# Deep Learning for NLP 2019
# Home Exercise 02

TECHNISCHE UNIVERSITÄT DARMSTADT

**Due on Tuesday, 30.04. at 13:00**

**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!

## Problem 1 TensorFlow Playground (3P)

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

a) 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. (1P)

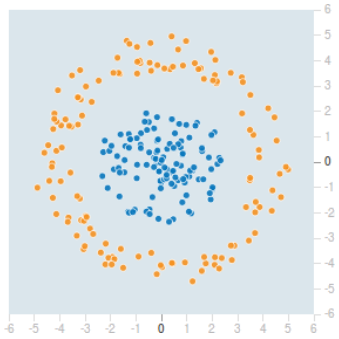


**Figure 1:** Circular dataset

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 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.

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

Which scenario produces the best results on the test set? Which scenario converges the fastest? Explain in up to three sentences. (2P)

## Problem 2 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 last week which learns to identify the sentiment in a review.

### Data

In the `hex02_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).

### Perceptron

- As the loss function, choose square-loss:

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

- For the activation function, use the sigmoid function.

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

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

  Reminder: $\mathscr{T}'$ is a mini-batch; a random subset of the whole training data $\mathscr{T}$. 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 $|\mathscr{T}'|$. Consider setting 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.

### Problem 2.1 Dataset reader (1P)

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

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

## Problem 2.2 Numpy implementation (6P)

a) 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 $w$. (4P)

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 $\mathbf{x}$ and trained weights $\mathbf{w}$.

b) Train your perceptron 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 100 epochs. 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. (2P)