**Introduction to Machine Learning** **Spring Semester**

Homework 3: April 24, 2023

*Due: May 9, 2023*

# Theory Questions

**Remark:** Throughout this exercise, when we write a norm $\|\cdot\|$ we refer to the $\ell_2$-norm.

1. **(15 points) Step-size Perceptron.** Consider the modification of Perceptron algorithm with the following update rule:

   $$\mathbf{w}_{t+1} \leftarrow \mathbf{w}_t + \eta_t y_t \mathbf{x}_t$$

   whenever $\hat{y}_t \neq y_t$ ($\mathbf{w}_{t+1} \leftarrow \mathbf{w}_t$ otherwise). Assume that data is separable with margin $\gamma > 0$ and that $\|\mathbf{x}_t\| = 1$ for all $t$. For simplicity assume that the algorithm makes $M$ mistakes at the first $M$ rounds, afterwhich it has no mistakes. For $\eta_t = \frac{1}{\sqrt{t}}$, show that the number of mistakes step-size Perceptron makes is at most $\frac{4}{\gamma^2}\log(\frac{1}{\gamma})$. (Hint: use the fact that if $x \leq a\log(x)$ then $x \leq 2a\log(a)$). It's okay if you obtain a bound with slightly different constants, but the asymptotic dependence on $\gamma$ should be tight.

2. **(15 points) Convex functions.**

   (a) Let $f : \mathbb{R}^n \to \mathbb{R}$ a convex function, $A \in \mathbb{R}^{n\times n}$ and $b \in \mathbb{R}^n$. Show that, $g(\mathbf{x}) = f(A\mathbf{x} + b)$ is convex.

   (b) Consider $m$ convex functions $f_1(\mathbf{x}), \ldots, f_m(\mathbf{x})$, where $f_i : \mathbb{R}^d \to \mathbb{R}$. Now define a new function $g(\mathbf{x}) = \max_i f_i(\mathbf{x})$. Prove that $g(\mathbf{x})$ is a convex function. (Note that from (a) and (b) you can conclude that the hinge loss over linear classifiers is convex.)

   (c) Let $\ell_{log} : \mathbb{R} \to \mathbb{R}$ be the log loss, defined by

   $$\ell_{\log}(z) = \log_2\left(1 + e^{-z}\right)$$

   Show that $\ell_{log}$ is convex, and conclude that the function $f : \mathbb{R}^d \to \mathbb{R}$ defined by $f(\mathbf{w}) = \ell_{log}(y\mathbf{w}\cdot\mathbf{x})$ is convex with respect to $\mathbf{w}$.

3. **(20 points) Ranking**. In this question, we consider a new learning task in which the objective is to rank items. Assume items are elements of $\mathcal{X} \subseteq \mathbb{R}^d$, and you are given a training set of $n$ lists of $k$ items each, and for each list you receive a "label" vector corresponding to the correct ranking of its items. More formally, you receive a training set

   $$S = \left\{\left((\mathbf{x}_1^i, \mathbf{x}_2^i, \ldots, \mathbf{x}_k^i), \mathbf{y}^i\right)\right\}_{i=1}^n$$

such that for all $1 \leq i \leq n$, $\mathbf{y}^i \in \mathbb{R}^k$ assigns a value for each item in $\bar{\mathbf{x}}^i = (\mathbf{x}^i_1, \mathbf{x}^i_2, \ldots, \mathbf{x}^i_k)$, interpreted as a ranking of the items. Your goal is to learn a ranking function $h : \mathcal{X}^k \to \mathbb{R}^k$ which correctly ranks the lists of items from $S$. The *Kendall-Tau* loss between two rankings $\mathbf{y}', \mathbf{y}$ is defined as follows:

$$\Delta(\mathbf{y}', \mathbf{y}) = \frac{2}{k(k-1)} \sum_{j=1}^{k-1} \sum_{r=j+1}^{k} \mathbf{1}\left\{sgn(y'_j - y'_r) \neq sgn(y_j - y_r)\right\}$$

Note that this function averages the total number of pairs of items which are in different order in $\mathbf{y}'$ compared to $\mathbf{y}$. Assume you are trying to learn a linear ranking function, i.e. a function of the form

$$h_{\mathbf{w}}((\mathbf{x}_1, \ldots, \mathbf{x}_k)) = (\mathbf{w} \cdot \mathbf{x}_1, \ldots, \mathbf{w} \cdot \mathbf{x}_k)$$

for some $\mathbf{w} \in \mathbb{R}^d$, and your goal is to minimize the Kendall-Tau loss over $S$: $\sum_{i=1}^n \Delta(h_{\mathbf{w}}(\bar{\mathbf{x}}^i), \mathbf{y}^i)$. Since this function is hard to optimize, you instead optimize the surrogate "hinge" loss $\sum_{i=1}^n \ell(h_{\mathbf{w}}(\bar{\mathbf{x}}^i), \mathbf{y}^i)$ where:

$$\ell(h_{\mathbf{w}}(\bar{\mathbf{x}}), \mathbf{y}) = \frac{2}{k(k-1)} \sum_{j=1}^{k-1} \sum_{r=j+1}^{k} \max\{0, 1 - sgn(y_j - y_r)\mathbf{w} \cdot (\mathbf{x}_j - \mathbf{x}_r)\}$$

(a) Prove that the hinge loss described above for the ranking objective is convex in $\mathbf{w}$.

(b) Prove that the hinge loss upper-bounds the Kendall-Tau loss, i.e. that $\Delta(h_{\mathbf{w}}(\bar{\mathbf{x}}), \mathbf{y}) \leq \ell(h_{\mathbf{w}}(\bar{\mathbf{x}}), \mathbf{y})$ for all $\mathbf{w} \in \mathbb{R}^d, \bar{\mathbf{x}} \in \mathcal{X}^k, \mathbf{y} \in \mathbb{R}^k$.

(c) Prove that if the data is separable with a margin $\gamma > 0$ (i.e. when there exists $\mathbf{w}^\star \in \mathbb{R}^d$ and $\gamma > 0$ such that $sgn(y^i_j - y^i_r)\mathbf{w}^\star \cdot (\mathbf{x}^i_j - \mathbf{x}^i_r) \geq \gamma$ for all $1 \leq i \leq n$ and all $1 \leq j < r \leq k$), minimizing the hinge loss will result in a ranking function which minimizes the Kendall-Tau loss.

4. **(15 points) Gradient Descent on Smooth Functions.** We say that a continuously differentiable function $f : \mathbb{R}^n \to \mathbb{R}$ is $\beta$-smooth if for all $\mathbf{x}, \mathbf{y} \in \mathbb{R}^n$

$$f(\mathbf{y}) \leq f(\mathbf{x}) + \nabla f(\mathbf{x})^T(\mathbf{y} - \mathbf{x}) + \frac{\beta}{2}\|\mathbf{x} - \mathbf{y}\|^2$$

In words, $\beta$-smoothness of a function $f$ means that at every point $\mathbf{x}$, $f$ is upper bounded by a qaudratic function which coincides with $f$ at $\mathbf{x}$.

Let $f : \mathbb{R}^n \to \mathbb{R}$ be a $\beta$-smooth and non-negative function (i.e., $f(\mathbf{x}) \geq 0$ for all $\mathbf{x} \in \mathbb{R}^n$). Consider the (non-stochastic) gradient descent algorithm applied on $f$ with constant step size $\eta > 0$:

$$\mathbf{x}_{t+1} = \mathbf{x}_t - \eta \nabla f(\mathbf{x}_t)$$

Assume that gradient descent is initialized at some point $\mathbf{x}_0$. Show that if $\eta < \frac{2}{\beta}$ then

$$\lim_{t \to \infty} \|\nabla f(\mathbf{x}_t)\| = 0$$

(Hint: Use the smoothness definition with points $\mathbf{x}_{t+1}$ and $\mathbf{x}_t$ to show that $\sum_{t=0}^{\infty} \|\nabla f(\mathbf{x}_t)\|^2 < \infty$ and recall that for a sequence $a_n \geq 0$, $\sum_{n=1}^{\infty} a_n < \infty$ implies $\lim_{n\to\infty} a_n = 0$. Note that $f$ is not assumed to be convex!)

# Programming Assignment

Submission guidelines:

- Download the file `skeleton_sgd.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 the labels $-1/+1$. In case you are unable to read the MNIST data with the provided script, you can download the file from here:

  `https://github.com/amplab/datascience-sp14/blob/master/lab7/mldata/mnist-original.mat`
- Your code should be written in Python 3.
- Your code submission should include one file: `sgd.py`.

1. **(20 points) SGD for Hinge loss.** We will continue working with the MNIST data set. The file template (`skeleton_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 with $L2$-regularization ($\ell(\mathbf{w}, \mathbf{x}, y) = C(\max\{0, 1 - y\langle \mathbf{w}, \mathbf{x}\rangle\}) + 0.5\|\mathbf{w}\|^2$), using the stochastic gradient descent implementation discussed in class. Namely, we initialize $\mathbf{w}_1 = 0$, and at each iteration $t = 1, \ldots$ we sample $i$ uniformly; and if $y_i \mathbf{w}_t \cdot x_i < 1$, 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 $\eta_0$ is a constant. Implement an SGD function that accepts the samples and their labels, $C$, $\eta_0$ 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) **(5 points)** Train the classifier on the training set. Use cross-validation on the validation set to find the best $\eta_0$, assuming $T = 1000$ and $C = 1$. For each possible $\eta_0$ (for example, you can search on the log scale $\eta_0 = 10^{-5}, 10^{-4}, \ldots, 10^4, 10^5$ and increase resolution if needed), assess the performance of $\eta_0$ by averaging the accuracy on the validation set across 10 runs. Plot the average accuracy on the validation set, as a function of $\eta_0$.

   (b) **(5 points)** Now, cross-validate on the validation set to find the best $C$ given the best $\eta_0$ you found above. For each possible $C$ (again, you can search on the log scale as 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) **(5 points)** Using the best $C$, $\eta_0$ you found, train the classifier, but for $T = 20000$. Show the resulting $\mathbf{w}$ as an image, e.g. using the following `matplotlib.pyplot` function: `imshow(reshape(image, (28, 28)), interpolation='nearest')`. Give an intuitive interpretation of the image you obtain.

(d) **(5 points)** What is the accuracy of the best classifier on the test set?

2. **(15 points) SGD for log-loss.** In this exercise we will optimize the log loss defined as follows:

$$\ell_{log}(\mathbf{w}, \mathbf{x}, y) = \log\left(1 + e^{-y\mathbf{w}\cdot\mathbf{x}}\right)$$

(in the lecture you defined the loss with $\log_2(\cdot)$, but for optimization purposes the logarithm base doesn't matter). Derive the gradient update for this case, and implement the appropriate SGD function.

- In your computations, it is recommended to use `scipy.special.softmax` to avoid numerical issues which arise from exponentiating very large numbers.

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

(b) **(5 points)** Using the best $\eta_0$ you found, train the classifier, but for $T = 20000$. Show the resulting $\mathbf{w}$ as an image. What is the accuracy of the best classifier on the test set?

(c) **(5 points)** Train the classifier for $T = 20000$ iterations, and plot the norm of $\mathbf{w}$ as a function of the iteration. How does the norm change as SGD progresses? Explain the phenomenon you observe.