CS-GY 9223G / Fall 2020

Homework 1

0. **(0.5 points)** Introduce yourself on the HW1Q0 thread on the class Piazza! Mention a little bit about your background, interests, and why you wish to learn about neural nets and deep learning.

   Solution -

1. **(1.5 points)** *Fun with vector calculus*. This question has two parts.

   a. Assume we have n real-valued scalar data points $x_1, x_2, \ldots, x_n$. Analytically derive a constant μ for which
   $$\sum_{i=1}^{n} (x_i - \mu)^2$$
   is minimized.

   b. Now assume that the n data points are real d-dimensional vectors. Analytically derive a constant vector μ for which
   $$\sum_{i=1}^{n} \|x_i - \mu\|_2^2$$
   is minimized.

   Solution

   For both parts, take the derivative, set to zero. The answer is the same:
   $$\mu = \frac{\sum_{i=1}^{n} x_i}{n}.$$

   In part b, you have to properly handle the squared Euclidean norm. Either decouple it into coordinates (so the problem reduces to part a) or take the vector derivative (which resembles the scalar derivative of a quadratic function).

2. **(2 points)** *Linear regression with non-standard losses*. In class we derived an analytical expression for the optimal linear regression model using the least squares loss for a finite dataset. If X is the matrix of training data points (stacked row-wise) and y is the vector of labels, then:

   a. Using matrix/vector notation, write down a loss function that measures the training error in terms of the ℓ_1-norm.

   b. Can you write down the optimal linear model in closed form? If not, why not?

   c. If the answer to b is no, can you think of an alternative algorithm to optimize the loss function? Comment on its pros and cons.

   Solution

   a. $L(w) = \|Xw - y\|_1$.

b. No! Unlike ℓ_2, There is no closed form expression for ℓ_1-regression. The reason is because the ℓ_1-norm is non-differentiable. (Try reasoning about why this is the case.)

c. Regular gradient descent wouldn't work either due to the non-differentiability. Acceptable answers include: smoothing out the ℓ_1-norm around $w = 0$ (by say using a Huber norm) before gradient descent; performing *sub*-gradient descent; performing iterative soft thresholding; or using linear programming (LP). Cons for most of these approaches are their relatively high computational cost when compared to regular gradient descent.

3. **(2 points)** *Hard coding a multi-layer perceptron.* In class, we derived the functional form for a single perceptron: $y = \text{sign}(\langle w, x \rangle + b)$, where x is the data point. Suppose your data is 5-dimensional (i.e., $x = (x_1, x_2, x_3, x_4, x_5)$) and real-valued. Find a simple 2-layer network of perceptrons that implements the *Increasing-Order* function, i.e., it returns +1 if

$$x_1 < x_2 < x_3 < x_4 < x_5$$

and -1 otherwise. Your network should have 2 layers: the input nodes, feeding into 4 hidden perceptrons, which in turn feed into 1 output perceptron. Clearly indicate all the weights and biases of all the 5 perceptrons in your network.

Solution

The idea is intuitive (although involves some creativity). Here is a solution (among maybe other ones). The first (hidden) layer can implement pairwise comparisons ($x_1 < x_2, x_2 < x_3, x_3 < x_4, x_4 < x_5$). Each of these can be implemented using perceptrons; for example, the first comparison is achieved by the following perceptron:

$$y_1 = \text{sign}(-x_1 + x_2)$$

the second by:

$$y_2 = \text{sign}(-x_2 + x_3)$$

and so on. The final output can be obtained by making sure that *all* of the hidden layers' outputs are turned on. This can be done by setting the bias to (say) 3.9.

$$z = \text{sign}(y_1 + y_2 + y_3 + y_4 - 3.9).$$

4. **(4 points)** This exercise is meant to introduce you to neural network training using Pytorch. Open the (incomplete) Jupyter notebook provided as an attachment to this homework in Google Colab (or other cloud service of your choice) and complete the missing items. Save your finished notebook in PDF format and upload along with your answers to the above theory questions in a single PDF.

Solution

A solution notebook has been posted on NYUClasses.