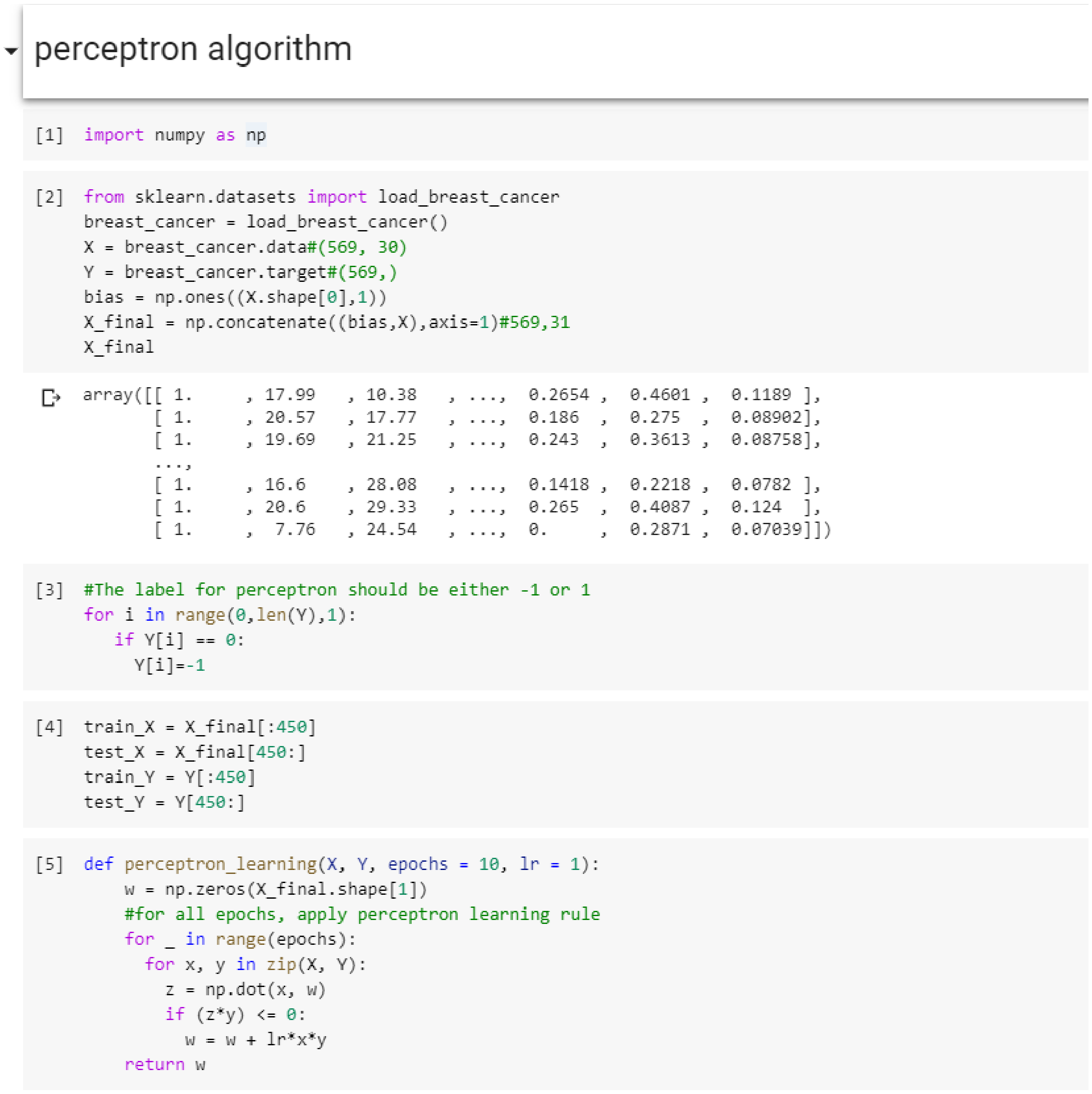
$$l_2 = \frac{|wx + b|}{||w||_2} = ||wx + b||$$

Problem 2 Recall the perceptron algorithm from class. It can be found in lecture notes posted on the course website.

1. Let us modify the update rule to add a learning rate α which multiplies the update applied to weights. The update rule becomes:$w' \leftarrow w + \alpha t^i x^i$. Show that the value of the learning rate does not change the perceptron's prediction after an update.
   Solution:
   Assume w is old weight, w' is new weight.
   $$w' = w + \alpha t^i x^i$$
   Substitute it into perceptron prediction formula:
   $$\begin{aligned} Z = w'^T x &= (w + \alpha t^i x^i)^T x \\ &= w^T x + \alpha t^i x^{iT} x \\ &= w^T x + \alpha t^i ||x||^2 \end{aligned}$$
   As we can tell from above equation, α (learning rate > 0) only rescales the weights but has no effect on the sign of the prediction.

2. Implement this modified perceptron algorithm in Python and NumPy. Include a copy of your code in your submission for this assignment.
   Solution:
   Please see the screen shots below.
3. Report the test error of your perceptron on the breast cancer dataset included with sklearn. Split the dataset (which includes 569 points) in 450 training examples and 119 test examples. One way to do that is as follows:
   Solution:
   Please see the screen shots below.
   Test error = 4.2017%

Attachments:



perceptron algorithm

```
[1]  import numpy as np
```

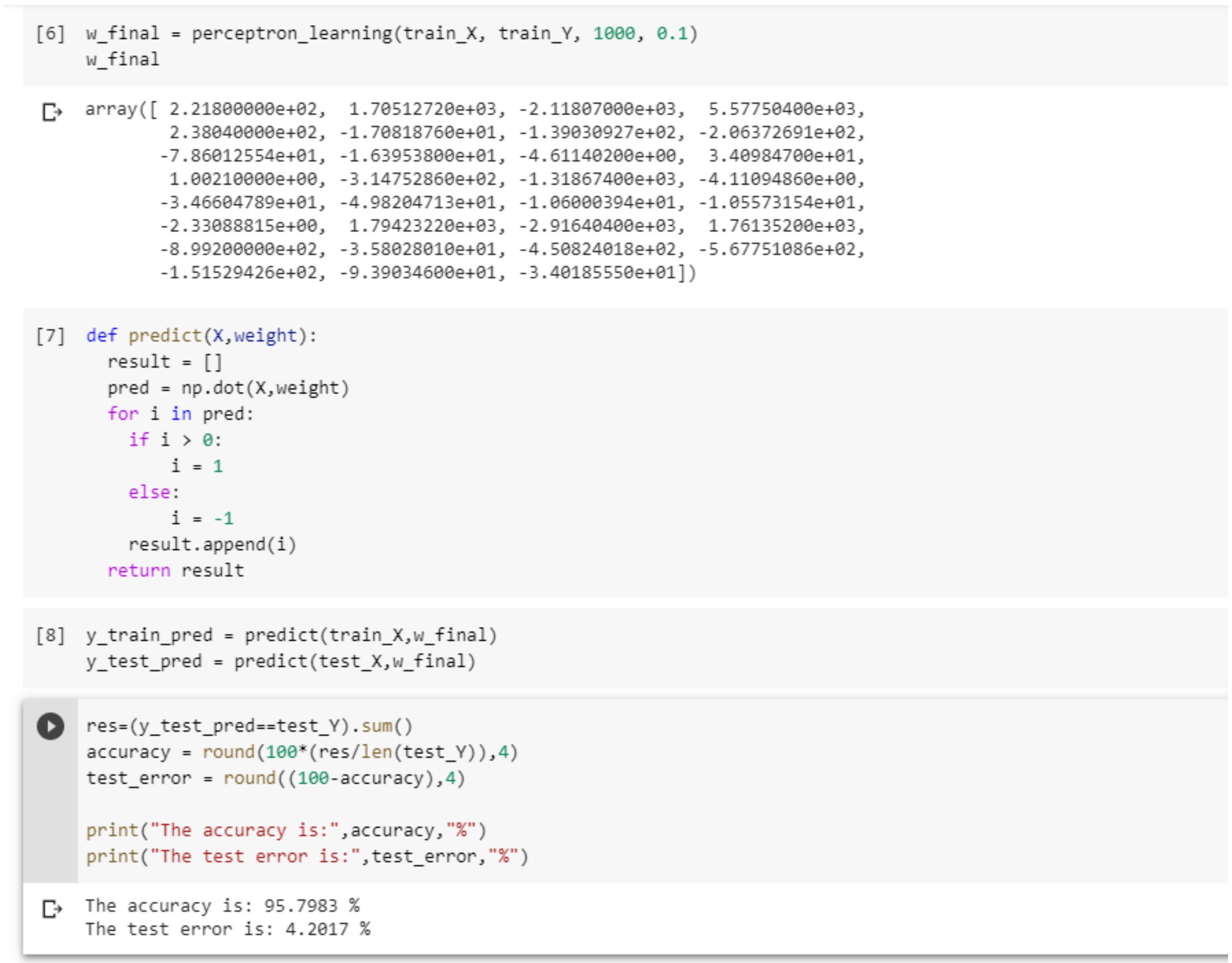
```
[2]  from sklearn.datasets import load_breast_cancer
     breast_cancer = load_breast_cancer()
     X = breast_cancer.data#(569, 30)
     Y = breast_cancer.target#(569,)
     bias = np.ones((X.shape[0],1))
     X_final = np.concatenate((bias,X),axis=1)#569,31
     X_final
```

```
array([[ 1.     , 17.99   , 10.38   , ...,  0.2654 ,  0.4601 ,  0.1189 ],
       [ 1.     , 20.57   , 17.77   , ...,  0.186  ,  0.275  ,  0.08902],
       [ 1.     , 19.69   , 21.25   , ...,  0.243  ,  0.3613 ,  0.08758],
       ...,
       [ 1.     , 16.6    , 28.08   , ...,  0.1418 ,  0.2218 ,  0.0782 ],
       [ 1.     , 20.6    , 29.33   , ...,  0.265  ,  0.4087 ,  0.124  ],
       [ 1.     ,  7.76   , 24.54   , ...,  0.     ,  0.2871 ,  0.07039]])
```

```
[3]  #The label for perceptron should be either -1 or 1
     for i in range(0,len(Y),1):
        if Y[i] == 0:
          Y[i]=-1
```

```
[4]  train_X = X_final[:450]
     test_X = X_final[450:]
     train_Y = Y[:450]
     test_Y = Y[450:]
```

```
[5]  def perceptron_learning(X, Y, epochs = 10, lr = 1):
         w = np.zeros(X_final.shape[1])
         #for all epochs, apply perceptron learning rule
         for _ in range(epochs):
           for x, y in zip(X, Y):
             z = np.dot(x, w)
             if (z*y) <= 0:
               w = w + lr*x*y
         return w
```



```
[6] w_final = perceptron_learning(train_X, train_Y, 1000, 0.1)
    w_final
```

```
array([ 2.21800000e+02,  1.70512720e+03, -2.11807000e+03,  5.57750400e+03,
        2.38040000e+02, -1.70818760e+01, -1.39030927e+02, -2.06372691e+02,
       -7.86012554e+01, -1.63953800e+01, -4.61140200e+00,  3.40984700e+01,
        1.00210000e+00, -3.14752860e+02, -1.31867400e+03, -4.11094860e+00,
       -3.46604789e+01, -4.98204713e+01, -1.06000394e+01, -1.05573154e+01,
       -2.33088815e+00,  1.79423220e+03, -2.91640400e+03,  1.76135200e+03,
       -8.99200000e+02, -3.58028010e+01, -4.50824018e+02, -5.67751086e+02,
       -1.51529426e+02, -9.39034600e+01, -3.40185550e+01])
```

```
[7] def predict(X,weight):
      result = []
      pred = np.dot(X,weight)
      for i in pred:
        if i > 0:
            i = 1
        else:
            i = -1
        result.append(i)
      return result
```

```
[8] y_train_pred = predict(train_X,w_final)
    y_test_pred = predict(test_X,w_final)
```

```
res=(y_test_pred==test_Y).sum()
accuracy = round(100*(res/len(test_Y)),4)
test_error = round((100-accuracy),4)

print("The accuracy is:",accuracy,"%")
print("The test error is:",test_error,"%")
```

```
The accuracy is: 95.7983 %
The test error is: 4.2017 %
```