CS480/680: Introduction to Machine Learning

Homework 1

Due: 11:59 pm, January 28, 2021, submit on Crowdmark (yet to be set up, stay tuned).

Exercise 1: Perceptron Implementation (5 pts)

Convention: All algebraic operations, when applied to a vector or matrix, are understood to be element-wise (unless otherwise stated).

Algorithm 1.1: The perceptron algorithm.

Input: $X \in \mathbb{R}^{n\times d}$, $\mathbf{y} \in \{-1,1\}^n$, $\mathbf{w} = \mathbf{0}_d$, $b = 0$, $\mathsf{max\_pass} \in \mathbb{N}$

Output: $\mathbf{w}, b, mistake$

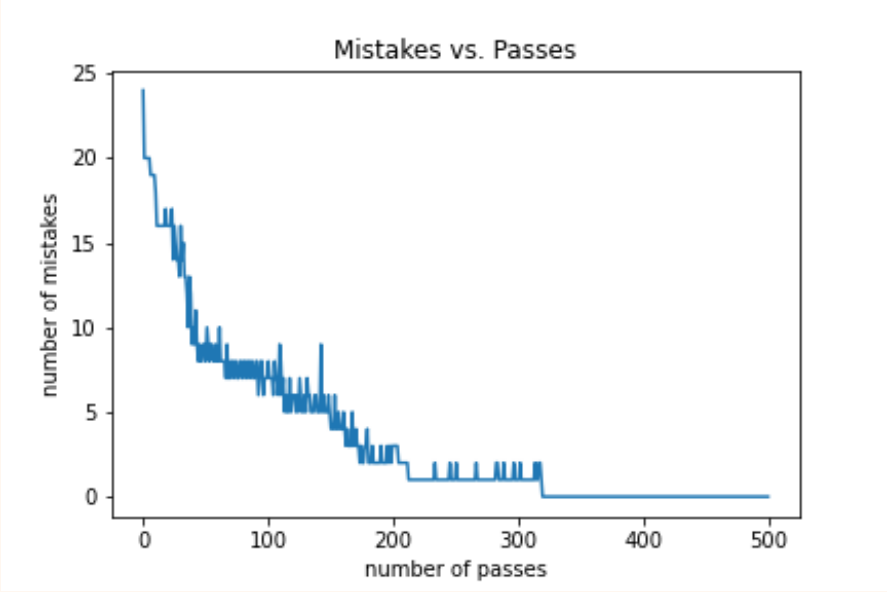
for $t = 1,2,\ldots,\mathsf{max\_pass}$ **do**
  $mistake(t) \leftarrow 0$
  for $i = 1,2,\ldots,n$ **do**
    if $y_i(\langle \mathbf{x}_i, \mathbf{w}\rangle + b) \leq 0$ **then**
      $\mathbf{w} \leftarrow \mathbf{w} + y_i\mathbf{x}_i$ // $\mathbf{x}_i$ is the i-th row of X
      $b \leftarrow b + y_i$
      $mistake(t) \leftarrow mistake(t) + 1$

Implement the perceptron in Algorithm 1.1. Your implementation should take input as $X = [\mathbf{x}_1^\top, \ldots, \mathbf{x}_n^\top]^\top \in \mathbb{R}^{n\times d}$, $\mathbf{y} \in \{-1,1\}^n$, an initialization of the hyperplane parameters $\mathbf{w} \in \mathbb{R}^d$ and $b \in \mathbb{R}$, and the maximum number of passes of the training set [suggested $\mathsf{max\_pass} = 500$]. Run your perceptron algorithm on the spambase dataset (use the version on the course website), and plot the number of mistakes (y-axis) w.r.t. the number of passes (x-axis).

Ans: **Answer 1.1**

Answer 1.1

Resultant image:



Perceptron Implementation:

```python
    def perceptron(
        X: List[List[float]],
        y: List[int],
        max_pass=500
    )-> [List[float], float, List[int]]:
        """
        @param          X: \in R^{nxd}
        @param          y: \in {-1,1}^n
        @param   max_pass: \in N
        """
        X = np.array(X)
        y = np.array(y)
        [n, d] = np.shape(X)
        w = [0] * d # w = 0_d
        b = 0
        mistake = []
        for t in range(0, max_pass): # max passes / iterations
            mistake.append(0)
            for i in range(0, n):
                x_i = X[i, :]
                if (y[i] * (np.dot(x_i, w) + b)) <= 0:
                    w = w + y[i] * x_i
                    b = b + y[i]
                    mistake[t] += 1

        return w, b, mistake

```