CS7180: Algorithmic and Statistical Aspects of Deep Learning Fall 2020

Problem Set 2

Instructor: Hongyang R. Zhang *Due:* ***October 26, 2020, 11:59pm***

Policy: We encourage discussions and collaborations on the homework. You should write up the solution on your own, and remember to mention any fellow students you collaborated with. There are up to three late days for all the problem sets and project submissions. Use them wisely. After that, the grade depreciates by 20% for every extra day. Late submissions are allowed only under extreme situations. Please discuss with the instructor well in advance if you cannot meet the deadline. All homework submissions are subject to the Northeastern University Honor Code.

Submission: We will use both Canvas and Gradescope for the homework submissions. Please submit your written solutions to Gradescope and upload your code to Canvas. Login to Gradescope through Canvas using your northeastern.edu account. We strongly recommend that you write up your solution in LaTeX, although you are allowed to send in scanned copies.

Length of submissions: Include as much of the calculations needed for us to understand the answer. After solving the problem, try to identify the main steps taken and critical points of proof and include them as a rule of thumb.

Problem 1

In this problem, we explore vanishing gradient issues in a multi-layer perceptron. We consider the MNIST dataset that has been introduced in problem set one. Instead, we construct a multi-layer neural network with rectified linear unit activations and use the cross-entropy loss (softmax plus negative log-likelihood) to evaluate the predicted result.

(a) **[5 points]** Explain what the vanishing gradient phenomenon is. Could you explain why it happens? By constrast, explain what the exploding gradient phenomenon is, and explain why it happens.

(c) **[25 points]** Open the gradient norm notebook that we handed out. Implement a multi-layer perceptron with 4 layers (including 3 hidden layers and 1 output layer). Follow the instructions in the notebook. Write down the discussions mentioned in "Part 3".

Problem 2

In this problem, we consider the matrix sensing problem that was mentioned briefly during the lecture. Suppose that we want to learn an unknown positive semidefinite matrix $X^\star = U^\star U^{\star\top}$, where $U^\star \in \mathbb{R}^{d\times r}$ is a rank-r matrix. We are given m input samples that are linear measurements of $X^\star$. For each sample $i = 1, 2, \ldots, m$, let $A_i \in \mathbb{R}^{d\times d}$ be a random matrix where every entry of A_i is drawn independently from a Gaussian distribution with zero mean and unit variance. The label of A_i, denoted by y_i, is equal to

$$y_i = \langle A_i, X^\star \rangle.$$

In the above notation, the inner product of A_i and $X^\star$ is defined as the sum of the entrywise product of both matrices. The problem that we would like to solve is, given $(A_1, y_1), (A_2, y_2), \ldots, (A_m, y_m)$, how can we recover $X^\star$?

(a) [**5 points**] We are going to set up an optimization problem to learn $X^\star$. Let $U \in \mathbb{R}^{d\times R}$ denote the variable matrix. We minimize the following mean squared loss.

$$f(U) = \frac{1}{2m}\sum_{i=1}^{m}(\langle A_i, UU^\top\rangle - y_i)^2.$$

Compute the gradient of $f(U)$ over U. Write down the gradient descent algorithm for minimizing $f(U)$.

(b) [**20 points**] Note that the rank of U is equal to R. In this part, we assume that $R = r$. Write a python file to implement the gradient descent algorithm. Let $t \geq 1$ denote the current epoch. Plot the distance between $U_t U_t^\top$ and $X^\star$ as a function of t.

Note: (i) use the following parameters in the implementation, $d = 100$, $r = 5$, $m = 10 \cdot d \cdot r$. (ii) The distance between two matrices U and V is the Frobenius norm of $U - V$.

(c) [**25 points**] To prove what we have observed above, we are going to divide the analysis into two parts (the second part is a bonus). In particular, we will consider the rank-1 case, where $r = 1$ (instead of 5). In the first part, we assume that the number of samples m is infinitely large. This is a hypothetical setting, but it is a good start for the analysis. Show that if R is also equal to one, then the only second-order stationary point of $f(U)$ is $\pm U^\star$.

(d) [**10 points**] In this part, we assume that the gradient descent algorithm is over-parametrized. More specifically, suppose that $R = d > r$. We consider different initialization to the gradient descent algorithm. Suppose that W is a random initialization where every entry of W is sampled independently from a Gaussian distribution with mean zero and unit variance. Use initialization $U_0 = \alpha \cdot W$, for α from 10E-5, 10E-4, 10E-3, 10E-2, 10E-1. Plot the distance between $U_t U_t^\top$ and $X^\star$ for each initialization U_0 as a function of t.

(e) [**10 points**] Describe what the implicit regularization hypothesis is. What does the hypothesis

suggest in the setting of part (d)? To verify whether this hypothesis is correct or not, plot the top ten largest singular values of U_t as t grows.

(f) **[Bonus]** This is the second part of the analysis following part (c). Could you extend the analysis in part (c) to finite samples, that is, when $m = O(d \cdot \text{poly}(r \log d))$? [Hint: consider the direction of improvement idea for dealing with approximate second-order stationary points]