Machine Learning Worksheet 7

Tomas Ladek, Michael Kratzer
3602673, 3612903
tom.ladek@tum.de, mkratzer@mytum.de

Problem 1

Let us rewrite the sigmoid activation function as

$$\sigma(x) = \frac{1}{1+e^{-x}} = \frac{e^x}{e^x(1+e^{-x})} = \frac{e^x}{e^x+e^{x-x}} = \frac{e^x}{e^x+1}$$

A neural network with the hyperbolic tangent function $\tanh(x)$ as activation function is equivalent to one with activation function $\sigma(x)$, because the output of the former can be emulated through weights by scaling and offsetting the in- and output of the latter, since

$$\begin{aligned}
\tanh(x) &= \frac{e^x - e^{-x}}{e^x + e^{-x}} \\
&= \frac{e^{-x}(e^{2x}-1)}{e^{-x}(e^{2x}+1)} \\
&= \frac{e^{2x}-1}{e^{2x}+1} \\
&= \frac{2e^{2x}-(1+e^{2x})}{e^{2x}+1} \\
&= \frac{2e^{2x}}{e^{2x}+1} - \frac{1+e^{2x}}{e^{2x}+1} \\
&= 2\frac{e^{2x}}{e^{2x}+1} - 1 \\
&= 2\sigma(2x) - 1
\end{aligned}$$

And therefore

$$\begin{aligned}
\tanh(x) &= 2\sigma(2x) - 1 \\
\tanh(x) + 1 &= 2\sigma(2x) \\
\frac{1}{2}(\tanh(x)+1) &= \sigma(2x) \\
\frac{1}{2}(\tanh(\frac{z}{2})+1) &= \sigma(z) \qquad \textit{with } z = 2x
\end{aligned}$$

Problem 2

$$\begin{aligned}\frac{d}{dx}\ \sigma(x) = \frac{d}{dx}\ \frac{e^x}{e^x+1} &= \frac{(e^x+1)e^x - e^x e^x}{(e^x+1)^2} \\ &= \frac{(e^x+1)e^x}{(e^x+1)^2} - \frac{(e^x)^2}{(e^x+1)^2} \\ &= \frac{e^x}{e^x+1} - \Big(\frac{e^x}{e^x+1}\Big)^2 \\ &= \sigma(x) - \sigma^2(x)\end{aligned}$$

$$\begin{aligned}\frac{d}{dx}\ \tanh(x) &= \frac{d}{dx}\ \frac{e^x - e^{-x}}{e^x + e^{-x}} = \frac{(e^x+e^{-x})(e^x+e^{-x}) - (e^x-e^{-x})(e^x-e^{-x})}{(e^x+e^{-x})^2} \\ &= \frac{(e^x+e^{-x})^2 - (e^x-e^{-x})^2}{(e^x+e^{-x})^2} = 1 - \Big(\frac{e^x-e^{-x}}{e^x+e^{-x}}\Big)^2 = 1 - \tanh^2(x)\end{aligned}$$

Problem 3

#trivial

Problem 4

With the Laplacian distribution given as

$$p(\mathbf{z}|\mathbf{x},\mathbf{w},\beta) = \prod_{n=1}^{N} p(z_n|x_n,\mathbf{w},\beta) = \prod_{n=1}^{N} \frac{1}{2\beta} exp(-\frac{|z_n - y(x_n,\mathbf{w})|}{\beta}) \tag{1}$$

Then the Log-Likelihood is

$$\begin{aligned}-ln(\prod_{n=1}^{N} p(z_n|x_n,\mathbf{w},\beta)) &= -\sum_{n=1}^{N} ln(p(z_n|x_n,\mathbf{w},\beta)) &(2)\\ &= -\sum_{n=1}^{N} ln(\frac{1}{2\beta} exp(-\frac{|z_n - y(x_n,\mathbf{w})|}{\beta})) &(3)\\ &= -\sum_{n=1}^{N} ln(\frac{1}{2\beta}) - \frac{|z_n - y(x_n,\mathbf{w})|}{\beta} &(4)\\ &= -N ln(\frac{1}{2\beta}) + \frac{1}{\beta}\sum_{n=1}^{N} |z_n - y(x_n,\mathbf{w})| &(5)\end{aligned}$$

Scaling likelihood by a constant only changes height, not location with respect to w. So we can set $\beta to 1$ and get constant plus the summed absolute errors.

$$-N ln(\frac{1}{2}) + \sum_{n=1}^{N} |z_n - y(x_n,\mathbf{w})| \tag{6}$$

So deriving this term with respect to w is the same as deriving the summed absolute errors.

Problem 5

Below six plots of the training curves for different learning rates used in `mlp_xor.NeuralNetwork(X,y,l)` and `mlp_sin.NeuralNetwork(X,y,l)` respectively.

mlp_xor.NeuralNetwork(X,y,l):

```
output:
[[  4.56613223e-05]
 [  9.97041292e-01]
 [  9.97041213e-01]
 [  6.24880067e-05]]
```

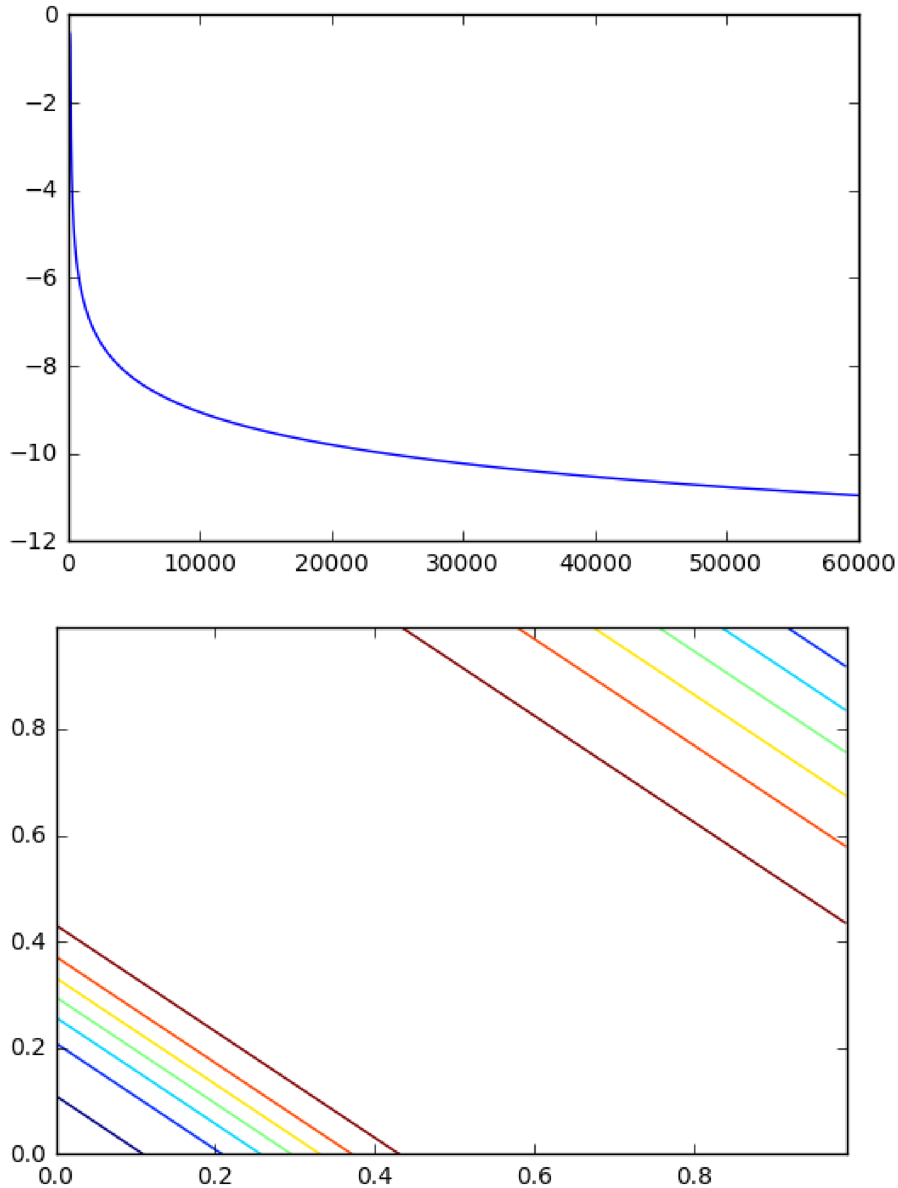
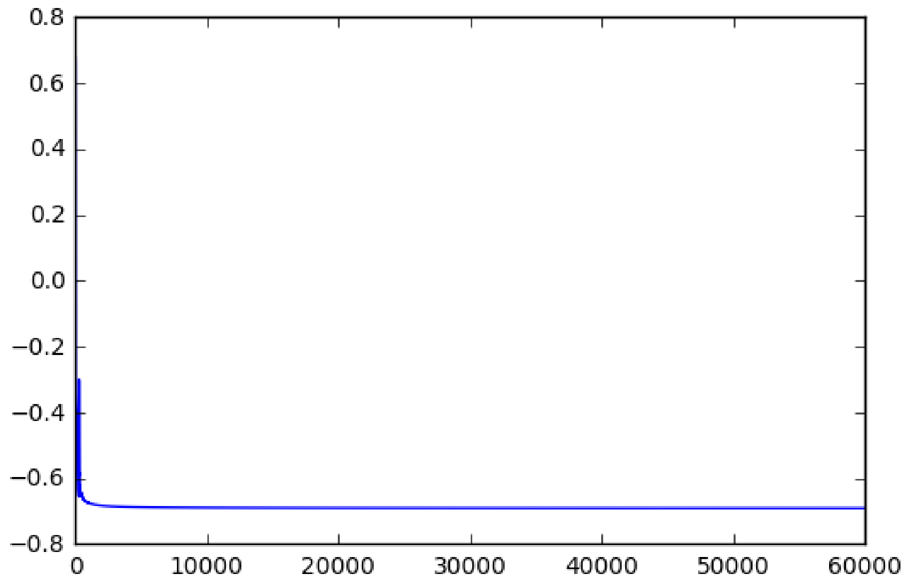


Figure 1: Function: XOR, Learning rate 0.3

```
output:
[[ 0.03188002]
 [ 0.9858056 ]
 [ 0.50156382]
 [ 0.50160351]]
```



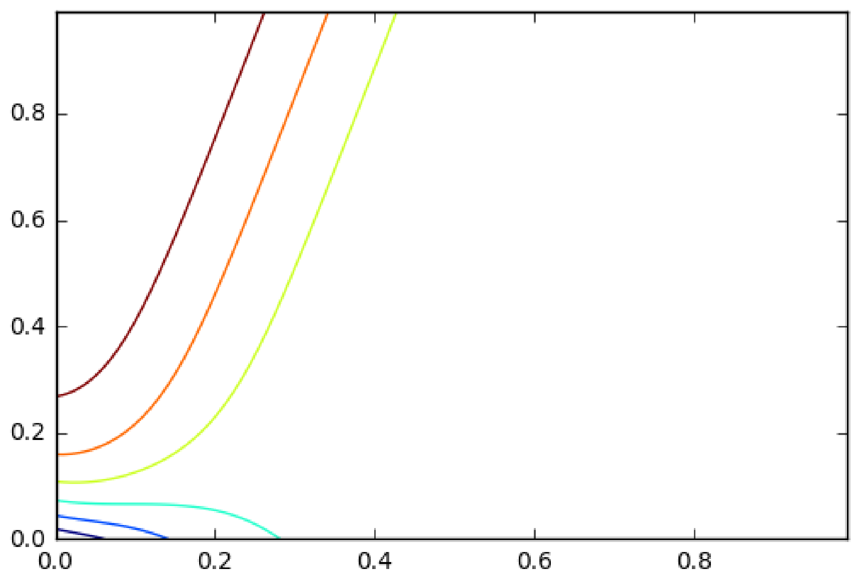


Figure 2: Function: XOR, Learning rate 0.6

```
output:
[[  8.46115472e-05]
 [  4.99961733e-01]
 [  9.90803009e-01]
 [  5.00350863e-01]]
```

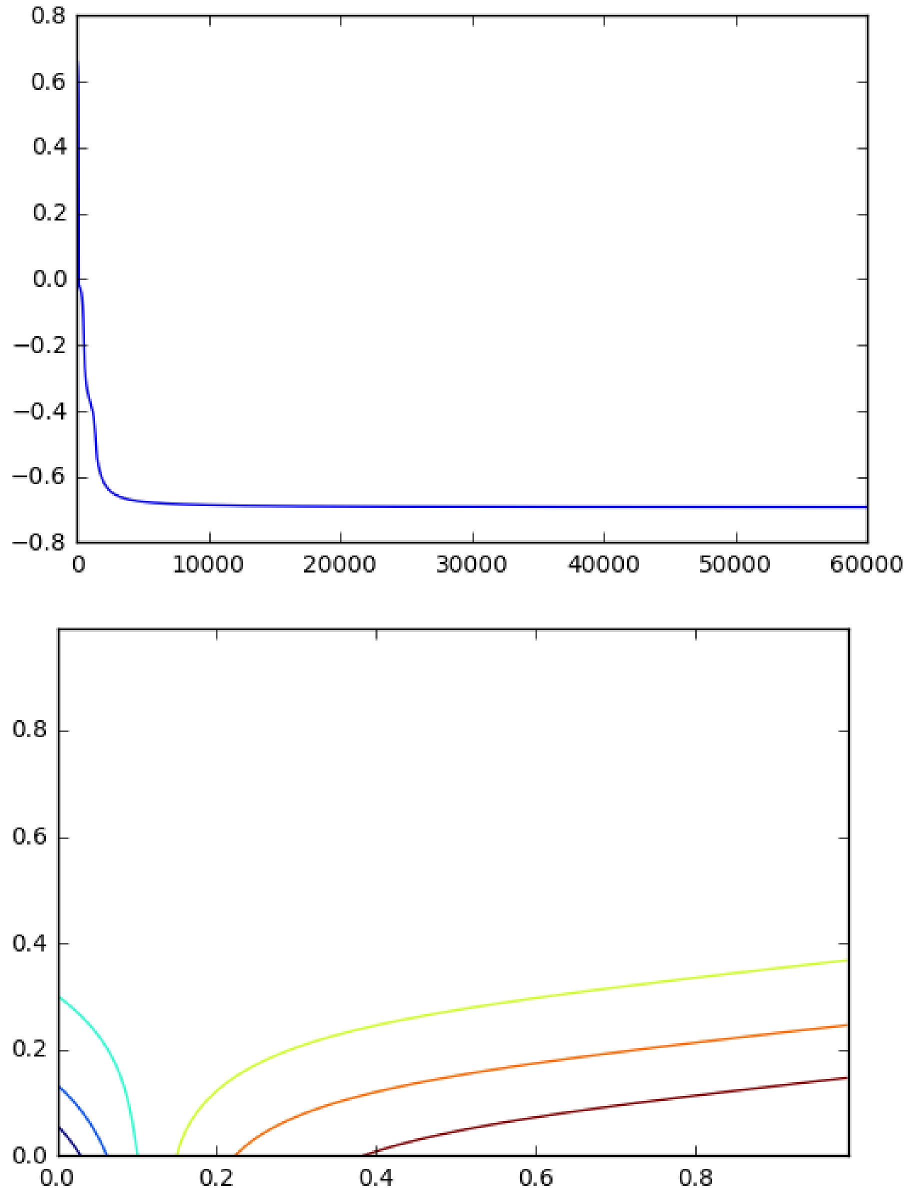


Figure 3: Function: XOR, Learning rate 0.05

mlp_sin.NeuralNetwork(X,y,l):

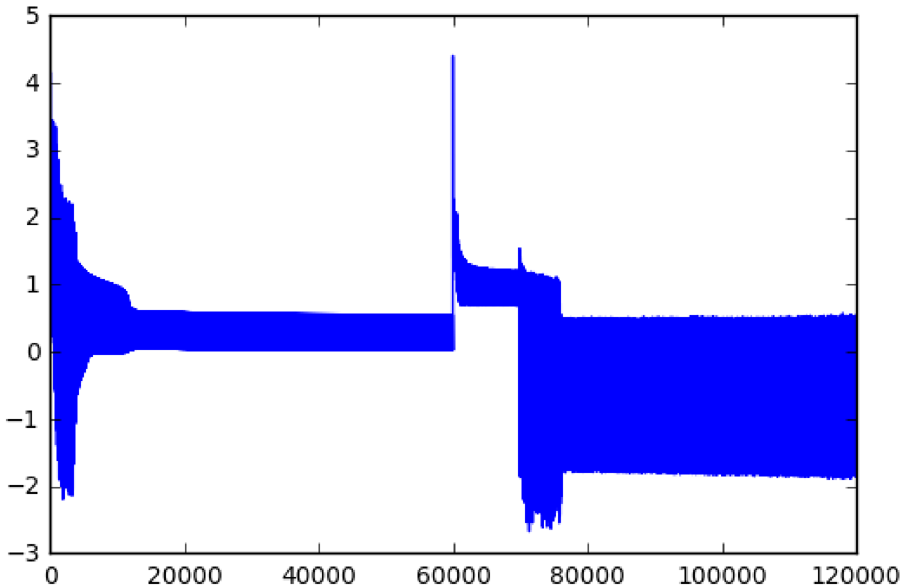


Figure 4: Function: Sin, Learning rate 0.3

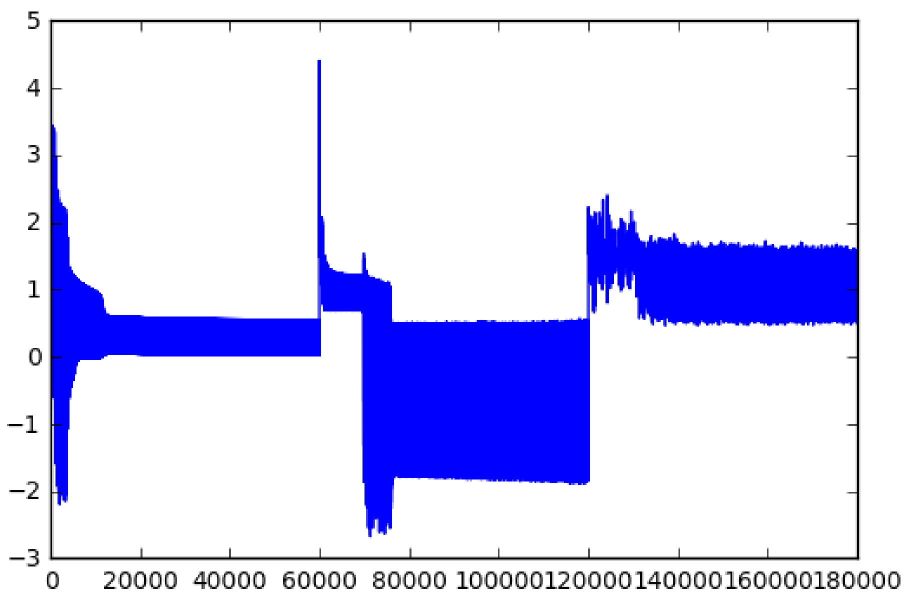


Figure 5: Function: Sin, Learning rate 0.7

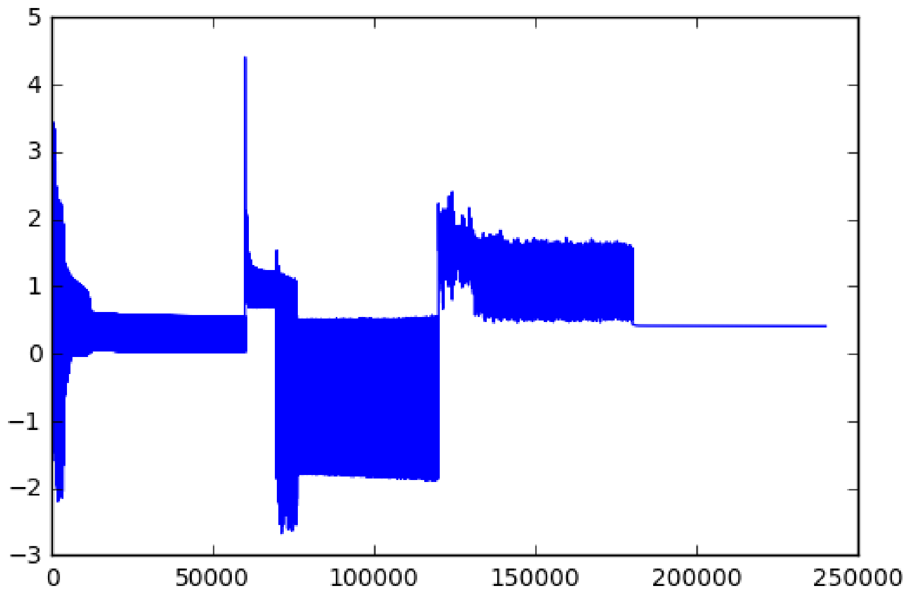


Figure 6: Function: Sin, Learning rate 0.05

Problem 6

The last input is always one because it represents the bias.

Problem 7

The implementation models the XOR problem in 2D space. So our decision boundary is a line in 2D space, which makes it impossible to seperate the 2 classes.