Introduction to Machine Learning	**Fall Semester**

Homework 5: December 20, 2020

Due: January 6, 2021

Theory Questions

1. **(10 Points, 5 points for each section) Suboptimality of ID3.** Solve exercise 2 in chapter 18 in the book: Understanding Machine Learning: From Theory to Algorithms.

2. **(15 points, 5 points for each section) AdaBoost.** Let $x_1, \ldots, x_m \in \mathbb{R}^d$ and $y_1, \ldots, y_m \in \{-1, 1\}$ its labels. We run the AdaBoost algorithm as given in the recitation, and we are in iteration t. Assume that $\epsilon_t > 0$.

   (a) **(Do not submit)** Show that $\epsilon_t e^{\alpha_t} = \sqrt{\epsilon_t(1-\epsilon_t)} = (1-\epsilon_t)e^{-\alpha_t}$. Use the latter equalities to show that $\sum_{j=1}^{n} D_t(x_j)e^{-\alpha_t y_j h_t(x_j)} = 2\sqrt{\epsilon_t(1-\epsilon_t)}$.

   (b) Show that the error of the current hypothesis relative to the new hypothesis is exactly 1/2, that is:
   $$\Pr_{x \sim D_{t+1}}[h_t(x) \neq y] = \frac{1}{2}.$$

   (c) Show that AdaBoost will not pick the same hypothesis twice consecutively; that is $h_{t+1} \neq h_t$.

   (d) Show that setting the weights to be $\frac{1}{2}\ln\frac{1-\epsilon_t}{\epsilon_t}$ brings Z_t to a minimum.

3. **(10 points, 5 points for each section) Sufficient Condition for Weak Learnability.** Let $S = \{(x_1, y_1), \ldots, (x_n, y_n)\}$ be a training set and let $\mathcal{H}$ be a hypothesis class. Assume that there exists $\gamma > 0$, hypotheses $h_1, \ldots, h_k \in \mathcal{H}$ and coefficients $a_1, \ldots, a_k \geq 0$, $\sum_{i=1}^{k} a_i = 1$ for which the following holds:
   $$y_i \sum_{j=1}^{k} a_j h_j(x_i) \geq \gamma \tag{1}$$
   for all $(x_i, y_i) \in S$.

   (a) Show that for any distribution D over S there exists $1 \leq j \leq k$ such that
   $$\Pr_{i \sim D}[h_j(x_i) \neq y_i] \leq \frac{1}{2} - \frac{\gamma}{2}.$$
   (Hint: Take expectation of both sides of inequality (1) with respect to D.)

   (b) Let $S = \{(x_1, y_1), \ldots, (x_n, y_n)\} \subseteq \mathbb{R}^d \times \{-1, 1\}$ be a training set that is realized by a d-dimensional hyper-rectangle classifier, i.e., there exists a d dimensional hyper-rectangle $[a_1, b_1] \times \cdots \times [a_d, b_d]$ that classifies the data correctly. Let $\mathcal{H}$ be the class of decision stumps of the form
   $$h(x) = \begin{cases} 1 & x_j \leq \theta \\ -1 & x_j > \theta \end{cases}, \quad h(x) = \begin{cases} 1 & x_j \geq \theta \\ -1 & x_j < \theta \end{cases},$$

for $1 \leq j \leq d$ and $\theta \in \mathbb{R} \cup \{\infty, -\infty\}$ (for $\theta \in \{\infty, -\infty\}$ we get constant hypotheses which predict always 1 or always -1). Show that there exist $\gamma > 0$, $k > 0$, hypotheses $h_1, \ldots, h_k \in \mathcal{H}$ and $a_1, \ldots, a_k \geq 0$ with $\sum_{i=1}^{k} a_i = 1$, such that the condition in inequality (1) holds for the training set S and hypothesis class $\mathcal{H}$.

(Hint: Set $k = 4d - 1$ and let $2d - 1$ of the hypotheses be constant.)

4. **(15 points, 7.5 points for each section) Linear regression with dependent variables.** Consider the regression problem where X is a $n \times d$ data matrix, y is a column vector of size n, and w is a column vector of size d of coefficients. As we discussed in the lecture, if there are dependent variables there are infinite possible solutions that achieve this minimum. One sensible criterion to choose one among all possible solutions, is to prefer a solution with a minimal ℓ_2 norm. That is, we search for w that solves the following problem:

$$\arg\min_{w} \|w\|^2$$
$$s.t. Xw = y$$

Assume that $d > n$ and that the matrix X has rank n (note that it in principle the rank can be smaller than n). And denote by $w^\star$ the optimum of the above problem.

(a) (No need to submit) Convince yourself that there exists a w such that $Xw = y$. Namely, the above problem has at least one feasible solution.

(b) Show that the optimal w can be written as a linear combination of the data point. Namely, there exists a vector $\alpha \in \mathbb{R}^n$ such that the solution is given by $w^\star = X^T \alpha$.

(c) Show that you can calculate $x^T w^\star$ for all x by using only dot products between $x \in \mathbb{R}^d$ (hint: express the solution using the kernel matrix $K_S = XX^T$).

Note that the above implies that you can use the "kernel trick" in this case. Namely, you can also work with features $\phi(x)$ as long as you can calculate the corresponding kernel.

5. **(15 points) Perceptron Lower Bound.** Show that for any $0 < \gamma < 1$ there exists a number $d > 0$, vector $w^\star \in \mathbb{R}^d$ and a sequence of examples $(x_1, y_1), \ldots, (x_m, y_m)$ such that:

(a) $\|x_i\| = 1$.

(b) $\frac{y_i x_i \cdot w^\star}{\|w^\star\|} \geq \gamma$.

(c) Perceptron makes $\left\lfloor \frac{1}{\gamma^2} \right\rfloor$ mistakes on the sequence.

(Hint: Choose $m = d = \left\lfloor \frac{1}{\gamma^2} \right\rfloor$ and let $\{x_i\}_i$ be the standard basis of $\mathbb{R}^d$)

6. **(15 points) Halving Algorithm.** Denote by $\mathcal{A}_{Hal}$ the Halving algorithm you have seen in class. Let $d \geq 6$, $\mathcal{X} = \{1, \ldots, d\}$ and let $\mathcal{H} = \{h_{i,j} : 1 \leq i < j \leq d\}$ where

$$h_{i,j}(x) = \begin{cases} 1 & (x = i) \vee (x = j) \\ 0 & otherwise \end{cases}.$$

Show that $M(\mathcal{A}_{Hal}, \mathcal{H}) = 2$.

(Definition of mistake bound $M(\mathcal{A}, \mathcal{H})$: Let $\mathcal{H}$ be a hypothesis class and $\mathcal{A}$ an online algorithm. Given any sequence $S = (x_1, h^\star(x_1)), \ldots, (x_m, h^\star(x_m))$ where m is an integer and $h^\star \in \mathcal{H}$, let $M_{\mathcal{A}}(S)$ be the number of mistakes $\mathcal{A}$ makes on the sequence S. Then $M(\mathcal{A}, \mathcal{H}) = \sup_S M_{\mathcal{A}}(S)$).

Programming Assignment

Submission guidelines:

- Download the supplied files from Moodle (2 python files and 1 `tar.gz` file). Details on every file will be given in the exercises. You need to update the code only in the skeleton files, i.e., the files that have a prefix "skeleton". Written solutions, plots and any other non-code parts should be included in the written solution submission.
- Your code should be written in Python 3.
- Make sure to comment out or remove any code which halts code execution, such as matplotlib popup windows.
- Your code submission should include these files: `adaboost.py`,`process_data.py`

1. **(30 points) AdaBoost.** In this exercise, we will implement AdaBoost and see how boosting can be applied to real-world problems. We will focus on binary sentiment analysis, the task of classifying the polarity of a given text into two classes - positive or negative. We will use movie reviews from IMDB as our data.

   Download the provided files from Moodle and put them in the same directory:

   - `review_polarity.tar.gz` - a sentiment analysis dataset of movie reviews from IMBD.[1] Extract its content in the same directory (with any of zip, 7z, winrar, etc.), so you will have a folder called `review_polarity`.
   - `process_data.py` - code for loading and preprocessing the data.
   - `skeleton_adaboost.py` - this is the file you will work on, change its name to `adaboost.py` before submitting.

   The main function in `adaboost.py` calls the `parse_data` method, that processes the data and represents every review as a 5000 vector x. The values of x are counts of the most common words in the dataset (excluding stopwords like "a" and "and"), in the review that x represents. Concretely, let $w_1, \ldots, w_{5000}$ be the most common words in the data, given a review r_i we represent it as a vector $x_i \in N^{5000}$ where $x_{i,j}$ is the number of times the word w_j appears in r_i. The method `parse_data` returns a training data, test data and a vocabulary. The vocabulary is a dictionary that maps each index in the data to the word it represents (i.e. it maps $j \to w_j$).

   (a) **(10 points)** Implement the AdaBoost algorithm in the `run_adaboost` function. The class of weak learners we will use is the class of hypothesis of the form:

   $$h(x_i) = \begin{cases} 1 & x_{i,j} \leq \theta \\ -1 & x_{i,j} > \theta \end{cases}, \quad h(x_i) = \begin{cases} 1 & x_{i,j} \geq \theta \\ -1 & x_{i,j} < \theta \end{cases},$$

   That is, comparing a single word count to a threshold. At each iteration, AdaBoost will select the best weak learner. Note that the labels are $\{-1, 1\}$. Run AdaBoost for $T = 80$ iterations. Show plots for the training error and the test error of the classifier implied at each iteration t, $sign(\sum_{j=1}^{t} \alpha_j h_j(x))$.

[1] `http://www.cs.cornell.edu/people/pabo/movie-review-data/`

(b) **(10 points)** Run AdaBoost for $T = 10$ iterations. Which weak classifiers the algorithm chose? Pick 3 that you would expect to help to classify reviews and 3 that you did not expect to help, and explain possible reasons for the algorithm to choose them.

(c) **(10 points)** In next recitation you will see that AdaBoost minimizes the average exponential loss:

$$\ell = \frac{1}{m} \sum_{i=1}^{m} e^{-y_i \sum_{j=1}^{T} \alpha_j h_j(x_i)}.$$

Run AdaBoost for $T = 80$ iterations. Show plots of ℓ as a function of T, for the training and the test sets. Explain the behavior of the loss.