DL4NLP 2022 — Exercise 4

Jonas Belouadi, Steffen Eger
Natural Language Learning Group (NLLG),
University of Bielefeld,
Summer Semester 2022

To prepare for the tutorial, you can already install the dependencies listed in `requirements.txt` and read the documents linked in the hints.

Task 1 (10min): TensorFlow Playground

TensorFlow offers a playground for experimenting with neural networks in a web browser[1].

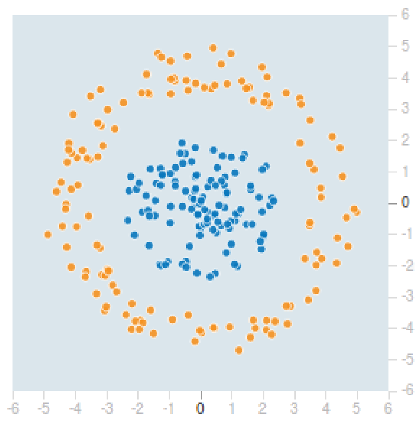

Figure 1: A circular dataset from the TensorFlow Playground

(a) Take a look at the circular data in Figure 1. Do you think a perceptron architecture (no hidden layers) can learn a good discriminator for this dataset? If so why?

(b) In a multi-layer perceptron the number of neurons can be chosen differently for each hidden layer. Pick the spiral dataset and specify at least four hidden layers. Then, try out two different scenarios:

- More neurons towards the input, less neurons towards the output.
- Less neurons towards the input, more neurons towards the output.

Which scenario produces the best results on the test set? Which scenario converges the fastest?

Task 2 (30min): Sentiment Classification

The movie-review dataset[2] consists of movie reviews labeled with sentiment polarity (i.e. "positive review" or "negative review"). Using TensorFlow and its high-level `tf.keras` API, your task is to implement a multi-layer perceptron which

[1]`https://playground.tensorflow.org`
[2]`http://www.cs.cornell.edu/people/pabo/movie-review-data`

learns to identify the sentiment in a review. For that matter, complete the code in `task3_mlp.py`. In the `datasets` folder, 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 embedding vector representing the review.

(a) Given what you know about word embeddings, how do you think the single vector representing the entire review was created? What is the problem with using word embeddings directly?

(b) Considering the binary nature of the dataset labels, which output activation would be a natural fit for this problem? Add your choice in `ACTIVATION HERE`.

Hints:

- For a quick overview of TensorFlow basics consider reading reading `https://www.tensorflow.org/guide/basics`.
- For information on the Keras Sequential API and Keras subclassing API refer to `https://www.tensorflow.org/tutorials/quickstart/beginner` and `https://www.tensorflow.org/tutorials/quickstart/advanced`.

(c) Add two hidden layers with 50 neurons each and `tanh` activation function to the MLP. Add the missing code in `HIDDEN LAYERS HERE`.

(d) Report accuracy and loss on the development set. Add the missing code in `EVALUATION HERE`.

(e) Train the model with the given parameters. Which result do you get on the development set?

Task 3 (20min): Custom Layers

Most of the time, when writing neural networks, you operate at a high level of abstraction, i.e. using the `tf.keras` API, which provides a set of many common layers. But sometimes you need to write your own, custom ones. In this task you will implement your own densely-connected layer using only low-level operations and use it in your MLP from Task 2. Complete the code in `task4_layer.py` and use it in your MLP in `task3_layer.py`.

(a) Add the missing code in `YOUR CODE HERE`.

Hints:

- See `https://www.tensorflow.org/tutorials/customization/custom_layers#implementing_custom_layers` for an overview on custom layers.
- In this task it is permissible to omit support for a bias neuron.

Bonus Task: Hyperparameter Optimization

Perform a random hyperparameter optimization. Try to find the best configuration for your MLP from Task 2 by experimenting with at least three of the following hyperparameters:

- batch size
- learning rate
- number of training epochs
- loss function
- number of hidden layers
- hidden layer dimension

- activation function
- 2-norm regularization of hidden layer weights (yes/no)
- optimizer

Evaluate your parameter sets on the development set. Then, report loss and accuracy of your best parameter set on the test set.

Bonus Task: Weight Matrix Initialization

There are many reasons why a neural network “refuses to learn”. Improper weight matrix initialization is one such issue which has strong consequences for the overall training convergence.

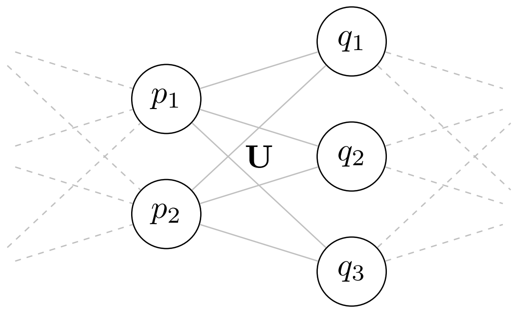

Figure 2: Standard multi-layer perceptron with two hidden layers.

(a) Take a look at the MLP in Figure 2. Imagine initializing all weights in matrix $\mathbf{U}$ to the same non-zero value (e.g., 1). Why is this a suboptimal choice?

(b) Now imagine initializing all weights in matrix $\mathbf{U}$ to zero. How does this affect the gradients at the hidden units in layer p during backprogagation?