**HS Machine Learning**
Lecturer: Çağrı Çöltekin
Summer 2016

# Assignment 03

Daniela Stier
August 01, 2016

1. **Report of accuracies for Logistic regression (LG), Multi-Layer Perceptron (MLP) and Convolutional Neural Network (CNN)**
   Report average accuracy of your models.

   (Metrics for positive = class 1, negative = class 0)

| | LG | MLP<br>embedding learned from scratch: own word vector with averaging | CNN |
|---|---|---|---|
| **Accuracy** | 0.843 | 54.8 | 78.7 |
| **Standard Error** | 0.0061318838867 | 1.04211323761 | 3.02258167797 |

- **Notes concerning the data given as input:**
  In a pre-processing step, stopwords and sentence markers have been deleted from both uni- and bigrams. Before, the effect of stopwords in each model (LG, MLP, CNN) has been tested: Apparently, the obtained accuracies did not vary to a huge extent (about <= 1%). Because results for input data excluding stopwords generally revealed higher values, stopwords have been excluded from uni- and bigrams for all created models. For this reason, and in order to train all models on the same data, the *n_gram_range* provided by *sklearn*'s *CountVectorizer* (employed for LG) was not used. Instead, bigrams (excluding stopwords) were created in a pre-processing step and then fed to the individual models. In the case of CNN, bigrams were excluded from the input data, and in a later step included with the help of filter lengths and number of maps.
- **Notes concerning (average) word vectors:**
  For both, MLP and CNN, no huge differences could be found for input in the form of simple word vectors (mere sequences of indices) and averaged word vectors (about <= 3%). Because results for averaged input data generally revealed higher values in the context of the here created MLP, average word vectors were employed at this point. The opposite situation held for the here created CNN, where results for simple word vectors achieved higher accuracies. For this reason, the simpler variant has been chosen in this context.
  Additionally, for the MLP no huge difference employing the aforementioned averaged word vectors and pre-trained word embeddings created with the help of Word2Vec could be found (about <= 4%). Because own constructed word vectors performed better, they were chosen in this context.
- **Notes concerning model settings:**
  It should be noted, that for all models several settings (including e.g. number of convolution maps, filter size, number of hidden layers, etc.) have been tested. For reasons of computational complexity and limits due to computing power of the available machine, exploration was limited to some extent.
  The here reported values represent the best results obtained under the given circumstances. Nevertheless, changing values for average accuracies and standard errors have to be considered due to changing subset contents in 10-fold cross validation splits.

## 2. Convolutional Neural Network (CNN)

Briefly explain your network architecture.

- The first layer is an Embedding Layer, where the averaged document vectors/ integer representation of the words of all documents are converted into word embeddings/ dense vectors. It takes as input an integer matrix of shape (2000, 2348), with 2000 documents and 2348 as the size of the longest document. Shorter documents are padded to this length. The vocabulary size is 5000 (pre-defined number of top most frequently occurring words in all documents), and vocabulary indices are mapped into 100 dimensions. Additionally, a fraction of 0.5 embeddings are dropped.
  → Output: (None, 2348, 100)
- The Convolution1D Layer learns 25 word-group-filters of length 5. It applies 25 convolution filters, so the output dimensionality will be 25 (i.e. 25 convolution maps) for each of the input vectors. *ReLu* activation function is applied to the output.
  → Output: (None, 2344, 50)
- In the subsequent step, a max Pooling Layer of length 2 is employed, dividing the input to this layer into half, looking at the maximum value within each convolution for each of the input vectors. *ReLu* activation function is applied to the output.
  → Output: (None, 1172, 50)
- The Flatten Layer flattens its input in order to add another Dense Layer in the next step.
  → Output: (None, 58600)
- The next two fully connected layers consequently reduce the output (first: *ReLu* activation), until the last layer returns one single output (last: *sigmoid* activation function).
  → Outputs: (None, 50); (None, 1)
- The created model is compiled and learning is configured. *Rmsprop* is used as optimizer, *binary_crossentropy* as loss function.
- *Ten-fold cross validation* is employed, so the created model is fit ten times on data and class-label subsets. Remaining data and class labels are, in a further step, used for testing and evaluation.

## 3. Discussion

Briefly discuss the results you have obtained. Include a comparison of each model for their accuracy as well as computational complexity.

- Surprisingly (for me), LG obtained better accuracies than both neural networks. Among the latter, MLP performed obviously worse than CNN.
  One reason for this behavior might be found in that LG directly maps to the fed input and thereby also covers bigrams. In comparison, some information gets lost for CNN due to built convolutions and subsequent local focus given max pooling. Information loss e specially holds for MLP, because the order in which words of a certain document appear is lost in a vector representation. CNN is able to achieve better results for the way word sequences (n-grams) are covered provided convolution filters of a specific length.
- The three classification tasks and the employed models seem to increase concerning computational complexity: LG < MLP < CNN.
  LG simply maps data onto a sigmoid function given minimal loss. For MLPs, the number of layers and their units increase the previously described complexity (increase compared to LG, because also employs sigmoid activation function, but additional complexity through multiple layer architecture). Especially hidden layers increase complexity, as every hidden

unit computes some value employing the value, which has been assigned from the input layer, and some activation function. The more layers there are, the more complex a model becomes. Additionally to these fully connected layers in the present MLP architecture (also can be found in a slightly varied form in the present CNN), multiple convolutions of a specific length are learned in the present CNN. In that way, layers are not fully connected, as for the MLP, but recognize local features with the help of shared weights. Additionally, (max) pooling helps focusing on reliable features. The inclusion of all these additional layers/steps reveal an again higher level of computational complexity.