CS 260: Machine Learning **Spring 2020**

Homework 3

Student Name: Andrew Choi *UID: 205348339*

- Feel free to talk to other students in the class when doing the homework. You should, however, write down your solution yourself. You also must indicate on each homework with whom you collaborated and cite any other sources you use including Internet sites.
- You will write your solution in LaTeX and submit the **pdf file through Gradescope**. You also need to submit the **zipped LaTeX files to CCLE**. We will grade your homework based on the final version of the pdf file submitted to Gradescope. We will not grade the zipped Latex files on CCLE. However, failure to submitting your LaTax files to CCLE will incur 2 points penalty out of 100 points.
- The homework (both pdf and zipped Latex source files) is due at **1:59 PM before the class**.

1. Exercise **9.1**

   Define a vector $g = (g_1, ..., g_m)$. Using the provided hint and the absolute value loss function, we can arrive at the following constraints.

   $$w^T x_i - g_i \leq y_i, -w^T x_i - g_i \leq -y_i \ \forall i \in [m]$$

   Then minimizing the empirical risk is equivalent to minimizing $\sum_{i=1}^{m} g_i$ under the above constraints. We can then define the following matrices and vectors necessary to solve the problem. Let $A \in \mathbb{R}^{2m \times (m+d)}$ be the matrix $[X - I; -X - I]$. Here, the ith row in X should be the entry x_i and I is the identity matrix. Next, let $v \in \mathbb{R}^{m+d}$ be the vector $(w_1, ...w_d, g_i, ...g_m)$. Then, let $b \in \mathbb{R}^{2m}$ be the vector $(y_1, ...y_d, -y_1, ... - y_d)^T$. Finally, let $c \in \mathbb{R}^{m+d}$ be the vector consisting of d first elements of 0 and m next elements of 1. Then the following linear program can be formulated that solves the ERM problem of absolute value loss linear regression.

   $$\min c^T v \quad \text{s.t} \quad Av \leq b$$

2. Exercise **9.3**

   Following the hint, we choose $d = m$ and set $x_i = e_i \ \forall i \in [d]$ which satisfies the the first condition that $R = \max_i ||x_i|| \leq 1$. Next, consider that $y_i = 1 \ \forall i \in [d]$. Then, it is intuitive to see that $w^* = \{1\}^d$ which also satisfies the conditions $||w^*||^2 = m$ and $\forall i \ y_i(x_i \bullet w^*) \geq 1$. Finally, if $w^{(1)} = \{0\}^d$, then it will take exactly m iterations of the perceptron algorithm to converge since there must be an update for each dimension which satisfies the final condition. This holds for any $d = m > 0$.

3. Exercise **9.5**

   Firstly, note that the labeling of the Perceptron algorithm depends on $\text{sign}(w^{(t)} \bullet x_i)$. If $\eta > 0$ and the Perceptron update is modified to $w^{(t+1)} = w^{(t)} + \eta y_i x_i$, then the result is modified to $\text{sign}(\eta w^{(t)} \bullet x_i)$. Note that the following are equivalent.

   $$\text{sign}(w^{(t)} \bullet x_i) = \text{sign}(\eta w^{(t)} \bullet x_i)$$

Therefore, the results of the labels will be equivalent. Furthermore, the resulting vector w^* will be in the same direction as the output of the vanilla Perceptron algorithm, w^v, as η simply increases the magnitude of w^* relative to w^v. In other words, w^* and w^v are equivalent unit vectors and thus the constant scale η has no effect on the direction of $w^{(T)}$ for all time steps T for the Perceptron algorithm. Following this, it is easy to see that the number of iterations until convergence will also stay the same regardless of the inclusion of η due to the above statements.

4. Exercise **18.1**

   a. Note that each level of a decision tree either contains a leaf or a split dependent on the feature $x_i \ \forall i \in [d]$ where a leaf only occurs if the labels for the subset of training examples that satisfy the same sequence of features leading up to the feature in question are identical. In this case, it can be seen that the height of the tree would be upper bound by the case in which only splitting occurs until all features are used up. Intuitively, it can then be seen that d features would result in d levels as well as an additional level for the labels themselves resulting in the upper bound of $d + 1$.

   b. Let us consider a decision tree with the maximum possible height as described in previously in part a. Such a tree has 2^d leaves with d levels of nodes above them. Under such a tree, we can assign a leaf to any possible combination of feature nodes which implies that we can shatter the domain set $\{0, 1\}^d$. Since the tree has only up to 2^d leaves, it should be intuitive to see that no set $\{0, 1\}^{d+1}$ can be shattered. Therefore, the VC-dimension is 2^d.

5. Exercise **18.2**

   a. First, we calculate the information gain of each of the input features. Denote $x_i \ \forall i \in [3]$. We then calculate the information gain, $\text{Gain}(i)$, as follows where $C(a)$ is the entropy function.

$$\begin{aligned}
\text{Gain}(1) &= C(0.5) - (0.75 * C(2/3) + 0.25 * C(0)) = 0.3113 \\
\text{Gain}(2) &= C(0.5) - (0.50 * C(1/2) + 0.50 * C(1/2)) = 0 \\
\text{Gain}(3) &= C(0.5) - (0.50 * C(1/2) + 0.50 * C(1/2)) = 0
\end{aligned}$$

   Since feature x_1 has the highest gain, we use this as the root node. Note that this causes the three inputs ((1,1,1),1), ((1,0,0),1) and ((1,1,0,),0) to all fall into the same side of the decision tree. From here it can be seen that no matter which subsequent feature is selected, it is impossible to perfectly classify every training example. In fact, choosing x_2 or x_3 will both lead to exactly one training example being misclassified which leads to the training error of 1/4.

   b. A decision tree of depth 2 that leads to zero error is possible if x_1 is not chosen as the root node. One such tree would arise from assigning the root node to feature x_2 and the subsequent feature to x_3. Note that swapping x_2 and x_3 also produces a tree with zero error as well.

6. Exercise **19.1**

   To prove Lemma 19.6, we simply follow the provided hints. First, through the linearity of expectation, we prove the first step.

$$\begin{aligned}\mathop{\mathbb{E}}_{S}\left[\sum_{i:|C_i\cap S|<k}\mathbb{P}[C_i]\right] &= \sum_{i=1}^{r}\mathbb{1}_S[|C_i\cap S|<k]\,\mathbb{P}[C_i]\\ &= \sum_{i=1}^{r}\mathop{\mathbb{P}}_{S}[|C_i\cap S|<k]\,\mathbb{P}[C_i]\end{aligned}$$

   Next, if we suppose that $k < \mathbb{P}[C_i]m/2$, then using Chernoff's bound, we can show the following inequality.

$$\mathop{\mathbb{P}}_{S}[|C_i\cap S] < k] \le \mathop{\mathbb{P}}_{S}[|C_i\cap S] < \mathbb{P}[C_i]m/2 \le e^{-\mathbb{P}[C_i]m/8}$$

   We can use the inequality $\max_a ae^{-ma} \le 1/(me)$ to show the following by multiplying both ends of the previous inequality by $\mathbb{P}[C_i]$. Then we get that $a = \mathbb{P}[C_i]$ and $m = m/8$ as shown.

$$\mathbb{P}[C_i]\mathop{\mathbb{P}}_{S}[|C_i\cap S] < k] \le \mathbb{P}[C_i]e^{-\mathbb{P}[C_i]m/8} \le \frac{8}{me}$$

   From here, we can conclude the proof by noting that if $k \ge 2$, then $8/(me) \le 2kr/m$ and therefore we can conclude the following lemma.

$$\mathbb{P}[C_i]\mathop{\mathbb{P}}_{S}[|C_i\cap S] < k] \le \frac{2rk}{m}$$

7. Error on the testing set attained by the Perceptron model implemented by scikit-learn is 0.

   Error on the testing set attained in MyPerceptron is also 0. Convergence was attained on the training set after 2 iterations of the algorithm.

8. The mean squared error on the testing set attained by the LinearRegression model implemented in scikit-learn is 27.20.

   The mean squared error attained on the testing set in MyLinearRegression is also 27.20.