# Exercise 4 - Classification

January 14, 2020

# 1 Decision Trees

a) ID3 Decision Tree Construction (3 points)

You are working at a bank which needs to assess when to give credits and when not. You are given the following labeled dataset with the goal to construct a model for predicting whether to give a credit or not. Construct a decision tree using pen and paper with the information gain / entropy based splitting criterion. Draw the tree and indicate each splitting attribute. Show your calculations.

| Age | Educational Degree | Married | Income Level | Give Credit? |
|---|---|---|---|---|
| old | secondary | no | high | yes |
| old | college | no | high | yes |
| old | college | no | high | yes |
| young | secondary | no | high | yes |
| old | secondary | yes | medium | yes |
| young | college | yes | low | yes |
| young | college | no | high | yes |
| old | secondary | no | medium | yes |
| young | primary | yes | medium | no |
| young | secondary | yes | medium | no |
| old | primary | yes | low | no |
| young | college | no | medium | no |

## 2 Neural Networks

a) Hyperparameter exploration (2 points)

Try varying the hyperparameters of a multi-layer perceptron on the TensorFlow Playground webpage (`https://playground.tensorflow.org`) using the *Circle* classification dataset (see figure below). Try different numbers of hidden layers and different numbers of neurons per layer. Does it work better to a) use more neurons near the input layer, b) more neurons towards the last hidden layer or c) use the same number of neurons in each hidden layer? Provide a justification for why a), b) or c) might work better.
Can a perceptron (without hidden layers) separate the circular dataset? Why or why not?

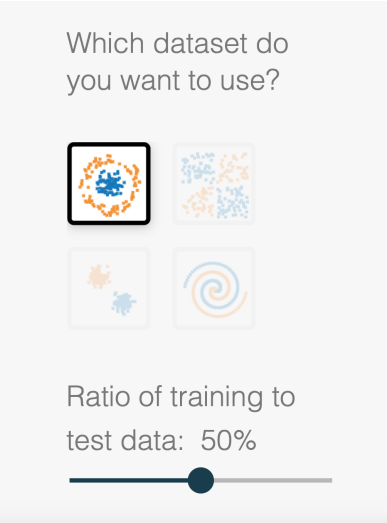


Figure 1: Mandatory settings on TensorFlow Playground for this exercise

b) Multi-Layer Perceptron for Sentiment Analysis (5 points)

Implement a multi-layer perceptron for classifying the movie reviews in the folder *data/moviereviews* as positive or negative using PyTorch (`https://pytorch.org/`). You can find the movie review texts in the file *reviews.txt*. The labels can be found in the file *labels.txt*, each line $i$ contains the label for the movie review in line $i$ of *reviews.txt*. You have to map each of the movie review texts to a fixed-size embedding vector, which you can use as input to your multi-layer perceptron. You can do so using the flair library[1]. Install it using *pip install flair*. You can use it to encode text as follows (there are multiple ways - pooling of pre-trained word vectors, recurrent neural nets, etc.):

```
from flair.embeddings import WordEmbeddings, DocumentRNNEmbeddings,
    FlairEmbeddings, DocumentPoolEmbeddings, Sentence

# instantiate pre-trained word embeddings
word_embeddings = WordEmbeddings("glove")

# document pool embeddings - you can try to exchange this with
    DocumentRNNEmbeddings!
document_embeddings = DocumentPoolEmbeddings([embeddings], fine_tune_mode='none'
    )

# create an example sentence object
sentence = Sentence("Colorless green ideas sleep furiously.")

# embed the sentence with the document embeddings (needed for each movie review)
document_embeddings.embed(sentence)

# check out the embedded sentence - it's a torch.Tensor object :-)
print(sentence.get_embedding())
```

Listing 1: Embedding documents using flair

Perform a 3-fold cross validation and report precision and recall. Report your hyperparameter configuration (learning rate, batch size, network structure, etc.).

[1] https://github.com/flairNLP/flair