Deep learning Assignment 1

Raymond Koopmanschap - 11925582
raymond.koopmanschap@student.uva.nl

1 MLP backprop and NumPy implementation

Throughout this assignment the convention is used that $\frac{\partial a}{\partial \vec{x}} \in \mathbb{R}^{1\times n}$

Question 1.1 a)

$$\frac{\partial L}{\partial x_i^{(N)}} = -\frac{\partial \sum_i t_i log x_i^{(N)}}{\partial x_i^{(N)}} = -\frac{t_i}{x_i^{(N)}}$$

if i=j: $$\frac{\partial x_i^{(N)}}{\partial \tilde{x}_j^{(N)}} = \frac{e^{\tilde{x_i}^{(N)}} \sum_{i=1}^{d_N} e^{(\tilde{x}_i^{(N)})} - e^{\tilde{x_i}^{(N)}} e^{\tilde{x_j}^{(N)}}}{(\sum_{i=1}^{d_N} e^{(\tilde{x}_i^{(N)})})^2} = \frac{e^{\tilde{x_i}^{(N)}}}{\sum_{i=1}^{d_N} e^{(\tilde{x}_i^{(N)})}}(\frac{\sum_{i=1}^{d_N} e^{(\tilde{x}_i^{(N)})}}{\sum_{i=1}^{d_N} e^{(\tilde{x}_i^{(N)})}} - \frac{e^{\tilde{x_j}^{(N)}}}{\sum_{i=1}^{d_N} e^{(\tilde{x}_i^{(N)})}})$$
$$= x_i^{(N)}(1 - x_j^{(N)})$$

if i $\neq$ j: $$\frac{\partial x_i^{(N)}}{\partial \tilde{x}_j^{(N)}} = -\frac{e^{\tilde{x_i}^{(N)}} e^{\tilde{x_j}^{(N)}}}{(\sum_{i=1}^{d_N} e^{(\tilde{x}_i^{(N)})})^2} = -\frac{e^{\tilde{x_i}^{(N)}}}{\sum_{i=1}^{d_N} e^{(\tilde{x}_i^{(N)})}} \frac{e^{\tilde{x_j}^{(N)}}}{\sum_{i=1}^{d_N} e^{(\tilde{x}_i^{(N)})}} = x_i^{(N)} x_j^{(N)}$$

Combined: $$\frac{\partial x_i^{(N)}}{\partial \tilde{x}_j^{(N)}} = x_i^{(N)}(\mathbb{1}_{i=j} - x_j^{(N)})$$

$$\frac{\partial x_i^{(l<N)}}{\partial \tilde{x}_j^{(l<N)}} = \mathbb{1}_{(i=j)\wedge \tilde{x}_i > 0}$$

$$\frac{\partial \tilde{x}_i^{(l)}}{\partial x_j^{(l-1)}} = W_{ij}$$

$$\frac{\partial \tilde{x}_i^{(l)}}{\partial W_{jk}^{(l)}} = x_k^{(l-1)} \mathbb{1}_{i=j}$$

$$\frac{\partial \tilde{x}_i^{(l)}}{\partial b_j^{(l)}} = \mathbb{1}_{i=j}$$

Question 1.1 b)

$$\begin{aligned}
\frac{\partial L}{\partial \tilde{x}^{(N)}} &= \frac{\partial L}{\partial x^{(N)}}\frac{\partial x^{(N)}}{\partial \tilde{x}^{(N)}} = \sum_i \frac{\partial L}{\partial x_i^{(N)}}\frac{\partial x_i^{(N)}}{\partial \tilde{x}^{(N)}} \\
&= \sum_i -\frac{t_i}{x_i^{(N)}}[x_i^{(N)}(\mathbb{1}_{i=j} - x_1^{(N)}) \cdots x_i^{(N)}(\mathbb{1}_{i=j} - x_{d_N}^{(N)})] \text{ (Where j denotes the j-th element of } \tilde{x}^{(N)}\text{)} \\
&= \sum_i -t_i[\mathbb{1}_{i=j} - x_1^{(N)} \cdots \;_{i=j} - x_{d_N}^{(N)}] \\
&= [x_1^{(N)} - \mathbb{1}_{argmax(t)=j} \cdots \mathbb{1}_{argmax(t)=j} - x_{d_N}^{(N)}] \text{ (Because of t being one-hot)} \\
&= (x^{(N)})^T - t^T \\
\frac{\partial L}{\partial \tilde{x}_i^{(l<N)}} &= \sum_j \frac{\partial L}{\partial x_j^{(l)}}\frac{\partial x_i^{(l)}}{\partial \tilde{x}^{(l)}} = \sum_j \frac{\partial L}{\partial x_i^{(l)}}\mathbb{1}_{(j=i)\wedge \tilde{x}_j>0} = \frac{\partial L}{\partial x_j^{l}}\mathbb{1}_{\tilde{x}_i>0} \text{ thus } \frac{\partial L}{\partial \tilde{x}^{(l<n)}} = \frac{\partial L}{\partial x^{(l)}} \odot \mathbb{1}_{\tilde{x}>0} \\
\frac{\partial L}{\partial x^{(l<N)}} &= \frac{\partial L}{\partial \tilde{x}^{(l+1)}}\frac{\partial \tilde{x}^{(l+1)}}{\partial x^{(l)}} = \frac{\partial L}{\partial \tilde{x}^{(l+1)}}W^{(l+1)} \\
\frac{\partial L}{\partial W_{ij}^{(l)}} &= \sum_k \frac{\partial L}{\partial \tilde{x}_k^{(l)}}\frac{\partial \tilde{x}_k^{(l)}}{\partial W_{ij}^{(l)}} = \sum_k \frac{\partial L}{\partial \tilde{x}_k^{(l)}}x_j^{(l-1)}\mathbb{1}_{k=i} = \frac{\partial L}{\partial \tilde{x}_i^{(l)}}x_j^{(l-1)} \text{ thus } \frac{\partial L}{\partial W^{(l)}} = (x^{(l-1)}\frac{\partial L}{\partial \tilde{x}^{(l)}})^T \\
\frac{\partial L}{\partial b^{(l)}} &= \frac{\partial L}{\partial \tilde{x}^{(l)}}\frac{\partial \tilde{x}^{(l)}}{\partial b^{(l)}} = \frac{\partial L}{\partial \tilde{x}^{(l)}}
\end{aligned}$$

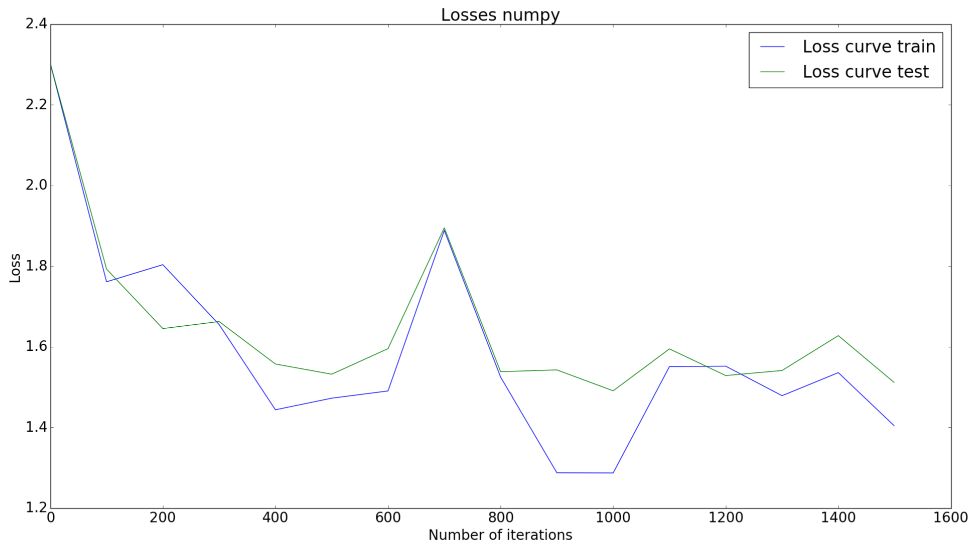
Question 1.1 c)

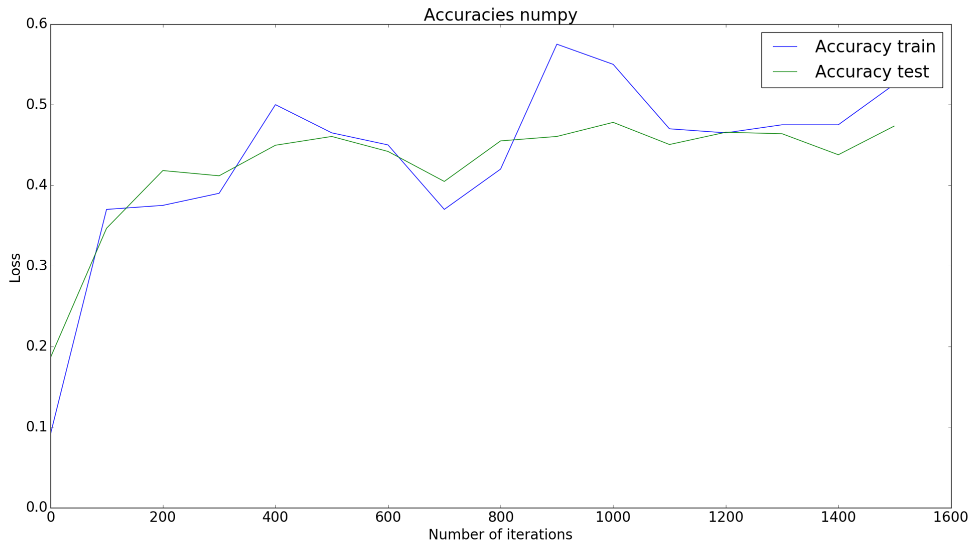
The change for each weight is the average over the loss for each example in a batch. Let the batch size be B and let $x_s^{(l)}$ indicate the vector x of example s in layer l. Then an example for one of the updates will be the equation below and the principle will be the same for each weight update.

$$\frac{\partial L}{\partial W^{(l)}} = \frac{1}{B}\sum_{s=1}^{B} x_s^{(l)}\frac{\partial L}{\partial \tilde{x}_s^{(l+1)}}$$

Question 1.2

Below are the loss curves and accuracy curves for the default parameter setting. The final test accuracy with the default parameters was 0.4733.





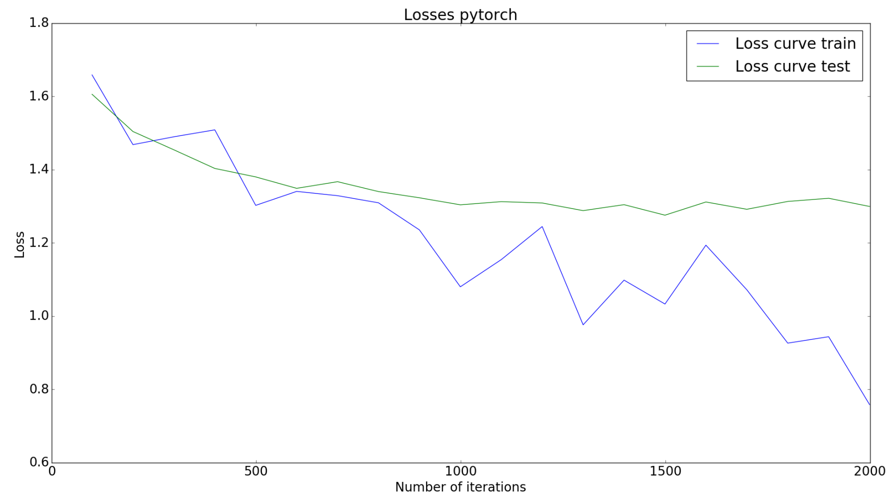
PyTorch MLP

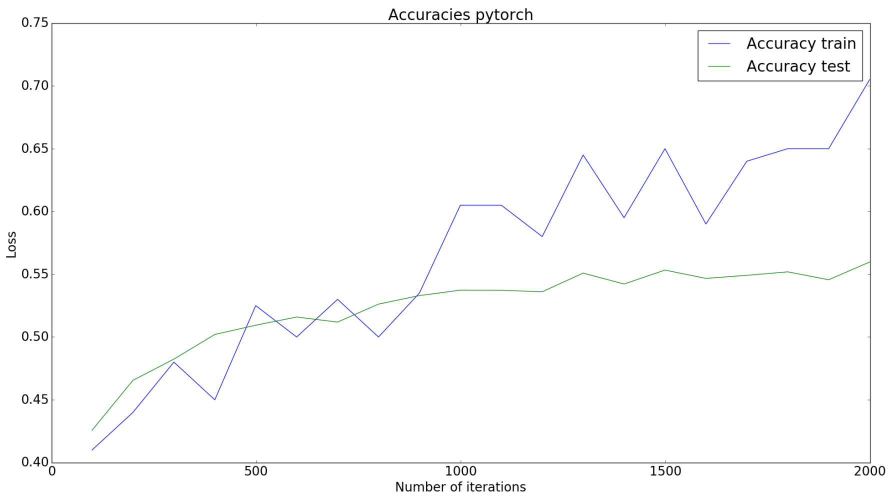
Tried modifications

- First different optimizers are tried. Adams works best overall, since in the literature this also seems to work best most often, I stick with Adam in future experiments
- Second modification is batch normalization before and after the ReLU activation function. After the activation gave overall better results, so also that will be used in the next couple of experiments. At this point the accuracy was 52.01, so just above the 52.
- Third modification was more iterations, since the loss curve showed still a downward trend and didn't stagnate yet. After 2500 iterations the accuracy was 52.45. After 3500 iterations the accuracy went down to 51.9 and the loss of the test set was going up while the loss on the training set was going down (overfitting). Therefore 2500 iterations seems best for now.
- Next modification is increasing the complexity of the network by adding an extra hidden layer, this will give the network more opportunity to learn complex functions. There is now a hidden layer of 200 units and one of 100 units. For 1500, 2000 respectively 2500 iterations it gave, 54.7, 55.29, 54.51 as accuracy. Because this seems to improve accuracy I have added another extra layer before the other layers with 100 units more then the previous one, until accuracy was not improved anymore. With [300, 200, 100] it was 55.62. With [400, 300, 200, 100], 55.97. After that accuracy didn't seems to really improve and the network was overfitting again.
- Adding weight decay didn't improve the accuracy.
- Increasing the learning rate 5e-3 or decreasing the learning rate 1e-3, 5e-4, didn't seem to increase performance.

The loss and accuracy of the train and test set of the best model are plotted below. The parameters for the best model are set as default parameters:

- Adam optimizer
- Batch normalization after ReLU activation function.
- Number of iterations: 2000
- Learning rate 2e-3
- Hidden layers: [400, 300, 200, 100]
- No weight decay


Losses pytorch
Loss curve train
Loss curve test
Loss
Number of iterations
0.6
0.8
1.0
1.2
1.4
1.6
1.8
0
500
1000
1500
2000


Accuracies pytorch
Accuracy train
Accuracy test
Loss
Number of iterations
0.40
0.45
0.50
0.55
0.60
0.65
0.70
0.75
0
500
1000
1500
2000

Custom module: Batch Normalization

Automatic differentation

See code

Manual implementation of backward pass

Question 3.2 a)

There is assumed that $\frac{\partial L}{\partial y}$ is given.

$$\begin{aligned}\frac{\partial L}{\partial \gamma_j} &= \sum_s \sum_i \frac{\partial L}{\partial y_i^s} \frac{\partial y_i^s}{\partial \gamma_j} = \sum_s \sum_i \frac{\partial L}{\partial y_i^s} \mathbb{1}_{i=j} \hat{x}_i^s = \sum_s \frac{\partial L}{\partial y_j^s} \hat{x}_j^s = \frac{\partial L}{\partial y_j} \hat{x}_j \\ \frac{\partial L}{\partial \gamma} &= \sum_s \frac{\partial L}{\partial y^s} \odot \hat{x}^s\end{aligned}$$

$$\begin{aligned}\frac{\partial L}{\partial \beta_j} &= \sum_s \sum_i \frac{\partial L}{\partial y_i^s} \frac{\partial y_i^s}{\partial \beta_j} = \sum_s \sum_i \frac{\partial L}{\partial y_i^s} \mathbb{1}_{i=j} = \sum_s \frac{\partial L}{\partial y_j^s} \\ \frac{\partial L}{\partial \beta} &= \sum_s \frac{\partial L}{\partial y^s}\end{aligned}$$

The last one will be presented in pieces due to the length of the derivation. First we will write it out to make it a bit simpler.

$$\begin{aligned}\frac{\partial L}{\partial x_j^r} &= \sum_s \sum_i \frac{\partial L}{\partial y_i^s} \frac{\partial y_i^s}{\partial x_j^r} = \sum_s \sum_i \frac{\partial L}{\partial y_i^s} \frac{\partial y_i^s}{\partial \hat{x}_i^s} \frac{\partial \hat{x}_i^s}{\partial x_j^r} = \sum_s \sum_i \frac{\partial L}{\partial y_i^s} \mathbb{1}_{i=j} \gamma_i \frac{\partial \hat{x}_i^s}{\partial x_j^r} \\ &= \sum_s \frac{\partial L}{\partial y_j^s} \gamma_j \frac{\partial \hat{x}_j^s}{\partial x_j^r}\end{aligned}$$

So next we have to calculate:

$$\frac{\partial \hat{x}_j^s}{\partial x_j^r} = \frac{\partial}{\partial x_j^r} \frac{x_j^s - \mu_j}{\sqrt{\sigma_j^2 + \epsilon}}$$

Since μ_j and σ_j are by themselves formulas with x_j^r in them, we will calculate them separately and then combine everyting.

$$\frac{\partial \mu_j}{\partial x_j^r} = \frac{1}{B}$$

$$\begin{aligned}\frac{\partial \sigma_j}{\partial x_j^r} &= \frac{\partial}{\partial x_j^r} (\frac{1}{B} \sum_s (x_j^s - \mu_j)^2) \\ &= \frac{1}{B} \sum_s 2(x_j^s - \mu_j)(\frac{\partial}{\partial x_j^r}(x_j^s - \mu_j)) \\ &= \frac{1}{B} \sum_s 2(x_j^s - \mu_j)(\mathbb{1}_{r=s} - \frac{1}{B}) \\ &= \frac{2}{B}(\sum_s (x_j^s - \mu_j)\mathbb{1}_{r=s} - \frac{1}{B} \sum_s (x_j^s - \mu_j)) \\ &= \frac{2(x_j^r - \mu_j)}{B}\end{aligned}$$

For the last step we used $\mu_i = \frac{1}{B}\sum_s x_j^s$

$$\frac{1}{B}\sum_s (x_j^s - \mu_j) = 0$$
$$\frac{1}{B}\sum_s x_j^s - \frac{1}{B}\sum_s \mu_j = 0$$
$$\frac{1}{B}\sum_s x_j^s - \mu_j = 0$$

Now we can combine everything

$$\begin{aligned}
\frac{\partial \hat{x}_j^s}{\partial x_j^r} &= \frac{\sqrt{\sigma_j^2+\epsilon}(\mathbb{1}_{r=s} - \frac{1}{B}) - (x_j^s - \mu_j)\frac{1}{2\sqrt{\sigma_j^2+\epsilon)}}\frac{2(x_j^r-\mu_j)}{B}}{\sigma_j^2+\epsilon} \\
&= \frac{\sqrt{\sigma_j^2+\epsilon}(\mathbb{1}_{r=s} - \frac{1}{B}) - \frac{(x_j^s-\mu_j)(x_j^r-\mu_j)}{B\sqrt{\sigma_j^2+\epsilon}}}{\sigma_j^2+\epsilon} \\
&= \frac{\mathbb{1}_{r=s} - \frac{1}{B}}{\sqrt{\sigma_j^2+\epsilon}} - \frac{(x_j^s-\mu_j)(x_j^r-\mu_j)}{B(\sigma_j^2+\epsilon)\sqrt{\sigma_j^2+\epsilon}} \\
&= \frac{B\mathbb{1}_{r=s} - 1}{B(\sigma_j^2+\epsilon)\sqrt{\sigma_j^2+\epsilon}} - \frac{(x_j^s-\mu_j)(x_j^r-\mu_j)}{B(\sigma_j^2+\epsilon)\sqrt{\sigma_j^2+\epsilon}} \\
&= \frac{B\mathbb{1}_{r=s} - 1 - (x_j^s-\mu_j)(x_j^r-\mu_j)}{B(\sigma_j^2+\epsilon)\sqrt{\sigma_j^2+\epsilon}}
\end{aligned}$$

Now we can fill this in the final equation and obtain

$$\begin{aligned}
\frac{\partial L}{\partial x_j^r} &= \sum_s \frac{\partial L}{\partial y_j^s}\gamma_j \frac{\partial \hat{x}_j^s}{\partial x_j^r} \\
&= \sum_s \frac{\partial L}{\partial y_j^s}\gamma_j \frac{B\mathbb{1}_{r=s} - 1 - (x_j^s-\mu_j)(x_j^r-\mu_j)}{B(\sigma_j^2+\epsilon)\sqrt{\sigma_j^2+\epsilon}}
\end{aligned}$$

Question 3.2 b)

See code

Question 3.2 c)

See code

Pytorch CNN

See code