Deep Learning - Assignment 1

Jasper Adegeest
11650273
University of Amsterdam
`jasper.adegeest@student.uva.nl`

1 Analytical derivation of gradients

1.1 Question 1.1

1. Derivative of the softmax function:

$$x_i^{(N)} = \frac{e^{\tilde{x}_i^{(N)}}}{\sum_{k=1}^{d_N} e^{\tilde{x}_k^{(N)}}}$$

Calculate derivative for $i = j$:

$$\begin{aligned}
\frac{\partial x_{i=j}^{(N)}}{\partial \tilde{x}_j^{(N)}} &= \frac{e^{\tilde{x}_{i=j}^{(N)}} \sum_{k=1}^{d_N} e^{\tilde{x}_k^{(N)}} - e^{\tilde{x}_{i=j}^{(N)}} e^{\tilde{x}_i^{(N)}}}{(\sum_{k=1}^{d_N} e^{\tilde{x}_k^{(N)}})^2} \\
&= \frac{e^{\tilde{x}_{i=j}^{(N)}} (\sum_{k=1}^{d_N} e^{\tilde{x}_k^{(N)}} - e^{\tilde{x}_i^{(N)}})}{(\sum_{k=1}^{d_N} e^{\tilde{x}_k^{(N)}})^2} \\
&= \frac{e^{\tilde{x}_{i=j}^{(N)}}}{\sum_{k=1}^{d_N} e^{\tilde{x}_k^{(N)}}} \frac{\sum_{k=1}^{d_N} e^{\tilde{x}_k^{(N)}} - e^{\tilde{x}_i^{(N)}}}{\sum_{k=1}^{d_N} e^{\tilde{x}_k^{(N)}}} \\
&= x_{i=j}^{(N)} \frac{\sum_{k=1}^{d_N} e^{\tilde{x}_k^{(N)}} - e^{\tilde{x}_i^{(N)}}}{\sum_{k=1}^{d_N} e^{\tilde{x}_k^{(N)}}} \\
&= x_{i=j}^{(N)} (1 - \frac{e^{\tilde{x}_i^{(N)}}}{\sum_{k=1}^{d_N} e^{\tilde{x}_k^{(N)}}}) \\
&= x_{i=j}^{(N)} (1 - x_i^{(N)})
\end{aligned}$$

Calculate derivative for $i \neq j$:

$$\begin{aligned}
\frac{\partial x_{i\neq j}^{(N)}}{\partial \tilde{x}_j^{(N)}} &= \frac{-e^{\tilde{x}_{i\neq j}^{(N)}} e^{\tilde{x}_j^{(N)}}}{(\sum_{k=1}^{d_N} e^{\tilde{x}_k^{(N)}})^2} \\
&= \frac{-e^{\tilde{x}_{i\neq j}^{(N)}}}{\sum_{k=1}^{d_N} e^{\tilde{x}_k^{(N)}}} \frac{e^{\tilde{x}_j^{(N)}}}{\sum_{k=1}^{d_N} e^{\tilde{x}_k^{(N)}}} \\
&= -x_{i\neq j}^{(N)} x_j^{(N)}
\end{aligned}$$

2. Derivative of the cross entropy function:

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

$$\begin{aligned}
\frac{\partial L}{\partial x_i^{(N)}} &= -\sum_k t_k \frac{\partial \log x_k^{(N)}}{\partial x_i^{(N)}} \\
&= -\sum_k t_k \frac{\partial \log x_k^{(N)}}{\partial x_k^{(N)}} \frac{\partial x_k^{(N)}}{\partial x_i^{(N)}} \\
&= -\sum_k \frac{t_k}{x_k^{(N)}} \frac{\partial x_k^{(N)}}{\partial x_i^{(N)}} \\
&= -\frac{t_i}{x_i^{(N)}}
\end{aligned}$$

3. Derivative of the ReLU

$$x_i^{(l)} = \max(0, \tilde{x}_i^{(l)})$$

If $\tilde{x}_i^{(l)} < 0$*:*

$$\frac{\partial x_i^{(l<N)}}{\partial \tilde{x}_i^{(l<N)}} = 0$$

If $\tilde{x}_i^{(l)} > 0$*:*

$$\frac{\partial x_i^{(l<N)}}{\partial \tilde{x}_i^{(l<N)}} = 1$$

If $\tilde{x}_i^{(l)} = 0$ *we define the derivative to be 0, but in reality the derivative will be undefined.*

4. Derivative of the linear module with respect to the parameters

$$\tilde{x}^{(l)} = W^{(l)} x^{(l-1)} + b^{(l)}$$

$$\begin{aligned}
\frac{\partial \tilde{x}_i^{(l)}}{\partial x_j^{(l-1)}} &= \frac{\partial}{\partial x_j^{(l-1)}} \sum_k \left(W_{ik}^{(l)} x_k^{(l-1)}\right) + b_i^{(l)} \\
&= W_{ij}
\end{aligned}$$

5. Derivative of the linear module with respect to the weights

Calculate derivative for $i = j$*:*

$$\begin{aligned}
\frac{\partial \tilde{x}_{i=j}^{(l)}}{\partial W_{jk}^{(l)}} &= \frac{\partial}{\partial W_{jk}^{(l)}} \sum_m \left(W_{i=j,m}^{(l)} x_m^{(l-1)}\right) + b_{i=j}^{(l)} \\
&= x_k^{(l-1)}
\end{aligned}$$

Calculate derivative for $i \neq j$*:*

$$\begin{aligned}
\frac{\partial \tilde{x}_{i=j}^{(l)}}{\partial W_{jk}^{(l)}} &= \frac{\partial}{\partial W_{jk}^{(l)}} \sum_m \left(W_{i\neq j,m}^{(l)} x_m^{(l-1)}\right) + b_{i\neq j}^{(l)} \\
&= 0
\end{aligned}$$

6. Derivative of the linear module with respect to the bias

Calculate derivative for $i = j$:

$$\begin{aligned}\frac{\partial \tilde{x}^{(l)}_{i=j}}{\partial b_j} &= \frac{\partial}{\partial b_j}\sum_k \left(W^{(l)}_{i=j,k}x^{(l-1)}_k\right) + b^{(l)}_{i=j} \\ &= 1\end{aligned}$$

Calculate derivative for $i \neq j$:

$$\begin{aligned}\frac{\partial \tilde{x}^{(l)}_{i\neq j}}{\partial b_j} &= \frac{\partial}{\partial b_j}\sum_k \left(W^{(l)}_{i\neq j,k}x^{(l-1)}_k\right) + b^{(l)}_{i\neq j} \\ &= 0\end{aligned}$$

1.2 Question 1.1

$$\begin{aligned}\frac{\partial L}{\partial \tilde{x}^{(N)}_i} &= \sum_j \frac{\partial L}{\partial x^{(N)}_j}\frac{\partial x^{(N)}_j}{\partial \tilde{x}^{(N)}_i} \\ &= -\frac{t_i}{x^{(N)}_i}x^{(N)}_{i=j}(1 - x^{(N)}_i) + \sum_{j\neq i} -\frac{t_j}{x^{(N)}_j}(-x^{(N)}_j x^{(N)}_i) \\ &= -t_i + t_i x^{(N)}_i + \sum_{j\neq i} t_j x^{(N)}_i \\ &= x^{(N)}_i(t_i + \sum_{j\neq i} t_j) - t_i \\ &= x^{(N)}_i - t_i\end{aligned}$$

$$\frac{\partial L}{\partial \tilde{x}^{(N)}} = x^{(N)} - t$$

$$\begin{aligned}\frac{\partial L}{\partial \tilde{x}^{(l<N)}_i} &= \sum_j \frac{\partial L}{\partial \tilde{x}^{(l)}_j}\frac{\partial x^{(l)}_j}{\partial \tilde{x}^{(l)}_i} \\ &= \sum_j \frac{\partial L}{\partial \tilde{x}^{(N)}_j}\frac{\partial x^{(l)}_j}{\partial \tilde{x}^{(l)}_i}\end{aligned}$$

If $\tilde{x}^{(l)}_i <= 0$:

$$\frac{\partial L}{\partial \tilde{x}^{(l<N)}_i} = 0$$

$$\frac{\partial L}{\partial x^{(l<N)}_i} = 0$$

$$\frac{\partial L}{\partial W^{(l)}_{jk}} = 0$$

$$\frac{\partial L}{\partial b} = 0$$

If $\tilde{x}^{(l)}_i > 0$:

$$\frac{\partial L}{\partial \tilde{x}^{(l<N)}_i} = x^{(N)}_i - t_i$$

$$\begin{aligned}\frac{\partial L}{\partial x_i^{(l<N)}} &= \sum_j \frac{\partial L}{\partial \tilde{x}_j^{(l+1)}} \frac{\partial \tilde{x}_j^{(l+1)}}{\partial x_i^{(l)}} \\ &= \sum_j (x_j^{(N)} - t_j) W_{ji}\end{aligned}$$

$$\begin{aligned}\frac{\partial L}{\partial W_{jk}^{(l)}} &= \sum_i \frac{\partial L}{\partial \tilde{x}_j^{(l)}} \frac{\partial \tilde{x}_j^{(l)}}{\partial W_{jk}^{(l)}} \\ &= \sum_i (x_i^{(N)} - t_i) x_k^{(l-1)}\end{aligned}$$

$$\begin{aligned}\frac{\partial L}{\partial b_i^{(l)}} &= \sum_i \frac{\partial L}{\partial \tilde{x}_j^{(l)}} \frac{\partial \tilde{x}_j^{(l)}}{\partial b_i^{(l)}} \\ &= \sum_j (x_j^{(N)} - t_j)\end{aligned}$$

Vector representations of the derivatives are as follows:

$$\frac{\partial L}{\partial \tilde{x}_i^{(l<N)}} = \max\left(0, (x^{(N)} - t)\right)$$

$$\frac{\partial L}{\partial x_i^{(l<N)}} = \max\left(0, (x^{(N)} - t)^T W_i\right)$$

$$\frac{\partial L}{\partial W_{jk}^{(l)}} = \max\left(0, (x^{(N)} - t) x_k^{T^{(l-1)}}\right)$$

$$\frac{\partial L}{\partial b} = \max\left(0, (x^{(l)} - t)\right)$$

Figure 1.2 shows that the model quickly reaches the top of 0.47 accuracy and declining afterwards. Hence the model is probably already overfitting, and I recommend using a lower learning rate and/or fewer steps to optimize the learning process.

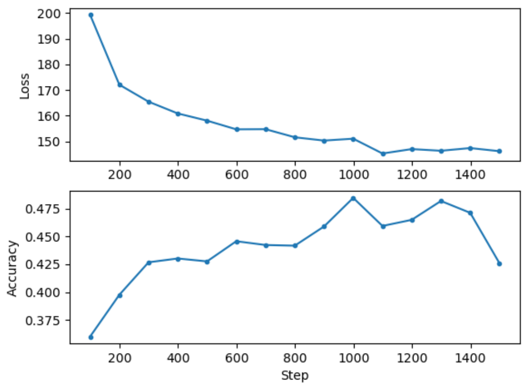

Figure 1: Numpy implementation with basic settings

2 MLP Pytorch

Hidden units	Learning rate	Max steps	Batch size	Accuracy
1x100	2e-3	1500	200	39.6%
1x500	2e-3	1500	200	32.3%
1x1000	2e-3	1500	200	24.8%
1x1000	2e-3	5000	200	39.1%
1x1000	2e-3	10000	200	41.1%
5x100	2e-3	1500	200	44.8%
5x100	2e-3	5000	200	50.0%
5x100	2e-3	10000	200	50.2%
3x1000	2e-3	5000	200	50.0%
3x1000	2e-3	10000	200	51.8%
5x1000	2e-3	5000	200	50.4%
5x1000	2e-3	10000	200	50.0%
5x1000	1e-3	5000	200	50.6%
5x1000	1e-3	10000	200	51.0%
3x4000	2e-3	5000	200	52.1%
3x4000	2e-3	10000	200	53.5%

Table 1: MLP Pytorch optimization

First of all relatively small network were tried, but it was quickly found that they were not able to contain enough information to differentiate between the images. Therefore larger networks were tried (3x1000, 5x1000, 3x4000), which required more steps to complete the training process. Whenever a training process had a very fluctuating loss, as with the 5x1000 neural net, the learning rate was halved. In the end the 3x4000 neurons performed best with an accuracy of 53.5% after 10.000 steps. Figure 2 shows that the model reached the end of its capabilities during the 75000 - 80000 range. After that the loss continues to decline, but no performance improvements on the test set were observed. Hence we conclude that this is the early stage of overfitting, with not yet having negative results on the test set.

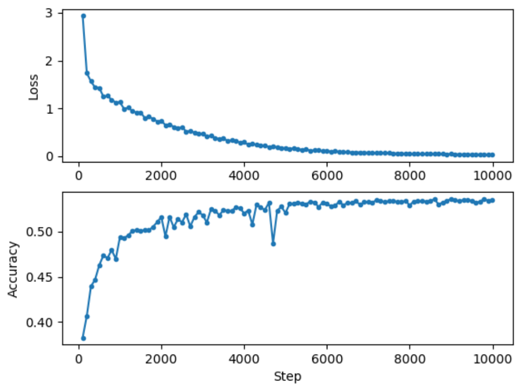

Figure 2: Pytorch 3x4000 MLP Accuracy and Loss plot

3 Custom Batch Normalization

1.

$$\begin{aligned}\frac{\partial L}{\partial \gamma} &= \frac{\partial L}{\partial y}\frac{\partial y}{\partial \gamma} \\ &= \frac{\partial L}{\partial y}\hat{x}\end{aligned}$$

2.

$$\begin{aligned}\frac{\partial L}{\partial \beta} &= \frac{\partial L}{\partial y}\frac{\partial y}{\partial \beta} \\ &= \frac{\partial L}{\partial y}\end{aligned}$$

3.

$$\begin{aligned}(\frac{\partial L}{\partial x})^r_j &= \sum_s \sum_i \frac{\partial L}{\partial y^s_i}\frac{\partial y^s_i}{\partial x^r_j} \\ &= \sum_s \sum_i \frac{\partial L}{\partial y^s_i}\frac{\partial y^s_i}{\partial \hat{x}^r_j}\frac{\partial \hat{x}^r_j}{\partial x^r_j} + \frac{\partial L}{\partial \hat{x}^s_i}\frac{\partial \hat{x}^s_i}{\partial \sigma^2_i}\frac{\partial \sigma^2_i}{\partial x^r_j} + \frac{\partial L}{\partial \hat{x}^s_i}\frac{\partial \hat{x}^s_i}{\partial \mu_i}\frac{\partial \mu_i}{\partial x^r_j} \\ &= \sum_s \sum_i \frac{\partial L}{\partial \hat{x}^r_j}\frac{1}{\sqrt{\sigma^2_i + \epsilon}} - 0.5\frac{\partial L}{\partial \hat{x}^s_i}\frac{x^s_i - \mu_i}{(\sigma^2_i + \epsilon)\sqrt{\sigma^2_i + \epsilon}}\frac{2(x^r_j - \mu_j)}{B} - \frac{\partial L}{\partial \hat{x}^s_i}\frac{1}{\sqrt{\sigma^2_i + \epsilon}}\frac{1}{B} \\ &= \sum_s \sum_i \frac{\partial L}{\partial \hat{x}^r_j}\frac{1}{\sqrt{\sigma^2_i + \epsilon}} - \frac{\partial L}{\partial \hat{x}^s_i}\frac{1}{B\sqrt{\sigma^2_i + \epsilon}}\hat{x}^s_i\hat{x}^r_j - \frac{\partial L}{\partial \hat{x}^s_i}\frac{1}{B\sqrt{\sigma^2_i + \epsilon}} \\ &= \sum_s \sum_i \frac{B\frac{\partial L}{\partial \hat{x}^r_j} - -\frac{\partial L}{\partial \hat{x}^s_i}\hat{x}^s_i\hat{x}^r_j - \frac{\partial L}{\partial \hat{x}^s_i}}{B\sqrt{\sigma^2_i + \epsilon}}\end{aligned}$$

4 PyTorch CNN

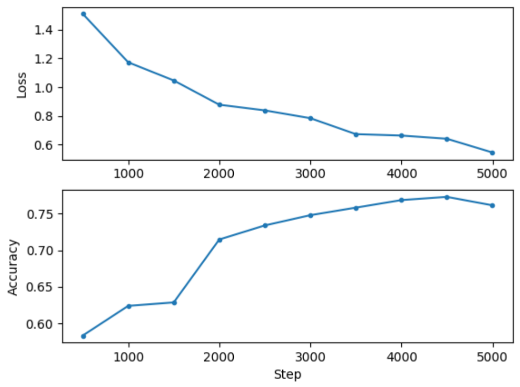

Figure 3: Pytorch CNN Accuracy and Loss plot

With these settings, Figure 4 shows that the loss is still decreasing but without additional improvements. This could mean that additional steps will lead to overfitting the model. But all with all a great improvement in maximum accuracy in comparison with the MLP model.