Introduction to Machine Learning	**Fall Semester**

Homework 5: Dec 11th, 2018

Due: Dec 25th, 2018

Theory Questions

1. **(18 points, 6 points for each section) VC-dimension of Neural Networks - Upper bound.** We will now finish what we have started in recitation 7 and in the previous assignment. Let $\mathcal{C}$ be the class of hypotheses implementable by neural networks (NN) with L layers (including the output layer, excluding the input layer), each layer has exactly d nodes (except the output layer which has a single node), and the *sign* activation function for all nodes.

   Denote by $\mathcal{H}$ the family of linear separators in R^d, we have seen that the output function of any single node i in the t^{th} layer implements a function which is a member of $\mathcal{H}$. Seen as a whole function, each layer implements a function from $\mathbb{R}^d$ to $\mathbb{R}^d$:

   $$f^{(t+1)}(\mathbf{z}_t) := \mathbf{z}_{t+1} = h(\mathbf{W}^{(t+1)}\mathbf{z}_t + \mathbf{b}^{(t)})$$

   where h operates element-wise. Denote by $\mathcal{F}$ the class of such functions.

   (a) Show that $\Pi_{\mathcal{F}}(n) \leq \left(\frac{en}{d+1}\right)^{d(d+1)}$ for every $n \geq d+1$.

   (b) Express $\mathcal{C}$ in terms of $\mathcal{H}$. Give a bound on the growth function of $\mathcal{C}$, for $n \geq d+1$.

   (c) Let N by the number of parameters in a multilayer NN as defined above. Express N in terms of d and L (number of layers).

   (d) (**Bonus 6 points**) Show that $2^n \leq (en)^N \Rightarrow n \leq 2N \log_2(eN)$.

   (e) We are finally in a position to derive a bound for the VC-dimension. Show that $\pi_{\mathcal{C}}(n) \leq (en)^N$, and use this to show that $VCdim(\mathcal{C}) \leq 2N \log_2(eN)$.

2. **Suboptimality of ID3**. **(16 Points, 8 points for each section)** Solve exercise 2 in chapter 18 in the course book: Understanding Machine Learning: From Theory to Algorithms.

3. **(18 points, 9 points for each section) AdaBoost.** Let $\mathbf{x}_1, \ldots, \mathbf{x}_m \in \mathbb{R}^d$ 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 $\epsilon_t e^{w_t} = \sqrt{\epsilon_t(1-\epsilon_t)} = (1-\epsilon_t)e^{-w_t}$. Use the latter equalities to show that $\sum_{j=1}^{n} D_t(\mathbf{x}_j)e^{-w_t y_j h_t(\mathbf{x}_j)} = 2\sqrt{\epsilon_t(1-\epsilon_t)}$.

   (b) Show that the error of the current hypothesis relative to the new hypothesis is exactly 1/2, that is:

   $$\Pr_{\mathbf{x} \sim D_{t+1}}[h_t(\mathbf{x}) \neq y] = \frac{1}{2}$$

   (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.

4. **(18 points, 9 points for each section) Sufficient Condition for Weak Learnability**. Let $S = \{(x_1, y_1), ..., (x_n, y_n\}$ be a training set and let $\mathcal{H}$ be a hypothesis class. Assume that there exists $\gamma > 0$, hypotheses $h_1, ..., h_k \in \mathcal{H}$ and coefficients $a_1, ..., a_k \geq 0$, $\sum_{i=1}^{k} a_i = 1$ 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 $\mathcal{D}$ over S there exists $1 \leq j \leq k$ such that

$$P_{i\sim\mathcal{D}}(h_j(x_i) \neq y_i) \leq \frac{1}{2} - \frac{\gamma}{2}$$

   .
   (**Hint**: Take expectation of both sides of inequality (1) with respect to $\mathcal{D}$.)

   (b) Let $S = \{(\mathbf{x}_1, y_1), ..., (\mathbf{x}_n, y_n)\} \subseteq \mathbb{R}^d \times \{-1, 1\}$ be a training set that is realized by a d-dimensional hyper-rectangle classifier, i.e., there exists a d dimensional hyper-rectangle $[a_1, b_1] \times \cdots \times [a_d, b_d]$, such that $y_i = 1$ if and only if $\mathbf{x}_i \in [a_1, b_1] \times \cdots \times [a_d, b_d]$. Let $\mathcal{H}$ be the class of decision stumps of the form

$$h(\mathbf{x}) = \begin{cases} 1 & x_j \leq \theta \\ -1 & x_j > \theta \end{cases}, \quad h(\mathbf{x}) = \begin{cases} 1 & x_j \geq \theta \\ -1 & x_j < \theta \end{cases}$$

   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 $h_1, ..., h_k \in \mathcal{H}$ and $a_1, ..., a_k \geq 0$ with $\sum_{i=1}^{m} a_i = 1$, 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.)

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 $\mathbf{x}$. The values of $\mathbf{x}$ are counts of the most common words in the dataset (excluding stopwords like *"a"* and *"and"*), in the review that $\mathbf{x}$ represents. Concretely, let $w_1, w_2, ..., w_{5000}$ be the most common words in the data, given a review r_i we represent it as a vector $\mathbf{x}_i \in \mathbf{N}^{5000}$ 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:

   $$h(\boldsymbol{x}_i) = \begin{cases} 1 & x_{i,j} \le \theta \\ -1 & x_{i,j} > \theta \end{cases}, \quad h(\boldsymbol{x}_i) = \begin{cases} -1 & x_{i,j} \le \theta \\ 1 & x_{i,j} > \theta \end{cases}$$

   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, $sign(\sum_{j=1}^{t} \alpha_j h_j(\mathbf{x}))$.

[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:

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

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.