Universität Zürich

Deep Learning in der Sprachtechnologie

Miniprojekt

Raphael Matile
12-711-222
raphael.matile@uzh.ch

Nicolas Diener
TODO
nicolas.diener@uzh.ch

12th January 2017

1 Task Description

Provide url to dataset

1.1 Dataset

1.2 Architecture Overview

Elman resp. Jordan Network

2 Evaluation

In the following, we shortly outline the results of our findings in two tables.

2.1 Jordan Network

For the detailed results, please see Table 1 with the detailed explanations.

2.2 Elman Network

For the results of the Elman Network, please see the Table 2 which includes comments about each change.

2.3 Achievend Results

Provide xsl sheet Provide commented program which achieved best results

Since we did not achieve a better result than the default configuration provided already in the repository [1], we link here to the default implementation used. For the Jordan network, please see `https://github.com/herrnici/is13/blob/master/examples/jordan-forward.py` respective `https://github.com/herrnici/is13/blob/master/examples/elman-keras.py` for the Elman Network.

[1] https://github.com/herrnici/is13/

Folds	Learning Rate	Decay	hidden Units	Epochs	Results (F1)	Comments
3	0.0627142536696559	false	100	50	96.45/93.65	Baseline without changes
3	0.0627142536696559	false	100	**25**	95.88/93.55	Goal: check whether we can reduce the time with similar results, having in mind the faster experimentation time for other parameters
3	0.0627142536696559	**1**	100	25	95.65/95.65	Introducing Conditional Decay (halving after 10 steps with no progress) incurs overfitting
3	0.0627142536696559	false	**200**	25	95.51/93.83	Larger hidden layers incur overfitting
4	0.0627142536696559	false	**100**	25	95.93/93.23	More folds incur overfitting

Table 1: Results for the Jordan network

Folds	Learning Rate	Hidden Units	Embedding Dimensions	Epochs	Results (F1)	Comments
3	0.1	100	100	50	95.41/92.54	Baseline without changes
3	0.1	100	100	**25**	94.01/91.58	Goal: check whether we can reduce the time with similar results, having in mind the faster experimentation time for other parameters
3	**0.001**	100	100	**250**	69.36/66.88	Verify, that lower learning rate requires more epochs in order to have a similar accuracy.
3	0.01	**200**	**200**	250	95.64/92.22	Since the learning rate was too low, we increase it again, but at the same time, we also increase the dimensionality for the embedding
3	**0.1**	200	200	250	96.44/92.56	Now, using factor 5 for epochs regarding to the beginning, we see, that no big increase was made, therefore, the minimum of the cost function was already good approximated before. Also using factor 2 for the hidden units did not change the result significantly.
4	0.1	**300**	200	**50**	95.67/92.4	Just increasing the hidden units compared to the initial configuration, does not really help: going broad without going deep could be an issue here
3	0.1	**200**	200	**250**	94.29/90.24	By using a decay, the training score already decreases, generalization is too intense, also visible in the test score. Also, using a different activation function than sigmoid (i.e. ReLu) does not improve the solution. Furthermore, we used a decay of 0.0001 which did not yield any improvement.
3	0.1	200	200	250	94.28/89.83	By using the same decay again, but the Sigmoid-Activation function and a second Layer of the SimpleRNN provided by Keras, we still can not reach an improvement but rather a deterioration in the result.
3	0.1	200	200	250	95.56/91.3	However, using sigmoid as activation function for the first and ReLu for the second layer in combination with no decay at all, we result again in a quite improvement with regards to the test set.
3	0.1	200	200	250	94.94/90.5	Interestingly enough, using a third SimpleRNN, the score performance reduces again, not only in the training but also in the evaluation set.

Table 2: Results for the Elman network