UNIVERSITÄT DES SAARLANDES
Prof. Dr. Dietrich Klakow
Lehrstuhl für Signalverarbeitung
NNTI Winter Term 2020/2021

Exercise Sheet 6

Computation Graphs and Gradient Descent

Deadline: 05.01.2021, 23:59

Exercise 6.1 - Taylor Series (1 + 1 + 1 points)

The commonly used activation function in hidden layers of a Neural Network is the Sigmoid function which is defined as:

$$\sigma(x) = \frac{1}{1+e^{-x}}$$

a) Prove that the derivative of the sigmoid function is $\sigma(x) - \sigma^2(x)$. (1 point)

b) We know from Newton's method the importance of Taylor series in optimization, additionally, Taylor expansion could be beneficial in providing a cheaper computation alternative for activation functions (for further reading: http://www.yildiz.edu.tr/~tulay/publications/Tainn2003-3.pdf).

So, find the first 3 terms in the Taylor series for the sigmoid function centered at 0. Hint: You can use the derivative form proved in (a) when calculating higher derivatives. (1 point)

c) Let's consider $f(w_n)$ to be a neural model, and assume that it has a single weight matrix w_n as its parameters, with n indicating the training step. Now, with the following definitions:

- $g(n) = \nabla f(w_n)$
- $\epsilon \in \mathbb{R}_+$ step size or learning rate
- $w_{n+1} = w_n - \epsilon g(n)$

Show that applying gradient descent on $f(w_n)$ pushes the error function towards a local minimum. Do this by approximating $f(w_{n+1})$ at the point w_n with only the first two terms of its Taylor series. (1 point)

Exercise 6.2 - Computation Graphs (1 + 1 + 3 points)

In the first assignment you were asked to implement a simple linear model for XOR, with $y \in \{0, 1\}$:

$$y = \boldsymbol{W}^T\boldsymbol{x} + b \tag{1}$$

You observed that this model cannot separate the two classes. Now consider the following expansion of model 1 that adds a hidden layer and non-linear activation functions:

$$y = \sigma(\boldsymbol{W_o}^T(\sigma(\boldsymbol{W_i}^T\boldsymbol{x} + b_a)) + b_o) \tag{2}$$

Where $\boldsymbol{x}$ is any of the XOR inputs, $\boldsymbol{W_i}$ is a 2×2 vector, $\boldsymbol{W_o}$ is a 2×1 vector and σ is the sigmoid function.

Let's use Mean Squared Error as the loss function, such that for N samples we want to optimize

$$L(\boldsymbol{W_i}, \boldsymbol{W_o}, b) = \frac{1}{2} \sum_{n=1}^{N} \Big(\sigma(\boldsymbol{W_o}^T(\sigma(\boldsymbol{W_i}^T \boldsymbol{x_n} + b_a)) + b_o) - y_n \Big)^2 \tag{3}$$

a) Draw a computation graph of 3, explicitly name each leaf and annotate it with the corresponding computations. You can ignore the sum and draw the graph for a single sample. If you have trouble drawing the graph by hand, try this resource. You may use intermediate symbols for computation steps, such as a for the activation of a neuron and z for $\sigma(a)$. (1 point)

b) Based on the computation graph you obtained in a), write down the partial derivatives of L with respect to the model parameters. Consider that $\boldsymbol{W_i}$ and $\boldsymbol{W_o}$ contain 4 and 2 parameters, whereas b_a and b_o are single parameters. The partial derivatives should be with respect to these parameters. (1 point)

c) Implement the computation graph in PyTorch beginning from the starter code in the ipython notebook. Follow the instructions in the notebook, and add your explanations in the Markdown cells.

   Learning XOR with only two hidden neurons is a bit tricky, you can adjust $\boldsymbol{W_i}$ and $\boldsymbol{W_o}$ such that the size of the hidden layer is large enough. (3 points).

Exercise 6.3 - Local Minima and Optima (0.5 + 0.5 + 1 points)

The paper by Dauphin et al. (2014) discusses the problem of saddle points in high-dimensional optimization problem. Read the paper carefully. You do not need to understand everything, but you should get a general idea about what is challenging about saddle points and how different training approaches deal with that.

Then, answer the following questions in up to 5 sentences each:

a) What happens to the eigenvalues of the Hessian as training error decreases, and why? (0.5 points)

b) What problems arise at saddle points for *gradient descent*, *Newton method* and *Trust region* optimization algorithms, and how are they related to the eigenvalues of the Hessian? (0.5 points)

c) How do Dauphin et al. (2014) tackle these problems with their *Saddle-free Newton algorithm* and how is it related to the original Newton method? (1 point)

Submission instructions

The following instructions are mandatory. If you are not following them, tutors can decide to not correct your exercise.

- You can and are encouraged to submit the assignment as a team of two students. Submitting as a team will be mandatory for the next assignment.

- Hand in zip file containing the PDF with your solutions and the completed ipython notebook.
- Therefore Make sure to write the Microsoft Teams user name, student id and the name of each member of your team on your submission.
- Your assignment solution must be uploaded by only **one** of your team members to the 'Assignments' tab of the tutorial team (in **Microsoft Teams**).
- If you have any trouble with the submission, contact your tutor **before** the deadline.