Homework 2

Aprendizagem Profunda

2024/2025

Grupo 86

Instituto Superior Técnico – Universidade de Lisboa

Índice

Contribution of each member of the group

All members of the group participated in every stage of this homework. Each member contributed equally to implementing the code, analyzing the results and writing the report.

Question 1

1.

The energy can be written as:

$$E(q) = E_1(q) + E_2(q),$$

where:

$$E_1(q) = \frac{1}{2}\|q\|^2 + \underbrace{\beta^{-1}\log N + \frac{1}{2}M^2}_{\text{constant in } q} \quad \text{and} \quad E_2(q) = -\text{lse}\big(\beta, Xq\big) = \beta^{-1}\log\sum_{i=1}^{N}\exp\big(\beta z_i\big).$$

As for the gradient of $E_1(q)$, two of the terms of the sum do not depend on q, which means their gradient is zero:

$$\nabla E_1(q) = \frac{\partial}{\partial q}\Big(\frac{1}{2}\|q\|^2\Big) = \frac{\partial}{\partial q}\Big(\frac{1}{2}q^\top q\Big) = q.$$

Now, for the gradient of $E_2(q)$, with

$$z = Xq \quad \text{and} \quad S = \sum_{i=1}^{N}\exp\big(\beta z_i\big),$$

we compute each partial derivative with respect to z_j:

$$\frac{\partial}{\partial z_j}\big(-\text{lse}\big(\beta, z\big)\big) = -\beta^{-1}\cdot\frac{\partial}{\partial z_j}\big(\log(S)\big) = -\frac{1}{\beta S}\cdot\frac{\partial S}{\partial z_j} = -\frac{1}{\beta S}\cdot\frac{\partial}{\partial z_j}\Big(\sum_{i=1}^{N}\exp(\beta z_i)\Big) =$$

$$= -\frac{1}{\beta S}\cdot\frac{\partial}{\partial z_j}\Big(\exp(\beta z_j)\Big) = -\frac{\beta}{\beta S}\exp(\beta z_j) = -\frac{\exp(\beta z_j)}{S} = -\frac{\exp(\beta z_j)}{\sum_{i=1}^{N}\exp\big(\beta z_i\big)} = -[\text{softmax}(\beta z)]_j.$$

So, using the chain rule to go from ∇_z to ∇_q:

$$\nabla E_2(q) = -\frac{\partial z}{\partial q}\cdot\nabla_z\big(-\text{lse}\big(\beta, z\big)\big) = -X^\top\text{softmax}(\beta Xq).$$

Because $\nabla E_1(q) = q$, the Hessian of $E_1(q)$ is the identity matrix:

$$\nabla^2 E_1(q) = I_{D\times D}.$$

This matrix is positive definite, hence also positive semidefinite, which confirms $E_1(q)$ is a convex function.
And for the Hessian of $E_2(q)$, let:

$$s = \text{softmax}(\beta Xq) \in \mathbb{R}^N.$$

Then:

$$\nabla\big(E_2(q)\big) = -X^\top s.$$

By the chain rule:

$$\nabla^2 E_2(q) = \nabla_q\big(-X^\top s\big) = -X^\top \cdot \nabla_q s = -X^\top \cdot \frac{\partial s}{\partial z} \cdot \frac{\partial z}{\partial q} = -X^\top \cdot \frac{\partial s}{\partial z} \cdot X =$$

$$= -X^\top \cdot \nabla_z \text{softmax}\big(\beta z\big) \cdot X = -X^\top \Big(\beta[\text{diag}(s) - ss^\top]\Big)X = -\beta X^\top \big[\text{diag}(s) - ss^\top\big]X.$$

But we have to deduce the expression of the derivative of the softmax function:

$$\nabla_z \text{softmax}(\beta z) = \begin{bmatrix} \frac{\partial s_1}{\partial z_1} & \frac{\partial s_1}{\partial z_2} & \cdots & \frac{\partial s_1}{\partial z_N} \\ \frac{\partial s_2}{\partial z_1} & \frac{\partial s_2}{\partial z_2} & \cdots & \frac{\partial s_2}{\partial z_N} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial s_N}{\partial z_1} & \frac{\partial s_N}{\partial z_2} & \cdots & \frac{\partial s_N}{\partial z_N} \end{bmatrix}$$

For each partial derivative:

$$\frac{\partial}{\partial z_j} \log s_i = \frac{1}{s_i} \cdot \frac{\partial s_i}{\partial z_j} \Leftrightarrow \frac{\partial s_i}{\partial z_j} = s_i \cdot \frac{\partial}{\partial z_j} \log s_i = s_i \cdot \frac{\partial}{\partial z_j}\Big(\beta z_i - \log \sum_{i=1}^{N} \exp\big(\beta z_i\big)\Big) =$$

$$= s_i \cdot \Big(\beta \cdot \frac{\partial z_i}{\partial z_j} - \frac{\partial}{\partial z_j}\Big(\log \sum_{i=1}^{N} \exp\big(\beta z_i\big)\Big)\Big) = s_i \cdot \Big(\beta \cdot 1\{i=j\} - \frac{1}{\sum_{i=1}^{N} \exp\big(\beta z_i\big)}\Big(\frac{\partial}{\partial z_j} \sum_{i=1}^{N} \exp\big(\beta z_i\big)\Big)\Big) =$$

$$= s_i \cdot \Big(\beta \cdot 1\{i=j\} - \frac{1}{\sum_{i=1}^{N} \exp\big(\beta z_i\big)}\Big(\frac{\partial}{\partial z_j} \exp\big(\beta z_j\big)\Big)\Big) = s_i \cdot \Big(\beta \cdot 1\{i=j\} - \frac{\beta \exp\big(\beta z_j\big)}{\sum_{i=1}^{N} \exp\big(\beta z_i\big)}\Big) =$$

$$= \beta \cdot s_i\Big(1\{i=j\} - s_j\Big).$$

So:

$$\nabla_z \text{softmax}(\beta z) = \beta \begin{bmatrix} s_1(1-s_1) & -s_1 s_2 & \cdots & -s_1 s_N \\ -s_2 s_1 & s_2(1-s_2) & \cdots & -s_2 s_N \\ \vdots & \vdots & \ddots & \vdots \\ -s_N s_1 & -s_N s_2 & \cdots & s_N(1-s_N) \end{bmatrix}.$$

Note that:

$$\text{diag}(s) = \begin{bmatrix} s_1 & 0 & \cdots & 0 \\ 0 & s_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & s_N \end{bmatrix}; \quad ss^\top = \begin{bmatrix} s_1 \\ s_2 \\ \vdots \\ s_N \end{bmatrix} \begin{bmatrix} s_1 & s_2 & \cdots & s_N \end{bmatrix} = \begin{bmatrix} s_1 s_1 & s_1 s_2 & \cdots & s_1 s_N \\ s_2 s_1 & s_2 s_2 & \cdots & s_2 s_N \\ \vdots & \vdots & \ddots & \vdots \\ s_N s_1 & s_N s_2 & \cdots & s_N s_N \end{bmatrix};$$

$$\text{diag}(\mathbf{s}) - ss^\top = \begin{bmatrix} s_1 - s_1^2 & -s_1 s_2 & \cdots & -s_1 s_N \\ -s_2 s_1 & s_2 - s_2^2 & \cdots & -s_2 s_N \\ \vdots & \vdots & \ddots & \vdots \\ -s_N s_1 & -s_N s_2 & \cdots & s_N - s_N^2 \end{bmatrix} = \begin{bmatrix} s_1(1-s_1) & -s_1 s_2 & \cdots & -s_1 s_N \\ -s_2 s_1 & s_2(1-s_2) & \cdots & -s_2 s_N \\ \vdots & \vdots & \ddots & \vdots \\ -s_N s_1 & -s_N s_2 & \cdots & s_N(1-s_N) \end{bmatrix}.$$

For any vector v:

$$v^\top \big[\text{diag}(s) - ss^\top\big] v = \sum_{i=1}^{N} s_i v_i^2 - \Big(\sum_{i=1}^{N} s_i v_i\Big)^2.$$

With $g(x) = x^2$, which is a convex function, and given that $\forall i : s_i \geq 0$ and $\sum_{i=1}^{N} s_i = 1$ (because s is a softmax), Jensen's inequality states that:

$$g\Big(\sum_{i=1}^{N} s_i v_i\Big) \leq \sum_{i=1}^{N} s_i g(v_i) \Rightarrow \Big(\sum_{i=1}^{N} s_i v_i\Big)^2 \leq \sum_{i=1}^{N} s_i v_i^2 \Leftrightarrow \sum_{i=1}^{N} s_i v_i^2 - \Big(\sum_{i=1}^{N} s_i v_i\Big)^2 \geq 0$$

Therefore, $\left[\text{diag}(s) - ss^\top\right]$ is positive semidefinite. Multiplying a positive semidefinite matrix by a negative scalar $-\beta$ makes it negative semidefinite, and further sandwiching by $X^\top$ and X keeps it negative semidefinite, which means $\nabla^2 E_2(q)$ is negative semidefinite and, hence, $E_2(q)$ is a concave function.
This proves that the original energy splits into a sum of a convex and a concave term:

$$E(q) = \underbrace{\left(\frac{1}{2}\|q\|^2 + \beta^{-1}\log N + \frac{1}{2}M^2\right)}_{\text{convex}} + \underbrace{\left(-\text{lse}\big(\beta, Xq\big)\right)}_{\text{concave}}.$$

2.

We want to linearize the concave function E_2 using a first-order Taylor approximation around q_t. In the previous exercise, we found that:

$$\nabla E_2(q) = -X^\top \text{softmax}(\beta X q).$$

Therefore:

$$E_2(q) \approx \widetilde{E}_2(q) := E_2(q_t) + (\nabla E_2(q_t))^\top \big(q - q_t\big) = E_2(q_t) - \Big[X^\top \text{softmax}\big(\beta X q_t\big)\Big]^\top \big(q - q_t\big).$$

Now we want to compute a new iterate by solving the convex optimization problem. Since the last two terms of $E_1(q)$, $E_2(q_t)$ and q_t are constant with respect to q, they do not affect the minimizer:

$$q_{t+1} = \underset{q}{\text{argmin}}\Big\{E_1(q) + \widetilde{E}_2(q)\Big\} = \arg\min_q\Big\{\frac{1}{2}\|q\|^2 - \Big[X^\top \text{softmax}\big(\beta X q_t\big)\Big]^\top q\Big\}.$$

So the function to be minimized simplifies to:

$$f(q) = \frac{1}{2}q^\top q - \underbrace{\big(X^\top \text{softmax}(\beta X q_t)\big)^\top}_{=v} q,$$

where $v := X^\top \text{softmax}(\beta X q_t)$. Minimizing via setting the gradient to zero:

$$\nabla f(q) = q - v = 0 \Leftrightarrow q = v = X^\top \text{softmax}(\beta X q_t).$$

Hence:

$$q_{t+1} = X^\top \text{softmax}(\beta X q_t),$$

which is the update rule we wanted to show.

3.

When $\beta = \frac{1}{\sqrt{D}}$, the result of the previous question is:

$$q_{t+1} = X^\top \text{softmax}\Big(\frac{1}{\sqrt{D}} X q_t\Big),$$

The computation performed in the cross-attention layer of a transformer with a single attention head, using the usual Transformer scaling factor (scaled dot-product attention) is:

$$\text{Attn}(Q, K, V) = \text{softmax}\Big(\frac{1}{\sqrt{D}} QK^\top\Big)V \in \mathbb{R}^{N\times D},$$

where:

$$Q = XW^D, \quad K = XW^K, \quad V = XW^V.$$

When $W_K = W_V = I$, then $K = V = X$. The query matrix Q is already computed as:

$$Q_t = \begin{bmatrix} q_t^{(1)} & \dots & q_t^{(N)} \end{bmatrix}^\top = \begin{bmatrix} (q_t^{(1)})^\top \\ \vdots \\ (q_t^{(N)})^\top \end{bmatrix} \in \mathbb{R}^{N\times D}.$$

Hence the attention becomes:

$$\text{Attn}(Q_t, X, X) = \text{softmax}\Big(\frac{1}{\sqrt{D}} Q_t X^\top\Big)X.$$

Now we study how it acts row by row.
The (i, j) entry of the matrix $Q_t X^\top \in \mathbb{R}^{N\times N}$ is the dot-product $(q_t^{(i)})^\top \cdot x_j^\top$ (note that x_j are the rows of the matrix X, so $x_j^\top$ are column-vectors).
Then $\text{softmax}\big(\frac{1}{\sqrt{D}} Q_t X^\top\big)$ is done row-wise, so for each row i we get a row-vector:

$$\alpha_i^\top = \text{softmax}\Big(\frac{1}{\sqrt{D}} (q_t^{(i)})^\top X^\top\Big) \in \mathbb{R}^N,$$

whose components $\alpha_{i,j}$ sum to 1. Multiplying on the right by $X \in \mathbb{R}^{N\times D}$, each row i is:

$$[\text{Attn}(Q_t, X, X)]_i = \alpha_i^\top X = \sum_{j=1}^{N} \alpha_{i,j} x_j \in \mathbb{R}^D.$$

But $\alpha_i = \text{softmax}(\frac{1}{\sqrt{D}} X q_t^{(i)})$ in standard vector form (column-vector), and $\alpha_i^\top X = \Big(X^\top \alpha_i\Big)^\top$.
Hence, the i-th output row is:

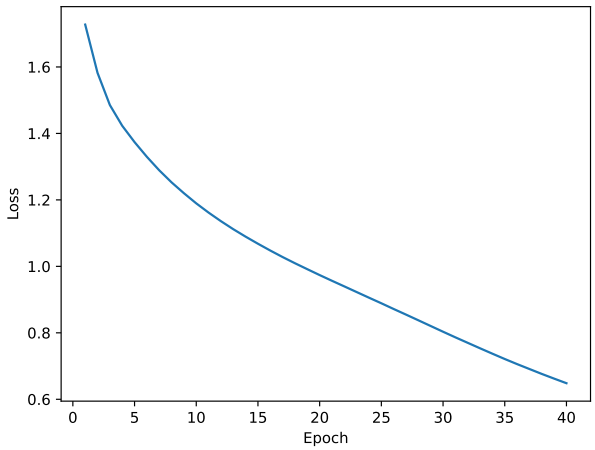
$$[\text{Attn}(Q_t, X, X)]_i = \Big(X^\top \alpha_i\Big)^\top = \Big(X^\top \text{softmax}\Big(\frac{1}{\sqrt{D}} X q_t^{(i)}\Big)\Big)^\top = \Big(q_{t+1}^{(i)}\Big)^\top$$

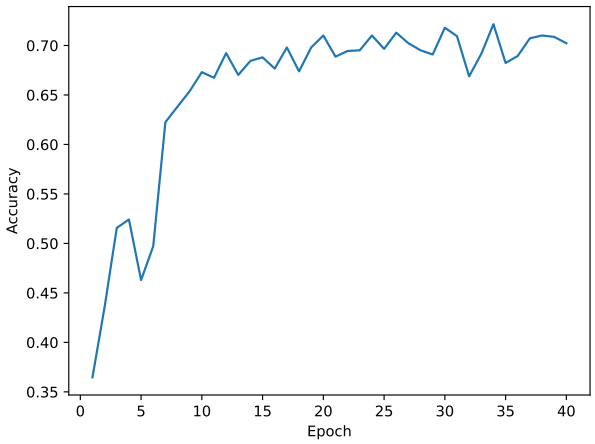
In other words, the Hopfield update for the i-th new state and the cross-attention output for the i-th query are the same operation, which means that setting $\beta = \frac{1}{\sqrt{D}}$ in the modern Hopfield update is the same as applying single-head cross-attention with identity projection matrices W_K and W_V and using scaled dot-product attention.

Question 2

1.

After tuning the learning rate using the values 0.1, 0.01 and 0.001, we conclude that the best configuration is the one with learning rate 0.01.
Plots of the training loss (left) and the validation accuracy (right) as a function of the epoch number:

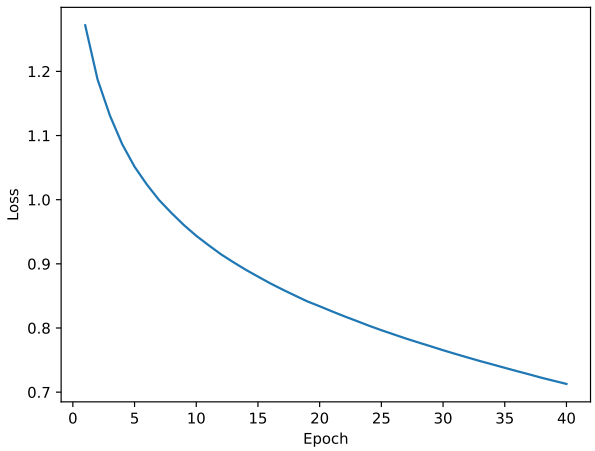


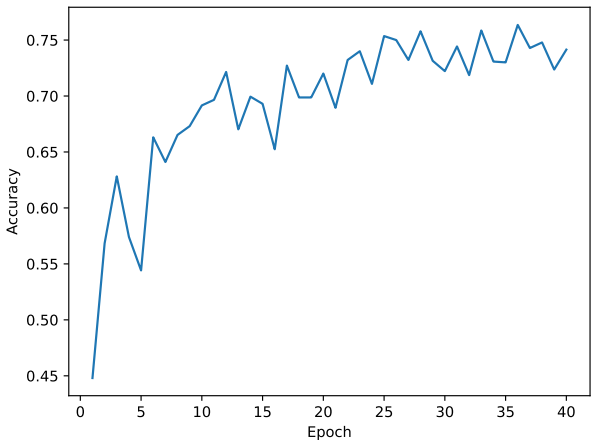


Final validation accuracy: 0.7023
Final test accuracy: 0.6927

2.

Plots of the training loss (left) and the validation accuracy (right) as a function of the epoch number:





Final validation accuracy: 0.7415
Final test accuracy: 0.7280

3.

In the model of Question 2.1., without batch normalization, the number of trainable parameters is 5340742, whereas in the model of Question 2.2., with batch normalization and global average pooling, the number of trainable parameters is 755718.
The number of trainable parameters includes the weights and biases of the convolutional and fully connected layers. In Question 2.2., additional trainable parameters are introduced due to the inclusion of batch normalization layers. However, the global average pooling operation reduces the input dimensionality for the first fully connected layer, leading to fewer parameters in that layer.
In terms of performance, the batch normalization in Question 2.2. improves convergence speed and generalization by normalizing activations within mini-batches. In Question 2.1., flattening results in a higher-dimensional input to the MLP, whereas the global average pooling in Question 2.2. reduces the number of trainable parameters in the MLP block, mitigating overfitting and enhancing translation invariance. These factors contribute to the superior performance of the model in Question 2.2.

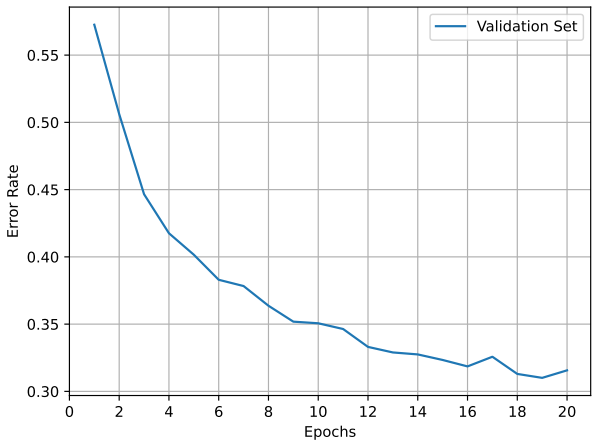
4.

In convolutional neural networks, small kernels are preferred over large ones because they reduce the number of trainable parameters and computational cost while allowing the stacking of multiple layers to achieve a larger receptive field with added non-linearities. This enhances feature extraction and improves parameter efficiency. Pooling layers, on the other hand, reduce spatial dimensions, lowering memory and computational requirements. They also introduce translation invariance, making the network robust to input variations, and help prevent overfitting by reducing the total parameter count. Together, small kernels and pooling layers enable efficient learning of hierarchical features while maintaining computational and parameter efficiency.

Question 3

1. a)

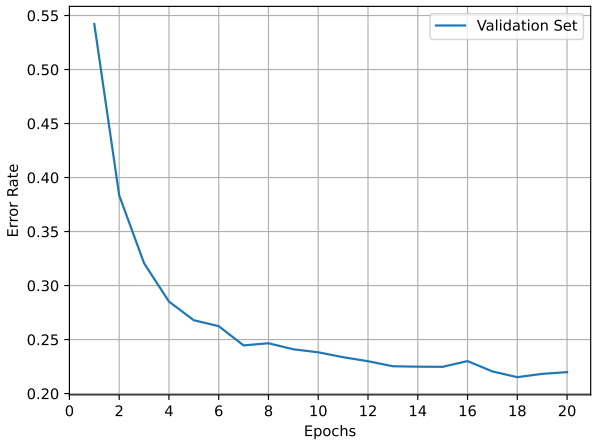
Validation CER at the end of each epoch:



Test CER: 0.2989
Test WER: 0.7960

1. b)

Validation CER at the end of each epoch:



Test CER: 0.2127
Test WER: 0.7350

1. c)

Test CER: 0.2166
Test WER: 0.7490
Test WER@3: 0.7100