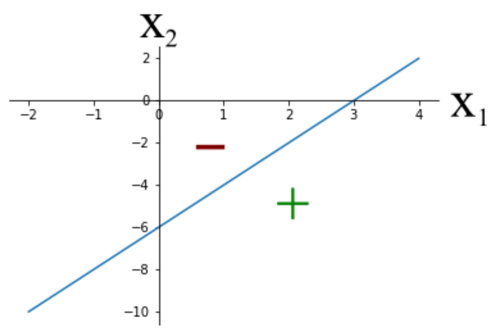
DSC 255: Machine learning

Week 6 — Solutions

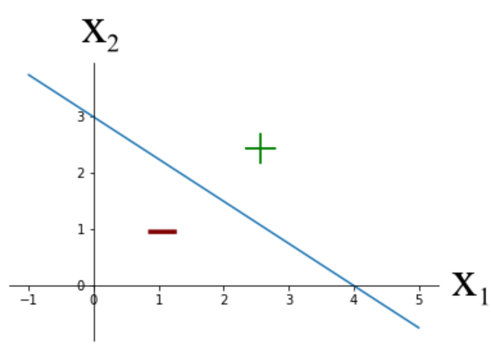
1. The decision boundary plot should look something like the plot below.



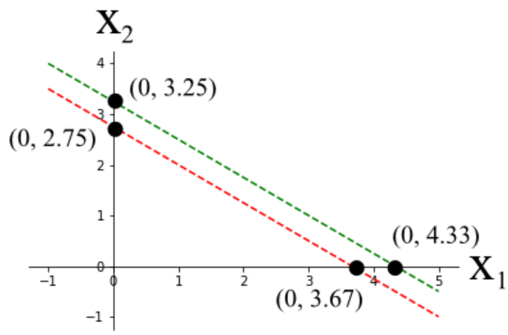
2. (a) **Definitely true.** If the data set were not linearly separable, Perceptron would never converge.
   (b) **Definitely true.** Since the data is linearly separable, Perceptron is guaranteed to converge, no matter what the ordering of the points might be.
   (c) **Possibly false.** Different orderings of the data can produce different numbers of updates before convergence. We saw examples of this in class.
   (d) **Possibly false.** There could be several updates on any given data point, and thus k is not necessarily upper-bounded by n.

3. Each time the Perceptron algorithm performs an update a point with label y, it updates its offset b as $b = b + y$. Thus if we start with $b = 0$ and perform p updates on points with $y = -1$ and q updates on points with $y = +1$, then the final value of b is $b = q - p$.

4. (a) The decision boundary plot should look something like the plot below.



(b) The left- and right-hand boundary plot should look something the plot below.



(c) The margin of this classifier is

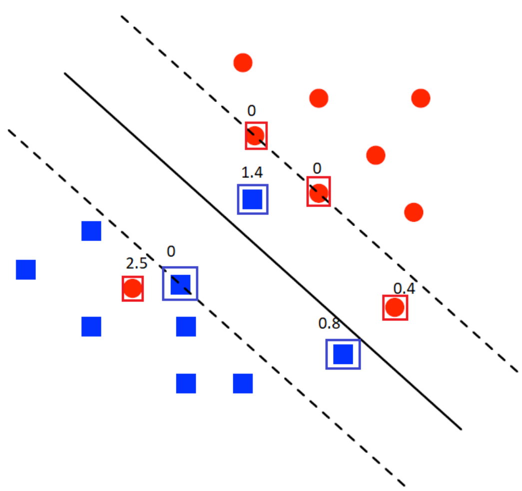
$$\gamma = \frac{1}{\|w\|} = \frac{1}{\sqrt{3^2+4^2}} = \frac{1}{5}.$$

(d) The point $x = (2, 2)$ satisfies

$$w \cdot x + b = 6 + 8 - 12 = 2 > 0$$

Thus this point would be classified as $+1$.

5. (a) Support vectors and their respective slack variables are marked in figure.



(b) The margin decreases if the factor C is increased.

6. (a) Possibly false. α_i is the number of updates on point i, and can be greater than 1.

(b) Necessarily true. The total number of updates performed is $\sum_i \alpha_i$.

(c) Necessarily true. There are k updates, so these can involve at most k different training points.

(d) Necessarily true. If the training data were not linearly separable, Perceptron would never halt.

7. *Perceptron project.*

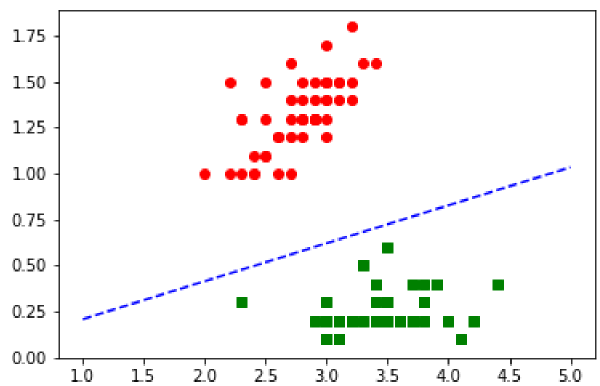
(a) The classification code can be written as follows.

```
def classify(w, b, x):
    return np.sign(np.dot(w,x) + b)
```

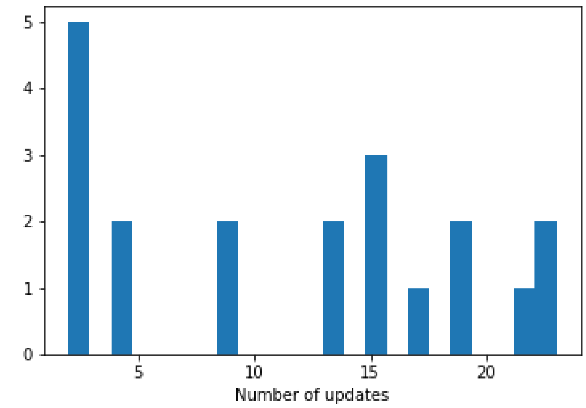
The perceptron algorithm can be written as follows.

```
def perceptron(data, labels):
    n = len(labels)
    inds = np.random.permutation(n)
    data = data[inds,:]
    labels = labels[inds]
    n_correct = 0
    w = np.zeros(np.shape(data)[1])
    b = 0
    while(n_correct < n):
        n_correct = 0
        for i in range(n):
            if (classify(w, b, data[i,:]) == labels[i]):
                n_correct += 1
            else:
                w = w + labels[i]*data[i,:]
                b = b + labels[i]
    return(w,b)
```

(c) The perceptron boundary should look something like the following plot.

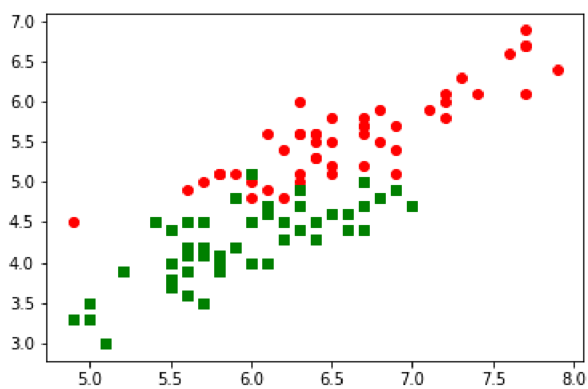


(d) The histogram should look something like the following.



8. *Support vector machine.*

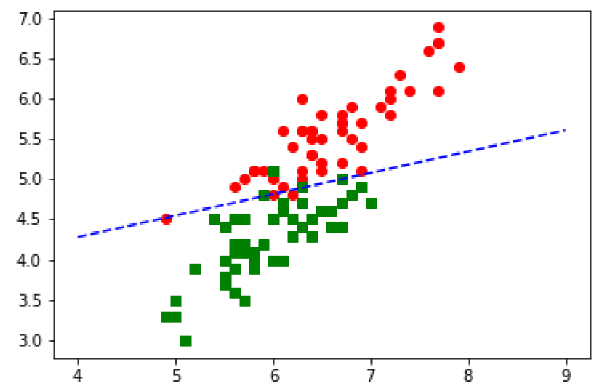
   (a) The data is not linearly separable. We can see this by inspecting the scatter plot.



   (b) The table you produce should look something like the following.

C value	1.5	3.0	4.5	6.0	7.5	9.0	10.5	12.0	13.5	15.0
Training error	0.07	0.05	0.06	0.05	0.05	0.05	0.05	0.04	0.05	0.07
# of support vectors	27	22	21	19	19	19	18	17	16	16

   (c) The boundary plot should look something like the following.


7.0
6.5
6.0
5.5
5.0
4.5
4.0
3.5
3.0
4
5
6
7
8
9