Introduction to Machine Learning (67577)

Exercise 5
Feature selection, Convex analysis and SGD

June 2018

Contents

Submission guidelines

- submission is due 20.6.2018
- The assignment is to be submitted via Moodle only.
- Files should be zipped to **ex_5_First_Last.zip** (First is your first name, Last is your last name. In English, of course).
- The .zip file size should not exceed 10Mb.
- Each question is enumerated, use these numbers when answering your questions.

Theoretical Part

Each theoretical question worths 10 points, except for 1b and 2a that worth 5 points each.

1. Feature selection-
   (a) Optional: Prove exercise 1 from recitation 9
   (b) Prove exercise 2 from recitation 9
2. Convex functions -
   (a) Prove or give a counterexample for the following claim: Given two functions $f, g : \mathbb{R} \to \mathbb{R}$, define a new function $h : \mathbb{R} \to \mathbb{R}$ by $h = f \circ g$. If f and g are convex than h is convex as well.
   (b) A function $f : C \to \mathbb{R}$ defined over a convex set C is convex iff its *epigraph* is a convex set, where epi(f) $= \{(u, t) : f(u) \leqslant t\}$.
   (c) Given $x \in \mathbb{R}^d$ and $y \in \{\pm 1\}$. Show that the hinge loss is convex in w. Meaning, define $f(w) = l^{hinge}(w, x, y)$ and show that f is convex in w.
   (d) Given $f_1, \ldots, f_m : \mathbb{R}^d \to \mathbb{R}$ convex function. Define $f : \mathbb{R}^d \to \mathbb{R}$ by $f(x) = \max_{i \in [m]} f_i(x)$. Define $I_x = \{i \in [m] : f_i(x) = f(x)\}$, for any $x \in \mathbb{R}^d$, show that $\cup_{i \in I_x} \partial f_i(x) \subseteq \partial f(x)$. Hint: try and find the relation between $f(x)$ and $f_i(x_1)$, where $i \in I_{x_1}$.
   (e) Optional: Given $f_1, \ldots, f_m : \mathbb{R}^d \to \mathbb{R}$ convex function, and $\gamma_1, \ldots, \gamma_m \in \mathbb{R}_+$, meaning that they are strictly positive. Define $f : \mathbb{R}^d \to \mathbb{R}$ by $f(x) = \sum_{i=1}^m \gamma_i f_i(x)$. Show that $\sum_{i=1}^m \gamma_i \partial f_i(x) \subseteq \partial f(x)$.
   * The definition of sum of sets, and scalar multiplication of a set is written in recitation 10.
   (f) Given dataset $\{(x_i, y_i)\}_{i=1}^m \subset \mathbb{R}^d \times \{\pm 1\}$. Define the function $f : \mathbb{R}^d \to \mathbb{R}$ that is defined by

$$f(w) = \frac{1}{m} \sum_{i=1}^{m} l^{hinge}(w, x_i, y_i)$$

   Find a member of the sub-gradient of f for each w?

Practical Part

In this part we will implement the GD and SGD algorithms on a handwritten digits dataset (the "Mnist" dataset includes the digits 0-9).

- Download the Mnist handwritten training dataset - "train-images-idx3-ubyte.gz" and "train-labels-idx1-ubyte.gz" from: http://yann.lecun.com/exdb/mnist/ . The dataset contains a 28 by 28 images of handwritten digits between 0-9. Our task here is classification.

- Download ex5files.zip from the course website and extract its contents.
- Your results should contain your code, and all the figures you are asked to plot. Additionally, please include a script ex5_runme.py that runs your code, generates the figures, and prints the parameters and errors you are asked to report.

Assumptions:

1. Each image will be described as a vector. While there are is a lot important information in the 2d form of the image, we will see that in this specific case we can do well in the 1d form as well.
2. When we write GD, we actually mean sub gradient descent.
3. The GD and SGD algorithms will use the loss function defined in (f) as the function f. While in SGD we define m as the batch size, in GD it will defined as the training set size (Meaning, the batch is the whole training set).
4. In the recitation we have talked about a case of SGD where at each iteration the algorithm draws 1 sample. In this exercise we are going to talk about an extended case of SGD, where at each iteration the algorithm draws a batch of samples from the training set without repetitions (repetitions could occur between different batchs). In order to implement it examine the function- np.linalg.permutation.
5. We would like to test our hypothesis using the 0-1 loss, but in the training process we are going to use the hinge loss as a surrogate loss (**Train on hinge loss, test on 0-1 loss**).

Coding assignments:

1. Create a file: "GradientDescentMethods.py" that will contain the functions: GD, SGD, and testError (20 points)
   (a) The declaration of the GD function should be: "*GD(data, label, iters,eta)*", where
   - *data*- a $n \times d$ matrix, where n is the amount of samples, and d is the dimension of each sample
   - *labels*- a $n \times 1$ matrix with the labels of each sample
   - *iters* - an integer that will define the amount of iterations.
   - *eta* - a positive number that will define the learning rate.

   The function will output a $d \times iters$ matrix, where in its i-th column it will contain the output of the sub-gradient descent algorithm over i iterations.

(b) The declaration of the SGD function should be: "*SGD(data, label, iters, eta, batch)*", where in addition to the definitions from the previous bullet, we add:

- *batch*- The amount of samples that the algorithm would draw and use at each iteration.

The output of the function is declared the same as in GD.

(c) The declaration of the *testError* function should be: "*testError(w, testData, testLabels)*", where

- w- a $d \times l$ matrix, where l is the amount of the linear hypothesis that you would like to examine, each of which are defined using a d- dimensional vector ($h_i(x) = \langle w_i, x \rangle$, where $w_i \in \mathbb{R}^d$).
- *testData* - a $n \times d$ matrix with the samples
- *testLabels*- a $d \times 1$ matrix with the labels of the samples.

The function will output a $l \times 1$ matrix with the respective 0-1 loss for each hypothesis (Make sure to take the sign of the inner product as the predicted label).

2. Define the training data and test data using the first 1000 samples (90%for training, 10% for test) of the digits **0,1** by using the function *genMnistDigits* in *readMnist*. Make sure to use the data that you get from *readMnist*, when calling it. A simple code is given:
data , labels = readMnist()
currData, currLabels = genMnistDigits([0,1],500,data, labels)

3. After writing all this code, lets play with it and try and analyze the results (30 points)

(a) Generate the data-
Make sure to add a 1 at the end of each sample to work with linear hypothesis class instead of affine.
Make sure to change the labels of 0 with -1.

(b) Run the SGD algorithm with batches of 5,50,150 and the GD algorithm over 150 iterations, with eta=1.

(c) **Draw** a graph of the training loss of each algorithm at each iteration and another graph of the test loss. Use the 0-1 loss for both losses.

(d) **Draw** the learned hypotheses for all 4 algorithms (3 SGD, 1 GD) using *showImage* that is located in *readMnist* at iteration $5, 15, 50, 100$(Make sure you take off the 1 at the end of the weights vector) .
"OMG"/"How did it do that" is what you should say after observing those weights :)

(e) **Question:** The SGD with the batch tries to imitate the GD but with less samples as you can see. Analyze the results in the last two bullets, and explain them. Try and understand the tradeoff between the batch size and the computational complexity and the accuracy of the subgradient in the learning process.

(f) Run through b and d again, but with eta = 0.1, 10. What can we learn from the learning rate results in this specific problem?
For what cases should we consider high learning/ low learning rate. You may consider the gradient of $f(x) = 10^8 \cdot |x|$ and $g(x) = 10^{-8} \cdot |x|$ at different points and its effect in GD when you answer the question.

(g) Bonus: Find a pair of Mnist digits that the GD or the SGD doesn't succeed very well on them and suggest an explanation for the bad results (5 points bonus).

The SGD is quite useful in real world problems. In the next exercise will use SGD on a non convex problem- such as a neural network!