Introduction to Machine Learning - 236756

Homework #2

Alexander Shender 328626114

Snir Hordan XXXXXXXXX

Technion - Israel Institute of Technology

I. Part 2 - Dry Assignment

A. Part 2a

The halfspace is homogeneous if the hyperpalce which defines them passed through the origin. Subsequently, the non-homogeneous halfspaces have to be defined by a hyperplance which is displaced by some parameter b (bias) from the origin.

We include this bias b term in the parameters w, thus bringing w to $R^(d+1)$, which is denoted $z \in R^(d+1)$ in the assignment. But this term doesn't have to be multiplied by any input, thus we simply add '1' to the input vector x. The answer is thus:

$$g : R^d \to R^{d+1} = (x, 1)$$

$$z : R^d \to R^{d+1} = (w, b)$$

B. Part 2b

First, some general notes on Datasets:

- the datasets A and C are not linearly separable in the original dimension
- the dataset B is linearly separable in original dimension

Some general notes on models:

- Perceptron can only classify linearly separable cases
- KNN can classify non-linear datasets, non-parametric
- Decision trees can classify non-linear datasets as well

YES = can fit NO = cannot fit

1. Perceptron
    - A: NO. The dataset is not linearly separable. This is actually similar to the XOR function, which cannot be learned by any linear model.
    - B:YES. The dataset is easily linearly separable.
    - C: NO. The dataset is not linearly separable. Only using the Kernel Trick (e.g. polynomial of 2nd order) can bring the dataset to a higher dimension, where they can be separated by a hyperplane.
2. KNN, $k = 1$
    - A: NO. As a simple example, it can be seen that on the far left on the border between the two groups there exists an orange dot really close to the blue ones. Using KNN with $k = 1$ it would classify this point as belonging to the blue group.
    - B: NO. The following point (orange) will be classified wrong, since its closest point is blue.
    - C: YES. From visual observation, all points would be classified correctly.
3. KNN, $k = 3$
    - A: NO. Same point as in the case with $k = 1$, this point would have a majority of 2 blue points in it's proximity, resulting in a wrong classification.
    - B: YES. From visual observation.
    - C: YES. From visual observation.

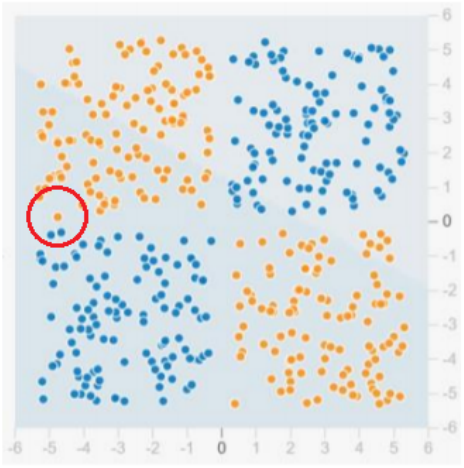

Figure 1: wrongly classified point at k=1

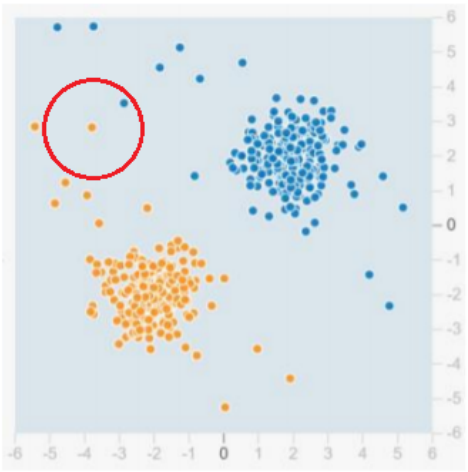

Figure 2: wrongly classified point at k=1

4. Decision tree

- A: YES. The decision tree can classify non-linear datasets, and in this example, it would require exactly 4 leaves to classify all the training dataset. It may look like this:

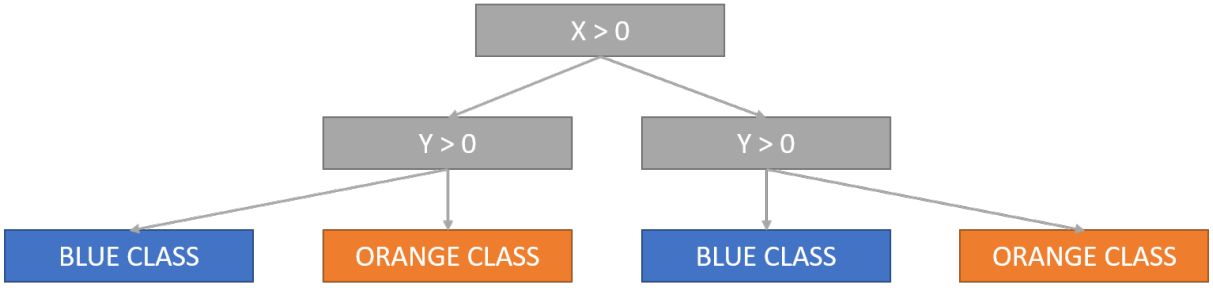

Figure 3: Decision tree for Dataset A with 4 leaves

- B: By having the limitation of 4 leaves, this dataset cannot be classified with 0 training error.
- C: Same as with B. By having limitless Decision tree depth this would be possible

C. Part 3.1

To find the derivative with respect to the model parameters $w_j, j = 1...d$, we first write the cost function explicitly for both cases of the classification, since the cost function is defined diferently for the case where $y = 0$ and $y = 1$.

First, we denote shortly the sigmoid function with parameters w:

$$\sigma_w(x) = \frac{1}{1 - e^{-w^T x}}$$

Thus, we can rewrite the cost function:

$$C = cost(x_i, y_i, w) = y_i \cdot log(\sigma(x_i, w)) + (1 - y_i) \cdot log(1 - \sigma(x_i, w))$$

This way, the left side is active for the case with $y = 1$, and the right side for the case where $y = 0$. We also take y_i out of the sigmoid function. It's worth noticing that the function is always negative except of the point where it is equal zero, when all predictions are perfectly correct. Thus, this is called the gradient ascend, since we are trying to "climb" to increase the "cost" function. Here the term "cost" has the opposite measning, i would better denote it "reward", since we are trying to maximize it.

Now we can calculate the derivatives with regards to the model parameters w. We summarize over all of the samples in a set.

Remembering:

$$\frac{\delta}{\delta w} log(f(w)) = \frac{1}{f(w)}(\frac{\delta}{\delta w} f(w))$$

$$\frac{\delta C}{\delta w_j} = \frac{1}{m} \cdot \sum_{i=1}^{m}((y_i) \cdot \frac{1}{\sigma(x_i, w)} \cdot \frac{\delta \sigma(x_i, w)}{\delta w_{x_i}}) + ((1 - y_i) \cdot \frac{1}{(1 - \sigma(x_i, w))} \cdot \frac{\delta(1 - \sigma(x_i, w))}{\delta w_{x_i}})$$

We calculate the partial derivatives of the sigmoid function with respect to the model parameters using the chain rule and the hint in the assignment.

$$\frac{\delta \sigma(x_i, w)}{\delta w_{x_i}} = \frac{\delta \sigma(x_i, w)}{\delta(x_i, w)} \cdot \frac{\delta(x_i, w)}{\delta(w)} = \sigma(x_i, w) \cdot (1 - \sigma(x_i, w)) \cdot x_{i_j}$$

where x_{i_j} is the j'th parameter of the sample x_i

inserting back into the equation:

$$\frac{\delta C}{\delta w_j} = \frac{1}{m} \cdot \sum_{i=1}^{m}((y_i) \cdot \frac{\sigma(x_i, w) \cdot (1 - \sigma(x_i, w))}{\sigma(x_i, w)} \cdot x_{i_j} - ((1 - y_i) \cdot \frac{\sigma(x_i, w) \cdot (1 - \sigma(x_i, w))}{(1 - \sigma(x_i, w))} \cdot x_{i_j})$$

Shortening:

$$\frac{\delta C}{\delta w_j} = \frac{1}{m} \cdot \sum_{i=1}^{m} (y_i \cdot (1 - \sigma(x_i, w)) \cdot x_{i_j} - ((1 - y_i) \cdot \sigma(x_i, w) \cdot x_{i_j})$$

$$= -\frac{1}{m} \cdot \sum_{i=1}^{m} (y_i x_{i_j} - y_i x_{i_j} \sigma(x_i, w) - \sigma(x_i, w) x_{i_j} + y_i x_{i_j} \sigma(x_i, w) =$$

$$= -\frac{1}{m} \cdot \sum_{i=1}^{m} (y_i x_{i_j} - \sigma(x_i, w) x_{i_j})$$

$$\frac{\delta C}{\delta w_j} = \frac{1}{m} \cdot \sum_{i=1}^{m} (x_{i_j} \cdot (y_i - \sigma(x_i, w))$$

We are used to the matrix form. Converting the expression to matrix form:

$$\frac{\delta C}{\delta w} = \frac{1}{m} X \cdot (y - \sigma(X, w))$$

where m is the number of samples in a set.

D. Part 3.2

Since this is the gradient ascent algorithm, we do want to go in the direction of the steepest ascent, which is exactly the gradient. So, we update our parameters in the following way:

$$w^{t+1} = w^t + \eta \cdot \frac{1}{m} \cdot \sum_{i=1}^{m} (x_{i_j} \cdot (y_i - \sigma(x_i, w^t))$$

or in a matrix form:

$$w^{t+1} = w^t + \eta \cdot \frac{1}{m} X \cdot (y - \sigma(X, w^t))$$

where :

- w^{t+1} - the parameters at the next iteration
- w^t - the parameters are the current iteration
- η - learning rate