Introduction to Machine Learning **Fall Semester**

Homework 3: February 14, 2024

Due: March 6, 2024

Theory Questions

Remark: Throughout this exercise, when we write a norm $\|\cdot\|$ we refer to the ℓ_2-norm.

1. **(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 ℓ_{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 $\mathbf{w}$.

2. **(10 points) Hinge loss with linearly separable data.** Consider a setup of learning linear classifiers over a dataset $\{(\mathbf{x}_i, y_i)\}_{i=1}^n$ where $\mathbf{x}_i \in \mathbb{R}^d, y_i \in \{-1, +1\}$ for all i. Assume the data is linearly separable, i.e., given a training set there exists $\mathbf{w}^* \in \mathbb{R}^d$ such that $y_i\mathbf{w}^*\cdot\mathbf{x}_i > 0$ for all i (note the strict inequality; assume no bias for simplicity). Recall the definition of the *hinge loss* given in lecture #6:

   $$\ell_{hinge}(r) = \max\{0, 1 - r\}.$$

   We would like to show that in the linearly separable case, minimizing the hinge loss over the training data will yield a classifier with optimal zero-one loss. Formally, let

   $$\mathbf{w}^*_{hinge} = \text{argmin}_{\mathbf{w}\in\mathbb{R}^d}\left\{\sum_{i=1}^n \ell_{hinge}(y_i\mathbf{w}\cdot\mathbf{x}_i)\right\}.$$

   Show that $sign(\mathbf{w}^*_{hinge}\cdot\mathbf{x}_i) = y_i$ for all i, meaning $\mathbf{w}^*_{hinge}$ achieves optimal zero-one loss.
   (**Hint:** First show that the hinge loss upper bounds the zero-one loss. Then consider the hinge loss of $c\mathbf{w}^*$ for some constant $c > 0$. What happens to the hinge loss as $c \to \infty$?)

3. **(15 points) Gradient Descent on Smooth Functions.** 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 $f : \mathbb{R}^n \to \mathbb{R}$ be a β-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!)

4. **(10 points) Solving hard SVM.** Consider two distinct points $\mathbf{x}_1, \mathbf{x}_2 \in \mathbb{R}^d$ with labels $y_1 = 1$ and $y_2 = -1$. Compute the hyperplane that Hard SVM will return on this data, i.e., give explicit expressions for $\boldsymbol{w}$ and b as functions of $\mathbf{x}_1$, $\mathbf{x}_2$.
(Hint: Solve the dual problem by transforming it to an optimization problem in a single variable. Use your solution to the dual to obtain the primal solution).

5. **(15 points) ℓ^2 penalty.** Consider the following problem:

$$\min_{\boldsymbol{w},b,\boldsymbol{\xi}} \frac{1}{2}\|\boldsymbol{w}\|^2 + \frac{C}{2}\sum_{i=1}^{n} \xi_i^2$$
$$\text{s.t. } y_i(\boldsymbol{w}^T\mathbf{x}_i + b) \geq 1 - \xi_i \quad \forall i = 1, \ldots, n$$

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

   (d) What is the dual problem?

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 ($\ell(\mathbf{w}, \mathbf{x}, y) = C \cdot \max\{0, 1 - y\langle \mathbf{w}, \mathbf{x}\rangle\} + 0.5\|\mathbf{w}\|^2$), 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 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) **(10 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 $\mathbf{w}$ 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 various built in functions (`scipy.special.softmax` might be helpful) in order to avoid numerical issues which arise from exponentiating very large numbers.

(a) **(5 points)** Train the classifier on the training set using SGD. 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 $\mathbf{w}$ 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.