Deep Learning for NLP 2020
Home Exercise 03

Due on Monday, 11.05. at 18:00

May 4, 2020

Submission Guidelines for all Home Exercises

- When submitting multiple files, submit one **zip-archive**.
- Submit python code as plain python scripts (**.py**). **Must** be runnable in the given Docker container.
- Submit answers to non-code assignments in **one PDF** file. Scans are permitted, if readable.
- Guidelines specific to neural network code:
  - Please submit your training/testing results (a copy of your console output is fine). Reasoning: Your network might train much slower on the tutor's system than on yours.
  - If you are aware that your network never stops training, please be honest and add a short statement saying so. Thank you!

1 Backpropagation by Hand (4P)

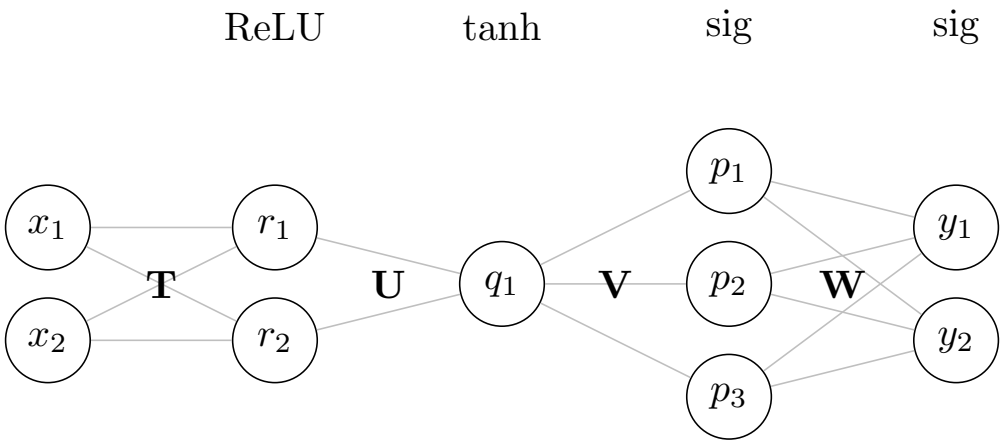

Figure 1: MLP for Backpropagation

Network Details Figure 1 shows an MLP without bias neurons. The activation function used in each layer is shown above the respective layer.

For the loss function, square loss is used. t_j denotes a true label, y_j denotes a network output:

$$\ell(t_j, y_j) = (t_j - y_j)^2 \qquad \frac{\partial \ell(t_j, y_j)}{\partial y_j} = 2(y_j - t_j)$$

The weight matrices are initialized as follows:

$$\mathbf{T} = \begin{pmatrix} 0.19 & -0.92 \\ -0.42 & -0.28 \end{pmatrix} \quad \mathbf{U} = \begin{pmatrix} 0.61 \\ -1.5 \end{pmatrix} \quad \mathbf{V} = \begin{pmatrix} 1.5 & -0.81 & -0.24 \end{pmatrix} \quad \mathbf{W} = \begin{pmatrix} -1.4 & -0.81 \\ -2.2 & -1.7 \\ -0.27 & -0.73 \end{pmatrix}$$

Task Using pen and paper and the backpropagation algorithm, calculate the gradients for these weights:

$$\frac{\partial E}{\partial w_{1,1}} \quad \text{and} \quad \frac{\partial E}{\partial v_{1,3}} \quad \text{and} \quad \frac{\partial E}{\partial t_{1,1}}$$

Use $\mathbf{x} = (-1, 1)$ as the input and $\mathbf{t} = (0, 1)$ as the output. Round your result to **four** decimal points after each calculation. Report your calculation process where necessary.

The formulas can be found in the `tu03` slides (see Moodle).

Hint: One of the three gradients can be determined without much calculation.

2 Sentiment Polarity in Movie Reviews (6P)

We revisit last week's task of identifying sentiments in movie reviews with an MLP and TensorFlow.

Data (same as in home exercise 02) In the `hex03_data` archive, you can find a training, development and test dataset. Each line in these datasets has three entries, separated by a tab character (`\t`). The first is the movie review (available only for reference), the second is the sentiment label (`POS`itive or `NEG`ative). To facilitate the task, the third entry is a 100-dimensional vector representing the review (we'll cover in later lectures on *word embeddings* how this sentence representation has been generated).

2.1 Dataset reader (0P)

To read the datasets, reuse your own reader from home exercise 02 or copy the one from the solution soon provided on Moodle.

2.2 MLP in TensorFlow (3P)

Implement a multilayer perceptron in TensorFlow. Try to keep it as configurable as possible for the hyperparameter optimization following in 2.3.

Report loss and accuracy on the dev set for the following configuration:

- square loss
- 2 hidden layers, both with a dimension of 50

- weights initialized by the standard normal distribution (mean 0, sigma 1)
- mini-batches of size 10
- learning rate $\alpha = 0.01$
- 20 epochs

2.3 Hyperparameter Optimization (3P)

In lecture 02 you learned about random hyperparameter optimization. Try to find the best configuration for your MLP from task 2.2 by experimenting with the following hyperparameters:

A: Known from the previous home exercise

- batch size
- learning rate
- number of training epochs

B: New parameters

- loss function: square loss, hinge loss, softmax + cross-entropy loss
- number of hidden layers: any values from the interval $[0, \infty[$
- hidden layer dimension: any values from the interval $[1, \infty[$
- activation function(s): sigmoid, tanh, ReLU, softplus, leaky ReLU, ELU, maxout
- 2-norm regularization of hidden layer weights (see lecture 03): yes/no
- optimizer: SGD, Adam, AdaGrad, etc.

To obtain the 3P for this task, you need to play around with all three parameters from set **A** and at least three parameters from set **B**. Create at least 10 hyperparameter sets by sampling random values and train with each on the training set. Report your top 3 parameter sets and their loss/accuracy on the dev set. Then, report loss and accuracy of your best parameter set on the test set.

Hints:

- Activation functions and hidden layer dimensions can also be chosen differently for each layer.
- When varying the loss function or the activation functions, it might make sense to redefine the truth labels in the datasets from 0 for NEGative and 1 for POSitive to something else (think of the co-domain of activation functions).