Problem Set #1

Moni Shahar
Deep Learning
Tel-Aviv University

November 2, 2021

Due date: November 22, 2021

Question 1

In this question we will run the back propagation algorithm manually, for the given input $X = (1, 2, -1)$ and an output $Y = 0$. Let the neural network N be defined as follows:

- 3 inputs x_1, x_2, x_3.
- Two hidden layers $H^{(1)}, H^{(2)}$ where both $H^{(1)}$ and $H^{(2)}$ contains 2 neurons. The activation function of all hidden layers is RelU.
- An output layer with single output with identity function as activation.
- all the weights of the network are initialized to 1

Assuming the loss function is RSS, calculate the partial of derivatives of the loss at point (X, Y).

Question 2

Write a Python code to implement forward and backward steps on a neural network.

- The code should support only fully connected layers
- The code should support sigmoid and RelU as non-linearities
- There is no need to get a textual definition of the layers
- You may assume the input given is correct
- You may assume all weights are initialized as a part of the input
- **Bonus**: support also batch normalization layers

Check your answer to question number 1 using your code.

Question 3

3.a

Given a set of N points in general position in the plane $\{(x_1^1, x_1^2), \cdots, (x_N^1, x_N^2)\}$, and a real vector $Y = (y_1, \cdots, y_N)$, prove that there exists a neural network with one hidden layer and step function activation, that can fit X, Y exactly.
Hint: Use a similar construction to what was done in class for classification. You will need to prove some condition on the matrix obtained.
Bonus: Generalize your result to points in $\mathbb{R}^d$

Question 4

In this question we implement a gradient descent solution for linear regression.

4.a

Prepare the data to check the solution.

- X is a set of 10000 points in $\mathbb{R}^4$. The points are drawn uniformly in $[0,1]^4$.
- The noise ϵ is i.i.d. and distributed normally with $\sigma^2 = 1$
- $Y = X_1 - 2X_2 + 3X_3 - 4X_4 + \epsilon$

4.b

Implement the Gradient Descent (GD) algorithm. The algorithm gets (i) a step size (learning rate), (ii) an initial solution and (iii) a function that returns the gradient of the RSS in a given point. The GD algorithm starts from an initial guess, and change it iteratively by moving in the opposite direction of the gradient. The algorithm runs a fixed number of iterations or until convergence.
The iteration can be written as follows:

$$W_{i+1} = W_i - \alpha \nabla_W \text{Loss}$$

Where α is the learning rate.

4.c

Check how each of the following changes affect the preformance:

- Exponential decay of the step-size
- Calculate the gradient in batches, namely, implement Stochastic Gradient Descent.
- Change the update step to include momentum, specifically use the following update:

  $$v_{i+1} = \gamma v_i + \alpha \nabla_W \text{Loss}$$

  $$W_{i+1} = W_i - v_{i+1}$$

  Where $\gamma \in (0, 1]$. **Note:** gamma may also be changed during the algorithm, what would be the impact of increasing it?