Deep Learning for NLP 2021
Homework 1

Due on April 29, 2022 at 23:59

TECHNISCHE UNIVERSITÄT DARMSTADT

Version of April 13, 2022

Overall: 13 points.

Don't forget to sign up in groups of two on Moodle until **April 22**! The submission on Moodle will be possible as soon as the group selection is finalized.

Submission Guidelines

Submit a zip archive containing:

- **one PDF** file answering the questions **and** containing relevant console outputs (e.g., for training/testing results),
- **one plain python script** called `p5.py` that is runnable with the conda environment of this course
- the **unzipped DATA folder** (from `hw01_data.zip`).

Name the zip archive you submit `hw1_num1_num2.zip` where `num1` and `num2` are the matriculation numbers of the two team members (e.g., the file name could be `hw1_1234567_9876543.zip`).
Please comment your code such that tutors can read it easily. This way, we may be able to give you some points even if your code does not run and/or if we cannot reproduce your results.
If you are aware that your network never stops training, please be honest and add a short statement saying so. Thank you!

1 Setup (0P)

Follow the instructions in the dl4nlp_env.zip file to install the conda environment that will be used to grade your homework.

You can verify that your setup works by running the Keras MNIST example(`mnist_mlp.py` in `hw01_data.zip`).

2 Sigmoid Activation Function (2P)

When optimizing functions, first or higher-order derivatives (gradients, Hessians) are of major importance. In neural network learning, we typically want to minimize weight parameters so that the difference between the net output and the true labels is minimized, e.g.

$$\min_{\mathbf{w}} \sum_{j=1}^{N} \Big(\sigma(\mathbf{x}_j \cdot \mathbf{w}) - y_j \Big)^2 .$$

Here, σ is an activation function. A frequently used activation function is the *sigmoid* function, defined as:

$$\text{sig}(x) = \frac{1}{1 + \exp(-x)}.$$

Show that:

$$\text{sig}'(x) = \text{sig}(x) \cdot \big(1 - \text{sig}(x)\big).$$

You may find the chain rule useful: $f(g(x))' = f'(g(x)) \cdot g'(x)$.

3 TensorFlow Playground (3P)

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

3.1 Cicular Dataset (1P)

Take a look at the circular dataset in figure 1. Can a perceptron architecture (no hidden layers) learn a good discriminator for this dataset? Justify your answer. Write at most two sentences.

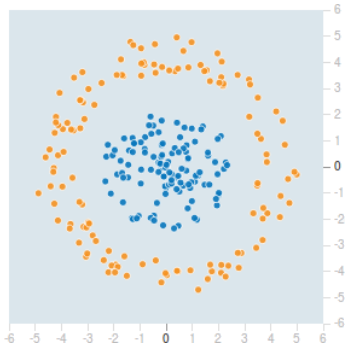

Figure 1: Circular dataset

3.2 MLP (2P)

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 three different scenarios:

- Same amount of neurons in every hidden layer.
- 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? Why do you think that is the case? Explain in up to three sentences.

[1] `https://playground.tensorflow.org`

4 Softmax (1P)

The softmax function (with temperature $T > 0$) is a mapping from a vector $\mathbf{z} \in \mathbb{R}^n$ to a vector $\mathbf{y} \in \mathbb{R}^n$ and is defined as:

$$\mathbf{y} = (y_1, \ldots, y_n) = \text{softmax}(\mathbf{z}) \quad \text{where} \quad y_i = \frac{\exp(z_i/T)}{\sum_{j=1}^{n} \exp(z_j/T)}, \forall i = 1, \ldots, n$$

In the domain of neural networks, the softmax function is oftentimes considered to be an activation function. What is the fundamental difference of the softmax activation function compared to conventional activation functions such as hyperbolic tangent function or sigmoid? State your answer in one sentence.

5 Sentiment Polarity in Movie Reviews (7P)

The movie-review dataset[2] consists of movie reviews labeled with sentiment polarity (i.e. "positive review" or "negative review"). Your task is to implement a variation of the perceptron from exercise 1 which learns to identify the sentiment in a review. You can use the provided solution from last week as a reference.

Data In the `hw01_data.zip` 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).

Perceptron

- As the loss function, choose square-loss:

$$L = \sum_{j=1}^{N} \ell(\mathbf{x}_j, y_j) = \sum_{j=1}^{N} (\sigma(\mathbf{x}_j^\intercal \mathbf{w}) - y_j)^2$$

- For the activation function, use the sigmoid function.
- For the weight update rule, use the following mini-batch stochastic gradient descent formula:

$$\mathbf{w}' \leftarrow \mathbf{w} - \frac{\alpha}{|\mathcal{T}'|} \cdot \sum_{(\mathbf{x},y)\in\mathcal{T}'} \Big(\sigma(\mathbf{x}^\intercal \mathbf{w}) - y\Big) \cdot \sigma'(\mathbf{x}^\intercal \mathbf{w}) \cdot \mathbf{x}$$

  Reminder: $\mathcal{T}'$ is a mini-batch; a random subset of the whole training data $\mathcal{T}$. A typical way of implementing random mini-batches is to randomly shuffle the whole training dataset before each epoch, then divide the training dataset into batches of size $|\mathcal{T}'|$. Set the NumPy random seed for reproducible results.

- Use the 100-dimensional vectors from the datasets for the input vectors $\mathbf{x}$. Encode the corresponding label as $y = 1$ for `POS` and $y = 0$ for `NEG` (i.e. according to the co-domain of the sigmoid activation function). **Add a bias, i.e., append a trailing 1 to each input vector x.**
- Initialize the weight vector via

```
w = np.random.normal(0,1,(N,1))
```

  where N is the dimensionality of your input data.

[2] `http://www.cs.cornell.edu/people/pabo/movie-review-data/`

If you are unfamiliar with NumPy, these resources may be helpful:
NumPy: the absolute basics for beginners
NumPy quickstart

5.1 Dataset Reader (1P)

Implement a reader for the dataset files which returns the input vectors **x** and labels y as numpy arrays. The shape and number of returned arrays is up to you.

5.2 NumPy Implementation (4P)

Implement the perceptron stated above **only using NumPy**. Include a method which computes the square loss and the accuracy of the model, given a dataset and a weight vector **w**.

Hint: In order to compute the accuracy, you need to find a meaningful way to interpret your perceptron's prediction $\sigma(\mathbf{x} \cdot \mathbf{w})$ for a given test input **x** and trained weights **w**.

5.3 Training (2P)

Train your perceptron (**also only using NumPy**) on the training data and observe its accuracy on the **development** set. Start with batch size $|\mathcal{T}'| = 10$, learning rate $\alpha = 0.01$, and 50 epochs. Report loss and accuracy on the dev set.

Experiment with different values for these three hyperparameters. Can you find a configuration which beats 70% accuracy on the development set? Report your best configuration and both the loss and accuracy it reaches on the **development and test** sets.