January 27th 2025.

Department of Electronics, Microelectronics,
Computer and Intelligent Systems

Deep Learning 1

example tasks

1. We consider a simple recurrent neural network that classifies sequences of two-bit binary numbers into two classes, depending on ratio of odd and even numbers. A sequence belongs to class 0 if the number of even numbers is larger or equal to the number of odd numbers, and to class 1 if there are more odd numbers. For example, a sequence $00, 01, 10, 11, 00$ contains three even and 2 odd numbers which means that it belongs to class 0.

   (a) Determine the exact parameters and appropriate activation functions for a basic recurrent cell and a fully connected output layer that would properly solve this classification problem. Model the input $\boldsymbol{x}^{(t)}$ as a binary column vector with two elements. Express the final model with appropriate equations.

   (b) How many parameters does your model have?

   (c) Demonstrate the correctness of your model on the following test sequence: $00, 10, 11$.

① 0 1 2 3 0

00,01,10,11,00 → #even > #odd ⇒ cl 0

00,01,10,11 → #even = #odd ⇒ cl 0

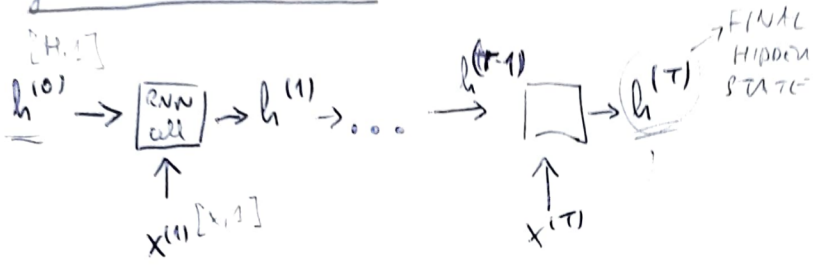
00,01,10,11,01 → #even < #odd ⇒ cl 1

RNN + FC on $h^{(T)}$

binary clf. → sigmoid act. in output layer

---

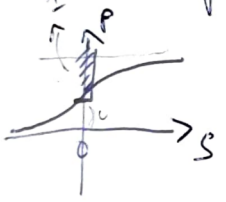
general architecture



[1,H] [H,1]

$$s = W h^{(T)} + b$$

$$p = \sigma(s)$$

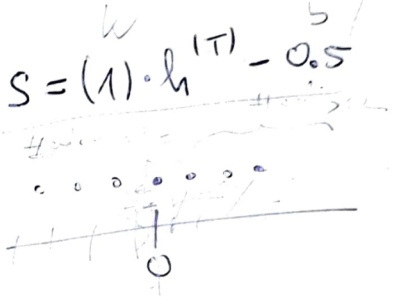


$$cl: \begin{cases} 0, & p < 0.5 \iff \#even \geq \#odd \\ 1, & p \leq 0.5 \iff \#even < \#odd \end{cases}$$

$p < 0.5 \iff s < 0$

$p \leq 0.5 \iff s \geq 0$

$$s = (1) \cdot h^{(T)} - 0.5$$



$$s = (1) \cdot h^{(T)} - 0.5$$

$$h^{(T)} = (\#odd - \#even)$$

$$h^{(t)} = (\#odd - \#even)\ @\ t$$

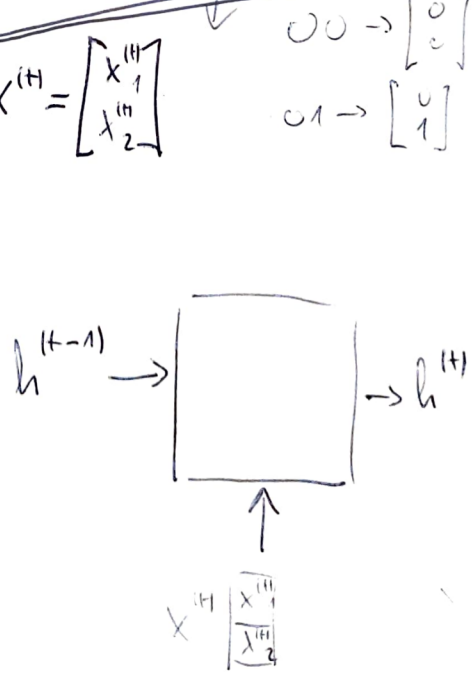
$$h^{(0)} = (\dots)\ @\ 0 = 0$$

$$h^{(t)} = \begin{cases} h^{(t-1)} + 1 & \text{if } x^{(t)} \text{ odd} \\ h^{(t-1)} - 1 & \text{if } x^{(t)} \text{ even} \end{cases}$$

$$x^{(t)} = \begin{bmatrix} x^{(t)}_1 \\ x^{(t)}_2 \end{bmatrix}$$

00 → $\begin{bmatrix} 0 \\ 0 \end{bmatrix}$

01 → $\begin{bmatrix} 0 \\ 1 \end{bmatrix}$



$$h^{(t)} = act\left(W_{hh} h^{(t-1)} + W_{hx} x^{(t)} + b_h\right)$$

$[H,H]$ $[H,I]$ $[H,1]$

$[:,1]$ $[1,2]$ $[:,1]$

$$h^{(t)} = \boxed{\quad} \cdot h^{(t-1)} + \boxed{\quad}\begin{bmatrix} x^{(t)}_1 \\ x^{(t)}_2 \end{bmatrix} + \boxed{\quad}$$

W_{hh} W_{hx} b_h

0	0	0 ez
0	1	1 utána
1	0	2 ...
1	1	3 előtt

$[0\ \ 1]$

-1

$[0\ \ 2]$

$$h^{(t)} = (1) \cdot h^{(t-1)} + [0\ \ 2]\begin{bmatrix} x^{(t)}_1 \\ x^{(t)}_2 \end{bmatrix} + (-1)$$

---

b) RNN: $W_{hh} \to 1$, $W_{hx} \to 2$, $b_h \to 1$ } 4

FC: $W \to 1$, $b \to 1$ } 2

$\to$ **6**

---

c) $h^{(0)} = 0$

$h^{(1)} = (1)\cdot 0 + [0\ \ 2]\begin{bmatrix} 0 \\ 0 \end{bmatrix} + (-1) = (-1)$

0 $x^{(1)} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$

$h^{(2)} = (1)(-1) + [0\ \ 2]\begin{bmatrix} 1 \\ 0 \end{bmatrix} + (-1) = (-2)$

2 $x^{(2)} = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$

$h^{(T)} = h^{(3)} = (1)(-2) + [0\ \ 2]\begin{bmatrix} 1 \\ 1 \end{bmatrix} + (-1) = (-1)$

3 $x^{(3)} = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$

$S = (1)\cdot(-1) - 0.5 = -1.5$

$p = \sigma(S) = \sigma(-1.5) < 0.5 \to$ cl: 0

2. We consider siamese metric learning with a quadratic contrastive loss. We perform the embedding with a single convolutional layer without padding and with stride 1, The initial parameters of the convolutional layer are:

$$\mathbf{w} = [\ \tfrac{1}{3} \quad \tfrac{1}{3} \quad \tfrac{1}{3}\ ], \qquad b = -0.1$$

We consider the following positive pair as an input:

$$\mathbf{x}_{p1} = [\ 1.0 \quad 1.0 \quad 1.3 \quad 1.0 \quad 1.0 \quad 1.3\ ], \qquad \mathbf{x}_{p2} = [\ 3.3 \quad 3.3 \quad 2.7 \quad 3.3 \quad 3.3 \quad 2.7\ ]$$

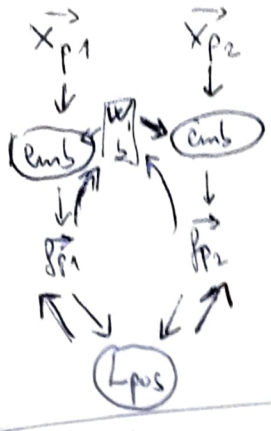
Questions:

(a) Write the equations that define the model and loss for positive and negative pairs.

(b) Execute a forward pass for the given input and calculate the loss.

(c) Determine the gradients with respect to model parameters.

2.



emb: $w = \frac{1}{3}[1\ 1\ 1]$

$b = -0.1$

$x_{p1} = [1, 1, 1.3, 1, 1, 1.3]$

$x_{p2} = [3.3, 3.3, 2.7, 3.3, 3.3, 2.7]$

---

a) $L_{pos}(x_{p1}, x_{p2}) = \|f(x_{p1}) - f(x_{p2})\|^2 = \sum_{i=1}^{d} (f_{p1,i} - f_{p2,i})^2$ — II

$L_{neg}(x_{p1}, x_{n1}) = (\max(0, -\|f_{p1} - f_{n1}\| + m))^2$

$f_{p1} = x_{p1} \circledast w^{(1)} + b^{(1)}$ // conv1d(x_{p1}, w, b) — I

$f_{p2} = x_{p2} \circledast w^{(2)} + b^{(2)}$

$L_{pos}(f_{p1}, f_{p2})$

---

b) $f_{p1} = [1\ 1\ 1.3\ 1\ 1\ 1.3] \circledast \frac{1}{3}[1\ 1\ 1] - 0.1$

$= \frac{1}{3}[3.3\ 3.3\ 3.3\ 3.3] - 0.1$

$= [1\ 1\ 1\ 1]$

$f_{p2} = [3.3\ 3.3\ 2.7\ 3.3\ 3.3\ 2.7] \circledast \frac{1}{3}[1\ 1\ 1] - 0.1$

$= \frac{1}{3}[9.9\ 9.3\ 9.3\ 9.3] - 0.1$

$= [3\ 3\ 3\ 3]$

$L_{pos}(f_{p1}, f_{p2}) = \sum_{i=1}^{4} (1-3)^2 = 16$

$\nabla_{f_{p1}} L = 2[-2\ -2\ -2\ -2]$

$= (-4)[1\ 1\ 1\ 1]$

$\nabla_{f_{p2}} L = (4)[1\ 1\ 1\ 1]$

---

c) $\nabla_{f_{p1}} L = ?$, $\nabla_{f_{p2}} L = ?$

$(\nabla_{f_{p1}} L)_k = \frac{\partial L}{\partial f_{p1,k}} = \frac{\partial}{\partial f_{p1,k}} \left(\sum_{i=1}^{4} (f_{p1,i} - f_{p2,i})^2 \right) = 2(f_{p1,k} - f_{p2,k}) \cdot (1)$

$\nabla_{f_{p1}} L = 2(\vec{f_{p1}} - \vec{f_{p2}})$

$\neq$

$\nabla_{f_{p2}} L = 2(\vec{f_{p2}} - \vec{f_{p1}})$

$(\nabla_{f_{p2}} L)_k = \frac{\partial}{\partial f_{p2,k}} \sum_{i=1}^{4} (f_{p1,i} - f_{p2,i})^2 = 2(f_{p1,k} - f_{p2,k})(-1)$

c) [cont'd]

$\nabla_{w^{(1)}} L = ?$ $\nabla_{w^{(2)}} L = ?$

$$\frac{\partial L}{\partial w_k} = \sum_{a=1}^{4} \frac{\partial L}{\partial f_{p,a}} \cdot \frac{\partial f_{p,a}}{\partial w_k} = \sum_{a=1}^{4} \left[\nabla_{f_p} L\right]_a \cdot x_{p,a+k}$$

$$- \left(\nabla_{f_j} L\right)^T \; x_{p,i+k}$$

$$f_{p,a} \sum_{k=0}^{2} x_{p,a+k} \cdot w_k$$

$$\nabla_w L = \nabla_{f_p} L \circledast x_p$$

$$\nabla_{w^{(1)}} L = [1 \;\; 1 \;\; 1.3 \;\; 1 \;\; 1 \;\; 1.3] \circledast (-4[1 \;\; 1 \;\; 1 \;\; 1])$$

$$= (-4)[4.3 \;\; 4.3 \;\; 4.6]$$

$$\nabla_{w^{(2)}} L = [3.3 \;\; 3.3 \;\; 2.7 \;\; 3.3 \;\; 3.3 \;\; 2.7] \circledast (4[1 \;\; 1 \;\; 1 \;\; 1])$$

$$= (4)[12.6 \;\; 12.6 \;\; 12]$$

$$\nabla_w L = \nabla_{w^{(1)}} L + \nabla_{w^{(2)}} L =$$

$$= (4)[8.3 \;\; 8.3 \;\; 7.4]$$

$$= [33.2 \;\; 33.2 \;\; 29.6]$$

$\nabla_{b^{(1)}} L = ?$, $\nabla_{b^{(2)}} L = ?$

$$\nabla_{b^{(1)}} L = \sum_{i=1}^{4} \left(\nabla_{f_p} L\right)_i$$

$$\nabla_{b^{(1)}} L = (-4) \cdot 4 = -16$$

$$\nabla_{b^{(2)}} L = (4) \cdot 4 = 16$$

$$\nabla_b L = \nabla_{b^{(1)}} L + \nabla_{b^{(2)}} L$$

$$= 0$$

3. Consider a complex datum X that denotes a collection of N tokens of dimension 4 that should be processed with single-head self-attention. The keys and queries both correspond to the first two dimensions of the input tokens while the values correspond to the last two dimensions of the input tokens.

   Determine the number of parameters of the layer.

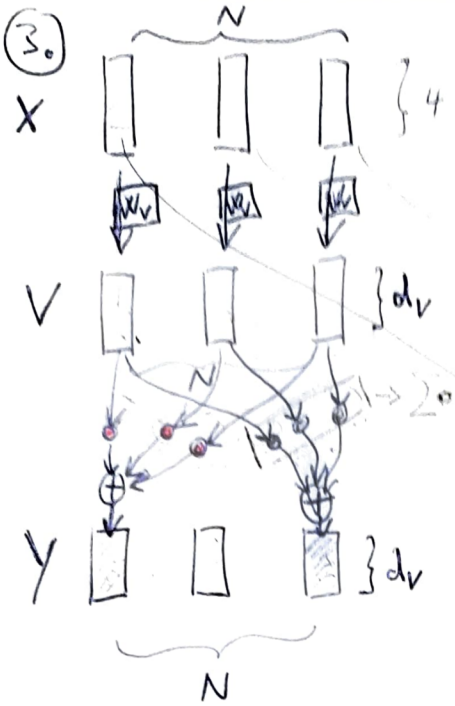
   Propose an implementation in numpy to recover the result `Xatt` consisting of:

   (a) implementation of the function `self_attn(Q, K, V)`

   (b) definition of the projection matrices Wq, Wk, Wv

   (c) correct invocation of the function `self_attn` that solves the task

(3.)



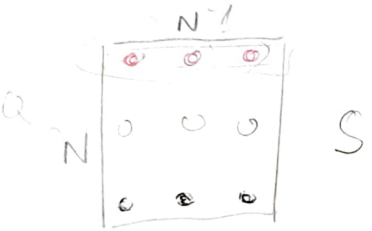
$W_V \rightarrow [d_{in}, d_V]$

$X \rightarrow [N, d_{in}]$

$V \rightarrow [N, d_V]$

W_Q W_K

$$V = X W_V$$

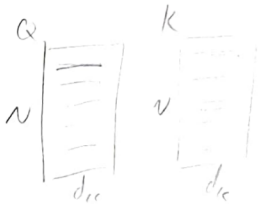


$[N, d_K]$

$$Q = X W_Q$$

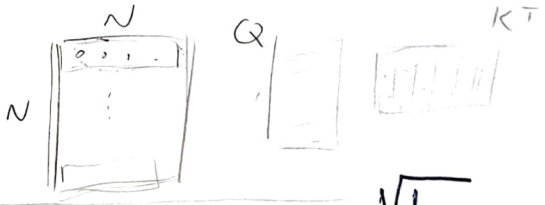
$[N, d_K]$

$$K = X W_K$$



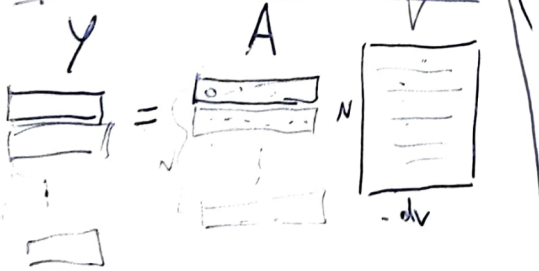
$$A = \text{Softmax}(S, \text{axis}=1)$$

(dim)



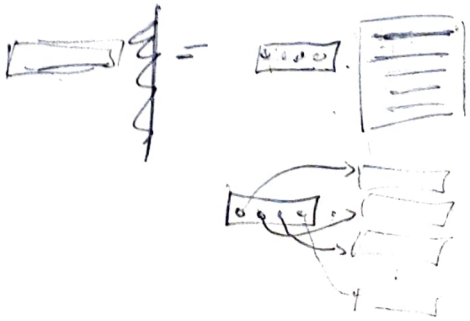
$$S = Q K^T / (\ldots) \quad \rightarrow \sqrt{d_K}$$

$$A = \text{softmax}(S, \text{axis}=1)$$



$N, d_V \quad N, N \quad N, d_V$

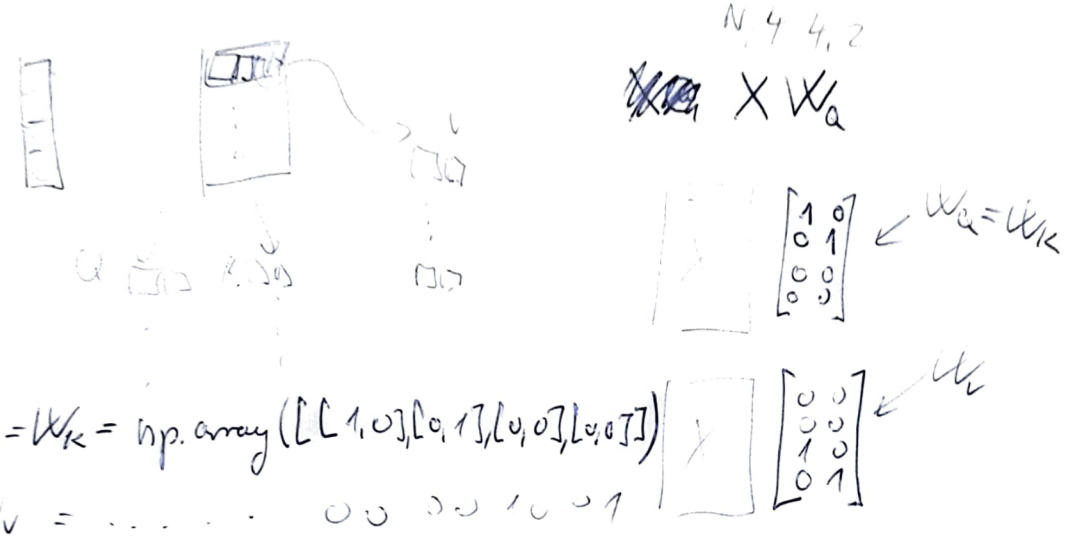
$$Y = AV$$



a) def self_attn(Q, K, V):

```
S = (Q @ K.T) / np.sqrt(Q.shape[1])
A = sp.special.softmax(S, axis=1)
return A @ V
```

b)



c) X_{ATT} = self_attn(X@W_Q , X@W_K , X@W_V)

# of params:

$W_Q \rightarrow 4 \cdot 2$

$W_K \rightarrow 4 \cdot 2$

$W_V \rightarrow 4 \cdot 2$

} 24

4. We consider siamese metric learning with triplet loss. There are 7 examples in the train set. Matrix $\boldsymbol{E}$ contains their embeddings and semantic classes. Each row of the matrix corresponds to one example, with the embedding given by the first two elements, and the third element designating the class.

$$\boldsymbol{E} = \left[\begin{array}{cc|c} -1 & 4 & 0 \\ -1 & 2 & 0 \\ 3 & -3 & 1 \\ 1 & -1 & 1 \\ 1 & 4 & 2 \\ 4 & 1 & 2 \\ 1 & 1 & 2 \end{array}\right]$$

Tasks:

(a) Define the L2 distance between two vectors $d_{\mathrm{L2}}(\boldsymbol{x}, \boldsymbol{z})$. Fill in the missing values in the following definition of triplet loss if you are given the embeddings of the anchor $\boldsymbol{a}$, the positive example $\boldsymbol{p}$, and the negative example $\boldsymbol{n}$.
$L(\boldsymbol{a}, \boldsymbol{p}, \boldsymbol{n}) = max(d_{\mathrm{L2}}(\_\_, \_\_)^2 - d_{\mathrm{L2}}(\_\_, \_\_)^2 + m, \_\_)$

(b) Classify the embedding $\boldsymbol{z} = [0, 2]$ using the training set embeddings by comparing with class centroids as in Lab 4. Show the calculation procedure.

(c) We consider using the learned metric embedding model as part of an image retrieval system. For any given query, the system returns the **five** most similar examples, according to the inverse L1 distance. Let the input query to the system be the embedding $\boldsymbol{q} = [1, 0]$, which belong to class $y = 2$. Which examples would be retrieved by the system? What would be the precision (P) and recall (R) for such a response.

5. Your task is to use a vanilla recurrent network with a ReLU nonlinearity for the task of tracking a users' bank account balance. You will use a 3-dimensional hidden state representation, where the first element will track the total amount of money available for withdrawal (account balance + the allowed overdraft), the second element will store the allowed overdraft, while the third element will serve as a flag checking whether the account is in negative balance (0 if the account **is** in negative balance, any positive value otherwise). The case where a user exceeds the allowed overdraft is currently not of concern.

   The initial hidden state of a test users' account is:

$$h^{(0)} = \begin{bmatrix} 10000 \\ 5000 \\ 765 \end{bmatrix}$$

   The 2-dimensional input vectors will contain the amount of money added to the account on the first element, and the amount of money withdrawn on the second element.

   (a) Determine the values of the remaining parameters of the recurrent neural network (W_{hh}, W_{xh}, b_h) so that the network would function as intended

   (b) Run the forward pass of the network with the determined parameters for the following inputs:

$$x^{(1)} = \begin{bmatrix} 1000 \\ 0 \end{bmatrix} \qquad x^{(2)} = \begin{bmatrix} 1000 \\ 3500 \end{bmatrix} \qquad x^{(3)} = \begin{bmatrix} 0 \\ 4000 \end{bmatrix}$$

   (c) Consider using an LSTM network for the same problem, which has the same input representation dimension and hidden state dimension. Determine the total number of parameters of such a network.