Ben-Gurion University - IEM Department
Introduction to Deep Learning (364-2-1071)
2023 Semester B
Homework 1

names: Stav Rozenfeld and Adi Gilboa

Question 1

1.a.

First we will check the dimension of the multiplication: $\mathbf{X}^\top\mathbf{B}\mathbf{X} = (1 \times n)(n \times n)(n \times 1) = (1 \times 1)$.

$\frac{\partial(\mathbf{X}^\top\mathbf{B}\mathbf{X})}{\partial\mathbf{X}_i} = \frac{\partial}{\partial X_i}\sum_{k,p} X_p B_{pk} X_k = \sum_{k,p}\frac{\partial(\mathbf{X_p})}{\partial\mathbf{X}_i} B_{pk} X_k + \sum_{k,p} X_p B_{pk}\frac{\partial(\mathbf{X_k})}{\partial\mathbf{X}_i}$
We know that delta is equal to one only when the indexes is equal therefore:
$\sum_k \delta_{ii} B_{ik} X_k + \sum_p X_p B_{pi}\delta_{ii} = \sum_k B_{ik} X_k + \sum_p X_p B_{pi} = \mathbf{X}^\top[\mathbf{B}]_i + [\mathbf{B}]_i\mathbf{X}$

1.b.

First we will check the dimension of the multiplication: $\mathbf{VXW} = (n \times m)(m \times p)(p \times n) = (n \times n)$.
Trace is a scalar value that represent the summation of the diagonal of a matrix.
$trace(\mathbf{VXW}) = \sum_i(\mathbf{VXW})_{ii} = \sum_{k,p,q} V_{kp} X_{pq} W_{qk}$
Now we will calculate the derivative of the trace:
$\frac{\partial(trace(\mathbf{VXW}))}{\partial\mathbf{X}_{ij}} = = \frac{\partial}{\partial X_{ij}}\sum_{k,p,q} V_{kp} X_{pq} W_{qk} = \sum_{k,p,q} V_{kp}\frac{\partial X_{pq}}{\partial X_{ij}} W_{qk} = \sum_{k,p,q} V_{kp}\delta_{pi}\delta_{qj} W_{pk} = \sum_k V_{ki} W_{ik}$

1.c.

$\|W\| := \sqrt{\mathbf{W}^\top\mathbf{W}}$, $\mathbf{W}^\top\mathbf{W} = \sum_k W_k W_k$
$\frac{\partial\|W\|}{\partial\mathbf{W}_i} = \frac{\partial\sqrt{\sum_k W_k W_k}}{\partial\mathbf{W}_i} = \frac{1}{2}(\sum_k W_k W_k)^{\frac{-1}{2}}\sum_k(\frac{\partial W_k}{\partial\mathbf{W}_i} W_k + W_k\frac{\partial W_k}{\partial\mathbf{W}_i}) = \frac{1}{2}(\sum_k W_k W_k)^{\frac{-1}{2}}\sum_k \delta_{ki} W_k + W_k\delta_{ki}$
$= \frac{1}{2}(\sum_k W_k W_k)^{\frac{-1}{2}} 2W_i = \frac{W_i}{\sqrt{\sum_k W_k W_K}}$

1.d.

A squared matrix is a matrix where the number of rows is equal to the number of columns. That's mean that $S \in \mathbb{R}^{(n \times n)}$. So the trace of S is equal to $\sum_K S_{kk}$.

$\frac{\partial trace(S)}{\partial S_{ij}} = \frac{\partial \sum_K S_{kk}}{\partial S_{ij}} = \sum_K \frac{\partial S_{kk}}{\partial S_{ij}} = \sum_k \delta_{ki}\delta_{kj}$

Question 2

2.a.

Consider a linear module as described in the assignment and L is a scalar function of the matrix Y and that the gradient of L with respect to Y is known.

2.a.1

$\frac{\partial L}{\partial \mathbf{W}} = \frac{\partial L}{\partial \mathbf{Y}}\frac{\partial \mathbf{Y}}{\partial \mathbf{W}} = \sum_{k,p,q} \frac{\partial L}{\partial \mathbf{Y}_{ij}}\frac{\partial \mathbf{Y}_{kq}}{\partial \mathbf{W}} = \sum_{k,p,q} \frac{\partial L}{\partial \mathbf{Y}_{ij}}\frac{\partial (x_{kp}w_{pq}+b_q)}{\partial W_{ij}}$

Now the derivative is for W and b is a scalar and the derivative is equal to zero.

$= \frac{\partial L}{\partial \mathbf{Y}} \sum_{k,p,q} \delta_{pi}\delta_{qj}x_{kp} = \frac{\partial L}{\partial \mathbf{Y}} \sum_k x_{ki} = [\frac{\partial L}{\partial \mathbf{Y}}]^\top [X]$

2.a.2

$\frac{\partial L}{\partial \mathbf{b}} = \frac{\partial L}{\partial \mathbf{Y}}\frac{\partial \mathbf{Y}}{\partial \mathbf{b}} = \sum_{k,p,q} \frac{\partial L}{\partial \mathbf{Y}_{ij}}\frac{\partial \mathbf{Y}_{kq}}{\partial \mathbf{b}} = \sum_{k,p,q} \frac{\partial L}{\partial \mathbf{Y}_{ij}}\frac{\partial (x_{kp}w_{pq}+b_q)}{\partial b_j}$

Now the derivative is for b and the rest is scalars.

$= \frac{\partial L}{\partial \mathbf{Y}} \sum_q \delta_{qj} = [1]^\top \frac{\partial L}{\partial \mathbf{Y}}$

2.a.3

$\frac{\partial L}{\partial \mathbf{X}} = \frac{\partial L}{\partial \mathbf{Y}}\frac{\partial \mathbf{Y}}{\partial \mathbf{X}} = \sum_{k,p,q} \frac{\partial L}{\partial \mathbf{Y}_{ij}}\frac{\partial \mathbf{Y}_{kq}}{\partial \mathbf{X}} = \sum_{k,p,q} \frac{\partial L}{\partial \mathbf{Y}_{ij}}\frac{\partial (x_{kp}w_{pq}+b_q)}{\partial X_{ij}}$

Now the derivative is for X and b is a scalar and the derivative is equal to zero.

$= \frac{\partial L}{\partial \mathbf{Y}} \sum_{k,p,q} \delta_{ki}\delta_{pj}w_{pq} = \frac{\partial L}{\partial \mathbf{Y}} \sum_q w_{jq} = [\frac{\partial L}{\partial \mathbf{Y}}][W]$

2.b.

2.b.i

When h is a generic activation function: $\frac{\partial L}{\partial \mathbf{X}} = \frac{\partial L}{\partial \mathbf{Y}}\frac{\partial \mathbf{Y}}{\partial \mathbf{X}} = \frac{\partial L}{\partial Y} \sum_{k,q} \frac{\partial Y_{kp}}{\partial X_{ij}} = \frac{\partial L}{\partial Y} \sum_{k,q} \frac{\partial h(X_{kp})}{\partial X_{ij}} = \frac{\partial L}{\partial Y} \sum_{k,q} h'(X_{kp})\frac{\partial X_{kp}}{\partial X_{ij}} = \frac{\partial L}{\partial Y} \sum_{k,q} h'(X_{kp})\delta_{ki}\delta_{pj} = \frac{\partial L}{\partial Y} \circ h'(X_{ij}) = \frac{\partial L}{\partial Y} \circ \mathbf{h}'(\mathbf{X})$

2.b.ii

For the ReLU case, the element-wise relationship becomes Yij $=$ max(0, Xij).
$\frac{\partial L}{\partial \mathbf{X}} = \frac{\partial L}{\partial \mathbf{Y}} \frac{\partial \mathbf{Y}}{\partial \mathbf{X}} = \frac{\partial L}{\partial Y} \sum_{k,q} \frac{\partial Y_{kp}}{\partial X_{ij}} = \frac{\partial L}{\partial Y} \sum_{k,q} \frac{\partial h(X_{kp})}{\partial X_{ij}} = \frac{\partial L}{\partial Y} \sum_{k,q} h'(X_{kp}) \frac{\partial X_{kp}}{\partial X_{ij}} = \frac{\partial L}{\partial Y} \sum_{k,q} h'(X_{kp}) \delta_{ki} \delta_{pj} =$
$\frac{\partial L}{\partial Y} \circ (X_{ij} > 0) = \frac{\partial L}{\partial Y} \circ (\mathbf{X} > 0)$
when $(X_{ij} > 0)$ is an indicator function that evaluates to 1 if X_{ij}> 0 and 0 otherwise.

2.c.

2.c.i

In this question, we are dealing with the final module before the loss evaluation in terms of $\frac{\partial L}{\partial \mathbf{Y}}$.
$\frac{\partial L}{\partial \mathbf{X}} = \frac{\partial L}{\partial \mathbf{Y}} \sum_{k,p} \frac{\partial \mathbf{Y}_{kp}}{\partial \mathbf{X}_{ij}}$
The softmax is defined as: $Y_{ij} = \frac{e^{X_{ij}}}{\sum_k e^{X_{ik}}}$.
$\frac{\partial (\text{softmax}(x_{ij}))}{\partial x_{ij}} = \frac{\partial \frac{e^{x_{ij}}}{\sum_k e^{x_{ik}}}}{\partial X_{ij}} = \frac{e^{x_{ij}} \sum_k e^{x_{ik}} - e^{2x_{ij}}}{(\sum_k e^{x_{ik}})^2}$
$\frac{\partial L}{\partial \mathbf{X}} = \frac{\partial L}{\partial \mathbf{Y}} \frac{e^{x_{ij}} \sum_k e^{x_{ik}} - e^{2x_{ij}}}{(\sum_k e^{x_{ik}})^2} = \frac{\partial L}{\partial \mathbf{Y}} (\frac{e^{x_{ij}} \sum_k e^{x_{ik}}}{(\sum_k e^{x}_{ik})^2} - \frac{e^{2x_{ij}}}{(\sum_k e^{x}_{ik})^2})$
Therefore, the final answer is: $\frac{\partial L}{\partial \mathbf{Y}} (softmax(x) - softmax(x)^2) =$
$\frac{\partial L}{\partial \mathbf{Y}} (softmax(x)) - \frac{\partial L}{\partial \mathbf{Y}} (softmax(x)) 1 (softmax(x))$

2.c.ii

In this question, we are dealing with the final module before the loss evaluation: $Y_{ij} = \text{softmax}(X_{ij})$, and the loss function is defined as: $L_i = -\frac{1}{S} \sum_k T_{ik} \log(x_{ik})$ for sample i.
The derivative of the loss with respect to $\mathbf{X}$ is given by:
$\frac{\partial L}{\partial \mathbf{X_{ij}}} = \frac{\partial (-\frac{1}{s}) \sum_{pk} T_{pk} log(x_{pk})}{\partial X_{ij}} = -\frac{1}{s} \sum_{pk} T_{pk} \delta_{pi} \delta_{kj} \frac{1}{x_{pk}} = -\frac{1}{s} \frac{T_{ij}}{X_{ij}} = -\frac{1}{s} \mathbf{T} \oslash \mathbf{X}$

Question 3

The achieved accuracy and loss curve for the default parameters:

Validation accuracies: [0.4244, 0.4672, 0.4586, 0.482, 0.411, 0.4588, 0.4648, 0.4638, 0.483, 0.4526]

Test accuracy: 0.4794

training losses: [1.8858180613234825, 1.55984281138002, 1.4876859021264848, 1.4454249754220376, 1.4136744187796868, 1.3828731619405819, 1.3473959059996976, 1.337990804963375, 1.3071571058731482, 1.2830195761435697]

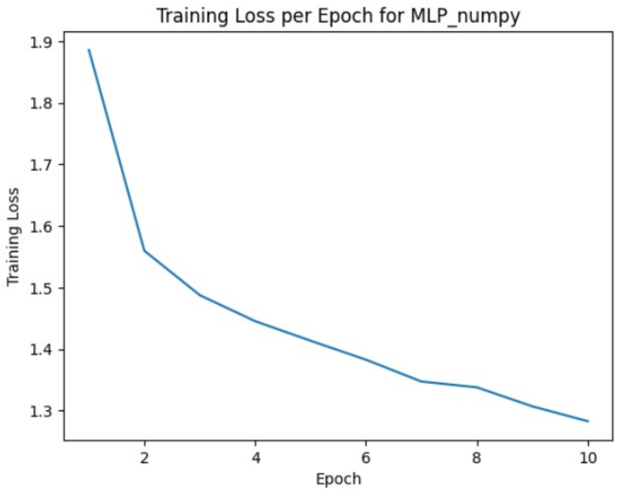

Figure 1: Training loss per epoch for default parameters, MLP numpy implementation

Question 4

4.a

The achieved accuracy and loss curve for the default parameters:
Validation accuracies: [0.4024, 0.4558, 0.4572, 0.481, 0.4674, 0.4862, 0.4568, 0.4858, 0.47, 0.4714]
Test accuracy: 0.4769
It can be seen that this accuracy is similar to the test accuracy in question 3.
training losses: [1.8826203600973146, 1.5337753781565913, 1.4626478707348858, 1.4233761142801356, 1.3696179824676948, 1.341387393807414, 1.3101794960831645, 1.274615979772008, 1.2484933640542533, 1.238007300429874]

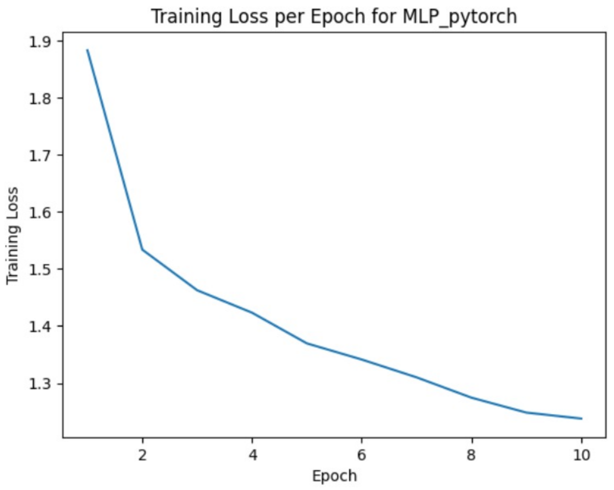

Figure 2: Training loss per epoch for default parameters, PyTorch implementation

4.b.

4.b.i

The learning rate is a crucial factor that determines how much the model should be adjusted during each update of its weights based on the estimated error. Selecting an appropriate learning rate can be challenging.
If the learning rate too small, the training process may be slow and prone to getting stuck. This can result in a **high loss value**, indicating poor model performance. As a result, **the accuracy of the model on the training and validation data may decrease** and the overall **training time will increase**.
If the learning rate too large, the model may learn suboptimal weights too quickly or experience an unstable training process. This can result in a **high loss value**, indicating poor model performance. **The accuracy of the model may take a longer time to improve** or reach its optimal performance. Setting a very large learning rate can significantly **increase the training time** as the model requires more iterations to converge. The training process may become inefficient and **time-consuming**.

4.b.ii

The basic idea for learning rate schedule is to gradually reduce the learning rate over time to allow the model to make smaller and more precise updates as it converges.
Usually, the learning rate is set to a higher value at the beginning of the training to allow faster convergence.
As the training progresses, the learning rate is reduced to enable convergence to the optimum and thus leading to better performance.
In the image below, we implement the Scheduler using 'MultiStepLR' of pytorch - reduces the learning rate by a multiplicative factor after each pre-defined mileston.

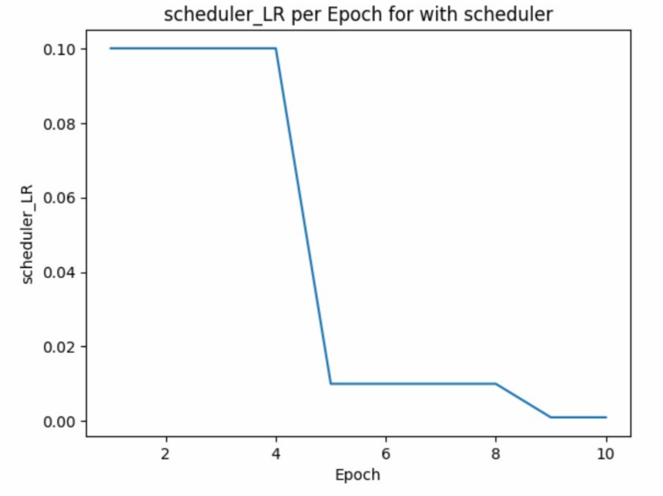

Figure 3: Scheduler implementation, starting from the default value with gamma = 0.1 in two milstones.

4.b.iii

The effect of different learning rates on the performance of the MLP network was investigated. The model was trained with nine different learning rates, ranging from 0.000001 to 100 at equal logarithmic intervals.

Based on the results, the best learning rate for the model was found to be **0.1 (1e-01) with an accuracy of 0.4962.** This learning rate demonstrated superior performance compared to other options as it balanced the trade-off between convergence speed and avoiding overshooting or oscillation during the optimization process. It allowed the model to update the weights and biases effectively, resulting in improved accuracy on the validation set.

4.b.iv

The plots of the two figures:

- Best validation accuracy as function of the learning rates
- Loss curves with different learning rates over time (iteration or epoch)

are:

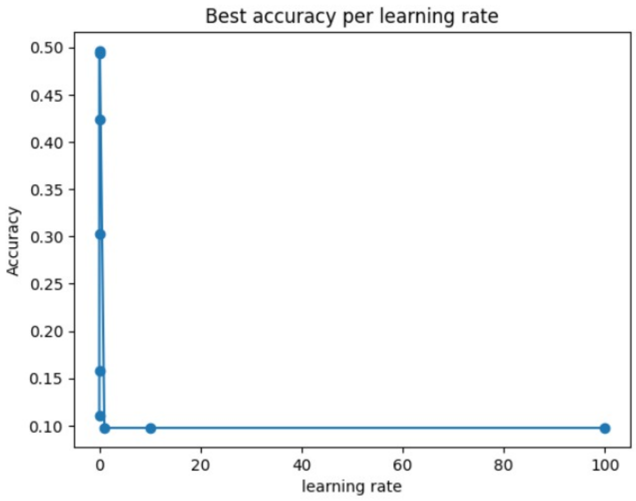

Figure 4: Best validation accuracy as function of the learning rates

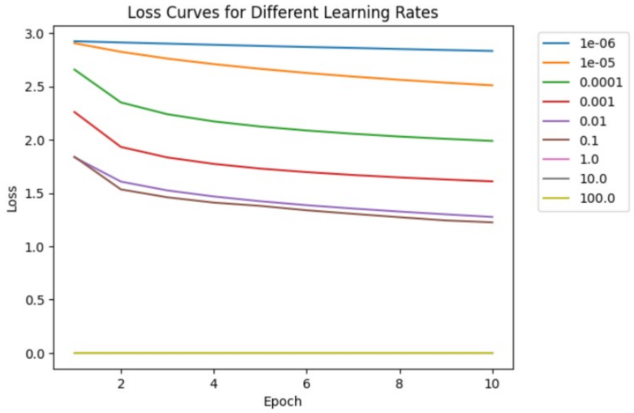

Figure 5: Loss curves with different learning rates over time

Question 5

Given that f is a random variable with a symmetric probability density around 0, i.e., $p(f) = p(-f)$, and the variance of f is σ^2, the variable b is defined as $b = \text{ReLU}(f) = \max(f, 0)$. We want to show that $\mathbb{E}(b^2) = \frac{\sigma^2}{2}$, which represents the second moment of the random variable b.

$\text{ReLU}(f)$ means that for all $f < 0$, $\text{ReLU}(f) = 0$, and for all $f \geq 0$, $\text{ReLU}(f) = f$.
Due to the definition of f, we understand that $\mathbb{E}(f) = 0$ because $p(f) = p(-f)$. This means that half of the time f is negative and half of the time f is positive, and in a symmetrical way as well.

Since $b = \text{ReLU}(f)$, we can define $\mathbb{E}(b^2) = \mathbb{E}[\text{ReLU}(f)^2]$.

Due to the previous point, we understand that half of the time when f is negative, then $\mathrm{ReLU}(f) = 0$, and half of the time when f is positive, then $\mathrm{ReLU}(f) = f$.
Therefore, we can write $\mathbb{E}[\mathrm{ReLU}(f)^2]$ as: $\mathbb{E}[\mathrm{ReLU}(f)^2] = 0.5 * \mathbb{E}(f^2|f \geq 0) + 0.5 * \mathbb{E}(0|f < 0)$.

Regarding the first part of the expression: $0.5 \cdot \mathbb{E}(f^2|f \geq 0)$, we will see what $\mathbb{E}(f^2)$ is:
$\mathrm{Var(f)} = \mathbb{E}(f^2) - (\mathbb{E}(f))^2$
As we mentioned before, $\mathbb{E}(f) = 0$, therefore $\mathrm{Var}(f) = \sigma^2 = \mathbb{E}(f^2)$.

Since f is symmetric about 0, then also $\mathbb{E}(f^2|f \geq 0) = \sigma^2$.
(Due to the symmetries, the positive and negative values are offset with each other, so the mean of the positive values will be the same as the mean of all values).

Regarding the second part of the expression, when $f < 0$, $\mathrm{ReLU}(f) = 0$ and therefore also $\mathrm{ReLU}(f)^2 = 0$, and $\mathbb{E}(0) = 0$.

Therefore, we get that $\mathbb{E}[\mathrm{ReLU}(f)^2] = 0.5 * \sigma^2 = \frac{\sigma^2}{2}$.
And hence: $\mathbb{E}[\mathrm{ReLU}(f)^2] = \mathbb{E}(b^2) = \frac{\sigma^2}{2}$.