Deep Learning for Natural Language Processing - Home Work 1
Group 150

Maximilian Kaiser, 2805164
Sewin Tariverdian, 2903040

SoSe 2020/2021 Date:April 29, 2021

Task 1: Setup

Task 2: Sigmoid Activation Function

Derivation

$sig(x) = \frac{1}{1+exp(-x)}$
To show: $sig'(x) = sig(x) * (1 - sig(x))$

Make first derivation of sig(x):

$$
\begin{aligned}
sig'(x) &= -(1 + exp(-x))^2 * exp(-x) * (-1) \\
&= \frac{exp(-x)}{(1 + exp(-x))^2} \\
&= \frac{exp(-x)}{(1 + exp(-x))^2} + \underbrace{\frac{1}{(1 + exp(-x))^2} - \frac{1}{(1 + exp(-x))^2}}_{0} \\
&= \frac{1 + exp(-x)}{(1 + exp(-x))^2} - \frac{1}{(1 + exp(-x))^2} \\
&= \frac{1}{1 + exp(-x)} - \frac{1}{(1 + exp(-x))^2} \\
&= sig(x) - sig^2(x) \\
&= sig(x) * (1 - sig(-x))
\end{aligned}
$$

Task 3: TensorFlow Playground

Circular Dataset
No, a single layer perceptron can create a discriminator, which has a dimension less then the given dataset. In this case only a line, which cant separate the blue from the orange points.
MLP Scenarios:

1. Same amount of neurons in every hidden layer.
2. More neurons towards the input, less neurons towards the output.
3. Less neurons towards the input, more neurons towards the output.

The second scenario was performing with the lowest loss value and converges the fastest. We think this is due to abstraction at the beginning and concretization at the end.

Task 4: Softmax

By taking all data points into account the softmax-function outputs a vector which values are related to each other and sum up to 1, where the sigmoid-function does not suffice this criteria.

Task 5: Sentiment Polarity in Movie Reviews

Reader
The given text files are being read through the first part of the *P5.py*-script, which iterates through the set and saves it as 101-dimensional input and 1-dimensional output vectors.

Numpy Implementation
The perceptron has been implemented as a class, which includes the global variables for hyperparameters/data and also functions for calculating the loss-/accuracy values. In order to find a meaningful way to represent the prediction, we decided to round the predicted output value. For values over 0.5 it predicts 1 and under 0.5 it predicts 0.

Training
We played around with the hyperparameters until we got a accuracy over 70%. The result in 1 was accomplished with this following setting: epochs =200, tau =7, alpha = 0.15

```
root@b1119bb011b4:/src# python P5.py
Done reading
Start Training with epochs 200 , tau size = 7 and Learning rate = 0.15
P5.py:38: RuntimeWarning: overflow encountered in exp
  return 1 / (1 + np.exp(-x))
Finished with 200  epochs
Loss on dev-set:  [464.80512818]
Accuracy on dev-set:  0.7091932457786116
Loss on Test-set:  [462.60972508]
Accuracy on Test-set:  0.7104440275171983
root@b1119bb011b4:/src#
```

Figure 1: Results after learning in Docker

In a different environment we tried to compute a graphical result of the loss function, to get a better understanding of our system and if it is actually learning. We were satisfied with the trend, but acknowledge that there is some space for fine-tuning.

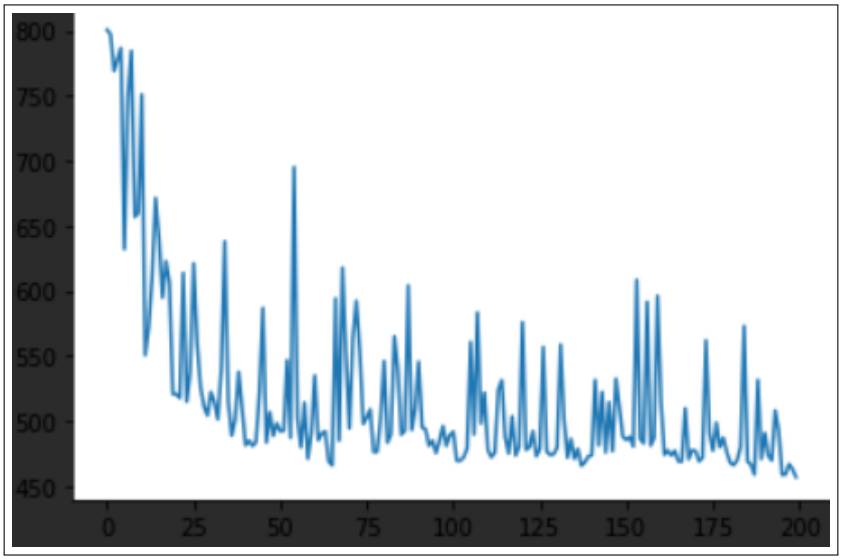

Figure 2: Graphical result of Loss function in respect to epochs-count