Shengjie Li
netID: sl1560

CS 536 : Perceptrons

1. Show that there is a perceptron that correctly classifies this data. Is this perceptron unique? What is the best perceptron for this data set, theoretically?

   (a) Because the class of each data point is only determined by the value of the X_k feature, it's clear that a perceptron can tell if X_k is greater than 0 or less than 0.

   (b) There are infinite perceptrons that can correctly classifies this data. In fact, if we let $a = max_i(X_k^i), \text{s.t. } Y^i == -1$ and $b = min_i(X_k^i), \text{s.t. } Y^i == 1$, every hyperplane $a <= x_k <= b$ can correctly classify this data.

   (c) Theoretically, the best perceptron would be a hyperplane of $x_k = 0$. A hyperplane of $x_k = \frac{a+b}{2}$ can classify the training data very well, but it would fail to generalize for data points with $X_k \in (a, 0) \bigcup (0, b)$.

2. We want to consider the problem of learning perceptrons from data sets. Generate a set of data of size m = 100 with $k = 20, \epsilon = 1$.

   - Implement the perceptron learning algorithm. This data is separable, so the algorithm will terminate. How does the output perceptron compare to your theoretical answer in the previous problem?

     The perceptron I get is:

```
w = [[  0.        ]
     [  0.02087407]
     [  0.17257196]
     [-0.55194861]
     [  0.97983356]
     [-0.85764329]
     [  0.85379284]
     [-0.89753315]
     [-2.63630402]
     [  0.3010298 ]
     [  1.77413824]
     [-1.21897327]
     [-1.87519926]
     [  0.64459163]
     [  1.12742088]
     [-0.18981816]
     [  2.24033418]
     [-0.9949524 ]
     [  2.1984183 ]
     [  0.9931059 ]
     [  7.35863809]]
```

The first element of my $\underline{w}$ is the bias unit. The margin of my output perceptron is 0.02223443.

While the theoretically best perceptron is:

```
w =  [ [ 0 . ]
       [ 0 . ]
       [ 0 . ]
       [ 0 . ]
       [ 0 . ]
       [ 0 . ]
       [ 0 . ]
       [ 0 . ]
       [ 0 . ]
       [ 0 . ]
       [ 0 . ]
       [ 0 . ]
       [ 0 . ]
       [ 0 . ]
       [ 0 . ]
       [ 0 . ]
       [ 0 . ]
       [ 0 . ]
       [ 0 . ]
       [ 0 . ]
       [ 1 . ] ]
```

The first element of my $\underline{w}$ is the bias unit. The margin of the theoretically best perceptron is 1.00739826.

Thus, my output perceptron has a smaller margin than the theoretically best perceptron, which means it might potentially fail to generalize well in other data sets, which is not a good thing.

3. For any given data set, there may be multiple separators with multiple margins - but for our data set, we can effectively control the size of the margin with the parameter ϵ - the bigger this value, the bigger the margin of our separator.

   - For $m = 100, k = 20$, generate a data set for a given value of ϵ and run the learning algorithm to completion. Plot, as a function of $\epsilon \in [0, 1]$, the average or typical number of steps the algorithm needs to terminate. Characterize the dependence.

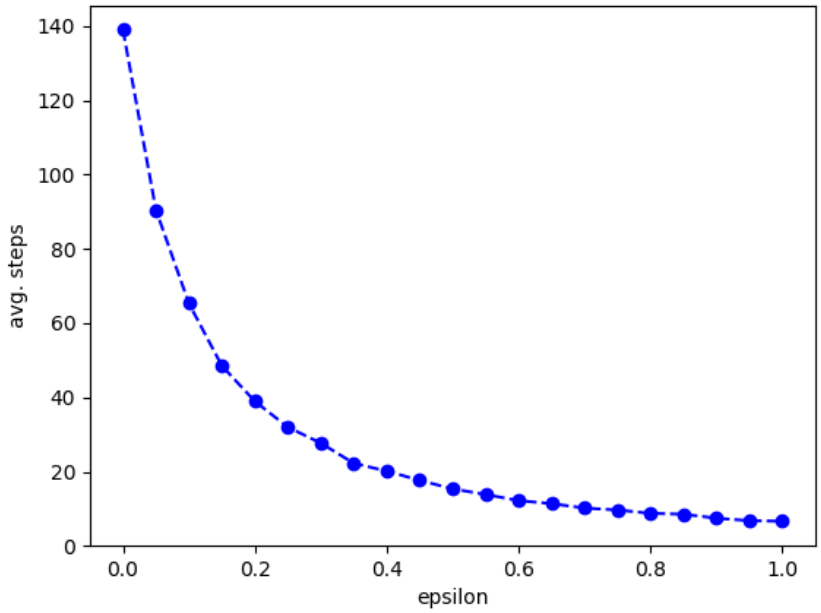

Figure 1: Average number of steps the algorithm needs to terminate

The bigger the ϵ is, the farther the data points would be separated. It would be easier for the learning algorithm to find a linear separator for larger ϵ since the margin is bigger. Thus, with the increase of ϵ, the average steps of the learning algorithm would first decrease, then start to level out since it would be nearly an instant fit. The shape of the plot looks like half of a hyperbola.

4. One of the nice properties of the perceptron learning algorithm (and perceptrons generally) is that learning the weight vector $\underline{w}$ and bias value b is typically independent of the ambient dimension. To see this, consider the following experiment:

   - Fixing $m = 100, \epsilon = 1$, consider generating a data set on k features and running the learning algorithm on it. Plot, as a function k (for $k = 2, \ldots, 40$), the typical number of steps to learn a perceptron on a data set of this size. How does the number of steps vary with k? Repeat for $m = 1000$.

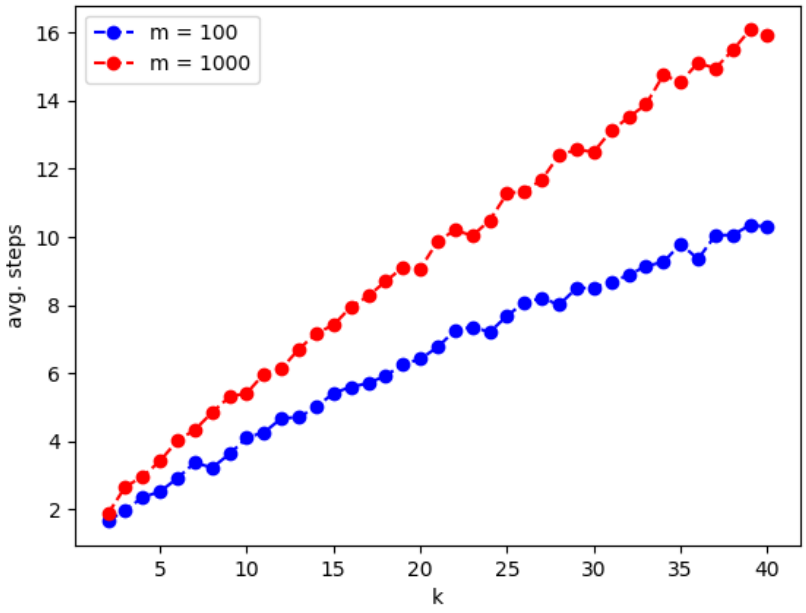

Figure 2: Average number of steps the algorithm needs to terminate

As shown in the Fig 2, with the increase of k, the average steps the algorithm needs to terminate slowly increases. The increase of average steps is generally linear.

5. As shown in class, the perceptron learning algorithm always terminates in finite time - if there is a separator. Consider generating non-separable data in the following way: generate each $X_1, \ldots, X_k$ as i.i.d. standard normals $N(0, 1)$. Define Y by

$$Y = \begin{cases} +1 & \text{if } \sum_{i=1}^{k} X_i^2 \geq k \\ -1 & \text{else.} \end{cases}$$

For data defined in this way, there is no universally applicable linear separator.

For $k = 2, m = 100$, generate a data set that is not linearly separable. (How can you verify this?) Then run the perceptron learning algorithm. What does the progression of weight vectors and bias values look like over time? If there is no separator, this will never terminate - is there any condition or heuristic you could use to determine whether or not to terminate the algorithm and declare no separator found?

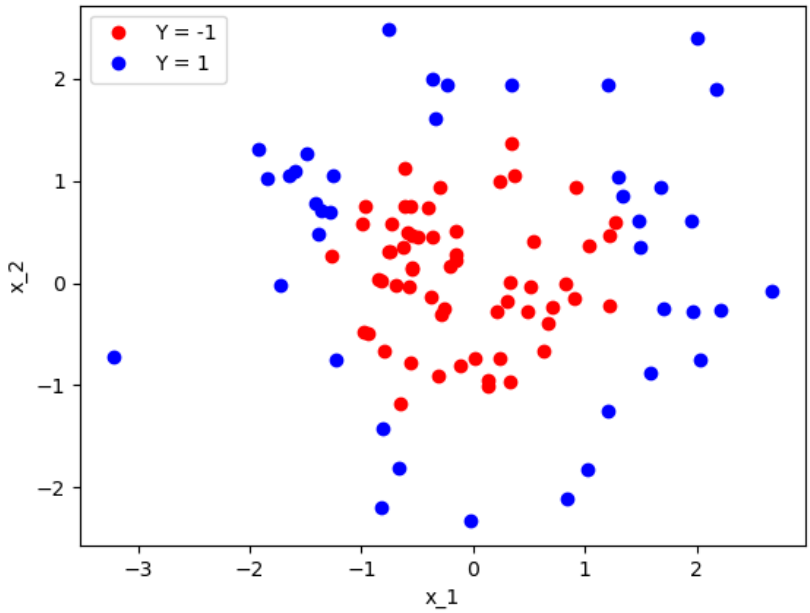

Figure 3: Data set

There are several methods to verify if the data is linearly separable.

(a) If the data has intersecting convex hulls, then the data is not linearly separable [1]. But this is hard to tell if the data is high dimensional.

(b) If the learning algorithm runs over a certain number of steps, then the data is not linearly separable.

Since the data is only 2-dimensional, I am going to plot my data here.

As Fig 3 shows, there is clearly no linear separator that can separate this data.

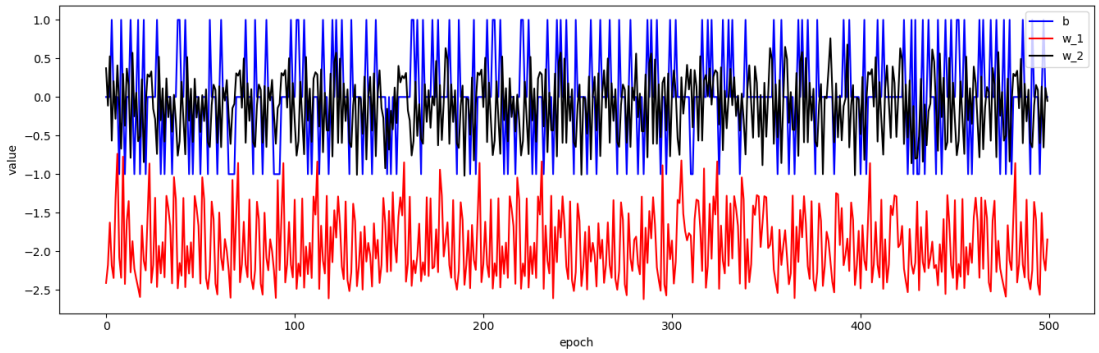

Figure 4: The progression of weight vectors and bias values

From the Fig 4 above we can see that the values are always swaying.

I propose a heuristic:

(a) Set the max epoch to be a large enough number, say 10000.

(b) Record the number of misclassified data points, say it's p.

(c) Update max epoch as $min(max\_epoch, current\_epoch + 2 * p)$

If the program runs over the *max_epoch*, I would say there's no separator for the data.

References

[1] D. Holmes, "Convex hulls solve svms." `http://web.mit.edu/dxh/www/convex.pdf`. [Online; accessed 27-February-2019].