Introduction to Machine Learning	**Fall Semester**

Homework 3: November 10, 2021

Due: November 24, 2021

Theory Questions

1. **(10 points) Composition of convex and affine functions.** Let $f : \mathbb{R}^n \to \mathbb{R}$ a convex function. Given $A \in \mathbb{R}^{n\times n}$ and $b \in \mathbb{R}^n$. Show that, $g(x) = f(Ax + b)$ is convex.

2. **(10 points) Max of Convex Functions.** Consider m convex functions $f_1(x), \ldots, f_m(x)$, where $f_i : \mathbb{R}^d \to \mathbb{R}$. Now define a new function $g(x) = \max_i f_i(x)$.

   (a) Prove that $g(x)$ is a convex function.

   (b) Assume f_i are differentiable. Show that for any x, if $f_i(x) = \max\{f_1(x), \ldots, f_m(x)\}$ then $\nabla f_i(x)$ is a sub-gradient of g at x.

3. **(15 points) The Multi-class Hinge-loss.** Consider the problem of multi-class prediction where the label Y has L values (e.g., $L = 10$ for MNIST). Denote $[L] = 1, 2, \ldots, L$. Assume the inputs are $x \in \mathbb{R}^d$. We will consider classifiers of the form $f(x; w_1, \ldots, w_L) = \arg\max_{y\in[L]} w_y \cdot x$ defined by L vectors $w_1, \ldots, w_L$. Given an input x and its correct label y the error of the classifier is $\Delta_{zo}(f(x; w_1, \ldots, w_L), y)$ where

$$\Delta_{zo}(\hat{y}, y) = \begin{cases} 0 & \hat{y} = y \\ 1 & \hat{y} \neq y \end{cases}$$

   As mentioned in class, this loss is hard to optimize. In the recitation we've seen the cross-entropy loss which is convex. In this question we consider another loss, called the *multi-class hinge loss* defined as follows:

$$\ell(w_1, \ldots, w_L, x, y) = \max_{\hat{y}\in[L]} w_{\hat{y}} \cdot x - w_y \cdot x + \Delta_{zo}(\hat{y}, y).$$

   Given a labeled training set $x_1, \ldots, x_n \in \mathbb{R}^d$ and $y_1, \ldots, y_n \in [L]$ the hinge-loss optimization problem would be:

$$\min_{w_1,\ldots,w_L} \sum_i \ell(w_1, \ldots, w_L, x_i, y_i).$$

   Denote a minimizer of this problem by $w_1^{opt}, \ldots, w_L^{opt}$ (note there may be multiple such minimizers).

   (a) Show that ℓ is a convex function of $w_1, \ldots, w_L$.

   (b) Show that $\ell(w_1, \ldots, w_L, x, y) \geq \Delta_{zo}(f(x; w_1, \ldots, w_L), y)$ for all values of w, x, y.

(c) Assume that for your training set there exists $w_1^\star, \ldots, w_L^\star$ that achieve zero training error (namely $\Delta_{zo}(f(x_i; w_1^\star, \ldots, w_L^\star), y_i) = 0$ for all i). Prove that w^{opt} would also have zero training error. Namely that $\Delta_{zo}(f(x_i; w_1^{opt}, \ldots, w_L^{opt}), y_i) = 0$ for all i.

4. **(15 points) Gradient Descent on quadratic function.** Let A be a positive definite matrix (PSD), and let $f(\mathbf{w}) = \mathbf{w}^T A \mathbf{w}$. Show that there exist a step size η such that $T \leq O(\log(1/\epsilon))$ iterations of gradient descent (with some arbitrary initial point $\mathbf{w}_1 \neq 0$) are sufficient to achieve ϵ-optimal solution. That is,

$$f(\mathbf{w}_T) \leq \epsilon.$$

Note that here we use the last-integrate of GD $\mathbf{w}_T$ instead of the average-iterate $\bar{\mathbf{w}}$. The O-notation may hide constants that depend on $\|\mathbf{w}_1\|$ and on the eigenvalues of A. (Hint: show that one can choose η so that the update of GD is of the form $\mathbf{w}_{t+1} = B\mathbf{w}_t$, where B is a PSD with eigenvalues values strictly smaller than 1).

Remark: f belongs to a family called "strongly convex" and "smooth" functions. It can be shown that GD has a similar convergence rate for any strongly convex and smooth function, and this rate is much better than the $O(1/\epsilon^2)$ rate we proved in class.

5. **(15 points) Gradient Descent on Smooth Function.** We say that a continuously differentiable function $f : \mathbb{R}^n \to \mathbb{R}$ is β-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, β-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 $\ell : \mathbb{R}^n \to \mathbb{R}$ be a β-smooth and non-negative function (i.e., $\ell(\mathbf{x}) \geq 0$ for all $\mathbf{x} \in \mathbb{R}^n$). Consider the (non-stochastic) gradient descent algorithm applied on ℓ with constant step size $\eta > 0$:

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

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

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

(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 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/datasciencesp14/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 ($\ell(w, x, y) = C(max\{0, 1 - y\langle w, x\rangle\}) + 0.5\|w\|^2$), using the stochastic gradient descent implementation discussed in class. Namely, at each iteration $t = 1, \ldots$ we sample i uniformly; and if $y_i w_t \cdot x_i < 1$, we update:

$$w_{t+1} = (1 - \eta_t)w_t + \eta_t C y_i x_i$$

   and $w_{t+1} = (1 - \eta_t)w_t$ 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) **(5 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 $\eta_0 = 10^{-5}, 10^{-4}, \ldots, 10^4, 10^5$ 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) **(5 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 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, η_0 you found, train the classifier, but for $T = 20000$. Show the resulting w as an image.

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

2. **(15 points) SGD for multi-class cross-entropy.** The skeleton file contains a second helper function to load the training, validation and test sets for all the digits. In this exercise we will optimize the multi-class cross entropy loss using SGD. Recall the multi-class cross-entropy loss discussed in the recitation (our classes are $0, 1, \ldots, 9$):

$$\ell_{CE}(w_0, \ldots, w_9, x, y) = \log\left(\sum_{i=0}^{9} e^{w_i \cdot x}\right) - w_y \cdot x$$

Derive the gradient update for this case, and implement the appropriate SGD function.

(a) **(5 points)** Train the classifier on the training set. Use cross-validation on the validation set to find the best η_0, assuming $T = 1000$. For each possible η_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 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) **(5 points)** Using the best η_0 you found, train the classifier, but for $T = 20000$. Show the resulting $w_0, \ldots, w_9$ as images.

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