Introduction to Machine Learning **Fall Semester**

Homework 3: Nov 11th, 2018

Due: Nov 25th (See the submission guidelines in the course web site)

Theory Questions

1. **(10 points) Perceptron Lower Bound.** Show that for any $0 < \gamma < 1$ there exists a number $d > 0$, vector $\mathbf{w}^* \in \mathbb{R}^d$ for which $\|\mathbf{w}^*\| \leq 1$ and a sequence of examples $(\mathbf{x}_1, y_1), ..., (\mathbf{x}_m, y_m)$ such that:

   (a) $\|\mathbf{x}_i\| = 1$.

   (b) $\frac{y_i \mathbf{x}_i \cdot \mathbf{w}^*}{\|\mathbf{w}^*\|} \geq \gamma$.

   (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 $\{\mathbf{x}_i\}_i$ be the standard basis of $\mathbb{R}^d$)

2. **(15 points) Halving Algorithm**. Denote by $\mathcal{A}_{Halving}$ the Halving algorithm you have seen in class. Let $d \geq 6$, $\mathcal{X} = \{1, 2, ..., d\}$ and let $\mathcal{H} = \{h_{i,j} \mid 1 \leq i < j \leq d\}$ where

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

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

   (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 $S = (x_1, h^*(x_1)), ..., (x_m, h^*(x_m))$ where m is an integer and $h^* \in \mathcal{H}$, let $M_{\mathcal{A}}(S)$ be the number of mistakes $\mathcal{A}$ makes on the sequence S. Then $M(\mathcal{A}, \mathcal{H}) = \sup_S M_{\mathcal{A}}(S)$.)

3. **(10 points) Max of Convex Functions**. Consider m convex functions $f_1(\boldsymbol{x}), \ldots, f_m(\boldsymbol{x})$, where $f_i : \mathbb{R}^d \to \mathbb{R}$. Now define a new function $g(\boldsymbol{x}) = \max_i f_i(\boldsymbol{x})$.

   (a) Prove that $g(\boldsymbol{x})$ is a convex function.

   (b) Show that a sub-gradient of g at point $\boldsymbol{x}$ is the gradient of a function f_i for which $f_i(\boldsymbol{x}) = \max\{f_1(\boldsymbol{x}), ..., f_m(\boldsymbol{x})\}$.

4. **(15 points) ℓ^2 penalty.** Consider the following problem:

$$\begin{aligned} &\min_{\mathbf{w}, b, \boldsymbol{\xi}} \frac{1}{2}\mathbf{w}^T\mathbf{w} + \frac{C}{2}\sum_{i=1}^{m} \xi_i^2 \\ &\text{s.t. } y_i(\mathbf{w}^T\mathbf{x}_i + b) \geq 1 - \xi_i \qquad \forall i = 1, \ldots, m \end{aligned}$$

(a) Show that a constraint of the form $\xi_i \geq 0$ will not change the problem. meaning, Show that these non-negativity constraints can be removed. That is, show that the optimal value of the objective will be the same whether or not these constraints are present.

(b) What is the Lagrangian of this problem?

(c) Minimize the Lagrangian with respect to $\mathbf{w}, b, \boldsymbol{\xi}$ by setting the derivative with respect to these variables to 0.

(d) What is the dual problem?

5. **(20 points) Interval growth function.** The goal of this exercise is to compute the growth function of the interval hypothesis class $H = \{h_{a,b} : a < b\}$ and $\chi = \Re$. Where $h(x) = 1$ if $x \in [a, b]$ else 0. In other words, your goal is to give an explicit expression to $\Pi_H(m) = max_{C:|C|=m}|H_C|$ where H_C is the restriction of H on the set C.

6. **(15 points - bonus) Sample complexity of agnostic PAC.** Let H be a hypothesis class of functions from a domain χ to $\{0, 1\}$ and let the loss function be the $0 - 1$ loss. Assume that $VCdim(H) = d < \infty$. Show that if

$$n \geq \frac{256d}{\epsilon^2} ln(\frac{128d}{\epsilon^2}) + \frac{256}{\epsilon^2} \ln(\frac{1}{\delta})$$

then:

$$P[e_p(ERM(S)) - min_{h \in H} e_p(h) > \epsilon] \leq \delta$$

To prove the above claim you can use the following lemma without proving it:
Lemma: Let $a \geq 1$ and $b > 0$. Then: $x \geq 4alog(2a) + 2b \rightarrow x \geq alog(x) + b$
You can also assume that δ is as small as you desire.

Programming Assignment

Submission guidelines

- Download the files `skeleton_sgd.py` and `skeleton_perceptron.py` from Moodle. In each of the following questions you should only implement the algorithm at each of the skeleton files. Plots, tables and any other artifact should be submitted with the theoretical section.
- In the file `skeleton_sgd.py` there is an helper function. The function reads the examples labelled $0, 8$ and returns them with $0-1$ labels. Case you are unable to read the MNIST data with the provided script, you can download the file from here: https://github.com/amplab/datascier sp14/blob/master/lab7/mldata/mnist-original.mat.
- Your code should be written with python 3.
- Make sure to comment out / remove any code which halts the code execution, such as matplitlib popup.
- Your submission should include exactly two files: `sgd.py`, `perceptron.py`.

In the following questions, we will study the performance of various hyperplane algorithms on the MNIST dataset. As before, we will have a held out *test set* that will be used to assess the performance of the selected best hypotheses. However, we will divide the rest of the data to a *training set* and *validation set*. For algorithms in which we have to select external parameters, we will use use the training set to train classifiers for various parameter values and use the validation set to check how well they do.

1. **(15 points) Perceptron.** Implement the Perceptron algorithm (in the file name `perceptron.py`). Do not forget to normalize the samples to have unit length (i.e., $\|\mathbf{x}_i\| = 1$).
   (a) **(6 points)** Use only the first $n = 5, 10, 50, 100, 500, 1000, 5000$ samples as an input to Perceptron. For each n, run Perceptron 100 times, each time with a different random order of inputs, and calculate the accuracy of the classifier on the *test set* of each run. You should therefore have 100 accuracy measurement per n. Print a table showing, for each n, the mean accuracy across the 100 runs, as well as the 5% and 95% percentiles of the accuracies obtained.
   (b) **(3 points)** The weight vector $\mathbf{w}$, returned by Percepton, can be viewed as a matrix of weights, with which we multiply each respective pixel in the input image. Run Perceptron on the entire *training set*, and show $\mathbf{w}$, as a 28×28 image, for example with `imshow(reshape(image, (28, 28)), interpolation='nearest')`. Give an intuitive interpretation of this image.
   (c) **(3 points)** Calculate the accuracy of the classifier trained on the full *training set*, applied on the *test set*.
   (d) **(3 points)** Choose one (or two) of the samples in the *test set* that was misclassified by Perceptron (with the full training set) and show it as an image (show the unscaled images). Can you explain why it was misclassified?
2. **(15 points) SGD.** We will continue working with the MNIST data set as in the previous section. The file template (`sgd.py`), contains the code to load the training, validation and test

sets for the digits 0 and 8 from the MNIST data. In this exercise we will optimize the Hinge loss (as you seen in the lecture) using the stochastic gradient descent implementation discussed in class. Namely, at each iteration $t = 1, \ldots$ we sample i uniformly; and if $y_i \mathbf{w}_t \cdot \mathbf{x}_i < 1$, we update:

$$\mathbf{w}_{t+1} = (1 - \eta_t)\mathbf{w}_t + \eta_t C y_i \mathbf{x}_i$$

and $\mathbf{w}_{t+1} = (1 - \eta_t)\mathbf{w}_t$ otherwise, where $\eta_t = \eta_0/t$, and η_0 is a constant. Implement an SGD function that accepts the samples and their labels, C, η_0 and T, and runs T gradient updates as specified above. In the questions that follow, make sure your graphs are meaningful. Consider using *set_xlim* or *set_ylim* to concentrate only on a relevant range of values.

(a) **(6 points)**Train the classifier on the *training set*. Use cross-validation on the *validation set* to find the best η_0, assuming $T = 1000$ and $C = 1$. For each possible η_0 (for example, you can search on the log scale $\eta_0 = 10^{-5}, 10^{-4}, ..., 10^4, 10^5$ and increase resolution if needed), assess the performance of η_0 by averaging the accuracy on the *validation set* across 10 runs. Plot the average accuracy on the *validation set*, as a function of η_0.

(b) **(3 points)**Now, cross-validate on the *validation set* to find the best C given the best η_0 you found above. For each possible C (again, you can search on the log scale is an in section (a)), average the accuracy on the *validation set* across 10 runs. Plot the average accuracy on the *validation set*, as a function of C.

(c) **(3 points)** Using the best C, η_0 you found, train the classifier, but for $T = 20000$. Show the resulting $\mathbf{w}$ as an image.

(d) **(3 points)** What is the accuracy of the best classifier on the *test set*?