Problem 1 (Feed Forward Network).

Consider the following network in Figure 1, which uses sigmoid activations, you are given the weight matrices

\[
W_h = \begin{bmatrix} 0.2 & 0.1 \\ 0.4 & -0.1 \end{bmatrix}, \quad W_o = \begin{bmatrix} -0.2 & 0.1 \\ -0.1 & 0.1 \end{bmatrix}, \quad b_h = \begin{bmatrix} -0.4 \\ -0.2 \end{bmatrix}, \quad b_o = \begin{bmatrix} 0.1 \\ 0.4 \end{bmatrix}
\]

(1.1)

(a) Compute the outputs of hidden and output layers in the forward pass, given the input \(x^T = (1, 0)\).

(b) Using the previous results, compute updated weights for the output layer in the backward pass. Use loss function \(L = |y_{true} - y_{pred}|^2\), the learning rate \(\eta = 0.1\) and true output \(y_{true} = (0.9, 0.1)\).

(c) What would be a better alternative to Euclidean loss here and why?

Problem 2 (Backpropagation).

Consider a neural network, which computes the function \(F: \mathbb{R}^d \to \mathbb{R}\)

\[
F(u_1, ..., u_d) = f_1 \circ f_{l-1} \circ ... \circ f_1(u_1, ..., u_d)
\]

(2.1)

where each layer computes \(f_j: \mathbb{R}^{n_j} \to \mathbb{R}^{n_{j+1}}\) and \(n_j\) - number of neurons on \(j\)th layer.

(a) Describe forward and backward passes of backpropagation, which computes \(\nabla F(x_1, ..., x_d)\) (in steps or pseudocode). Assume you use the memory to store intermediate values (like partial derivatives to avoid computing them twice).

(b) Suppose \(u^j\) is the output of \((j-1)\)-th layer, show the weights update for the parameters of this layer \(W^j\) using the outputs of your algorithm.

(c) What is the computational complexity of your algorithm in terms of time and memory? What if you did not store the intermediate values?

Problem 3 (Deep Relevance Matching Model). You are given a query "buy iphone". Compute the score of the document "cheap clothes online store" using DRMM.

(a) i You have the similarity matrix with similarities in Table 1. Compute the count-based matching histogram using 5 bins.

ii What are the advantages of the matching histogram over using plain similarity scores?

iii The experiments show that LogCount-based histogram performs best and Normalized by size histogram performs worst. What do you think can be the reasons for that?

<table>
<thead>
<tr>
<th></th>
<th>cheap</th>
<th>clothes</th>
<th>online</th>
<th>store</th>
</tr>
</thead>
<tbody>
<tr>
<td>buy</td>
<td>0.4</td>
<td>0.2</td>
<td>-0.2</td>
<td>0.6</td>
</tr>
<tr>
<td>iphone</td>
<td>-0.8</td>
<td>0.2</td>
<td>-0.2</td>
<td>0.6</td>
</tr>
</tbody>
</table>

Table 1: Similarity matrix
(b) Perform the calculation of the feed-forward matching network, suppose you have just one layer with \( \tanh \) activation computing \( z_i = \tanh(Wh_i) \), where \( i \) - index of the query term, \( h_i \) - the histogram for the \( i \)th term and \( W \) - network weights per each bin in the histogram (we set \( b = 0 \) for simplicity). Propose reasonable network weights \( W \) for all bins (justify your choice) and calculate \( z_i \) for all query terms. For picking the weights, assume that the considered documents are not longer than 10 terms.

(c) i Calculate the final score for the document, given term weights \( g("buy") = 0.9 \) and \( g("iphone") = 0.06 \).

ii The experiments show that using IDF to learn term weights gives better results than word embeddings. Share you insights what can be the cause of it.

Problem 4 (CNNs).

(a) Give an example of an NLP task where a feed forward architecture will be more appropriate than CNN and vice versa, when CNN will be more appropriate than FF.

(b) Suppose an input is the 1D of shape 16 \( \times \) 1 and we are using CNN with no pooling and convolutional kernels of size 6 \( \times \) 1 in each layer, applying them with stride 2 or stride 1 (give answers for both strides). How many layers can there be if we:

- Do not use zero-padding (make a sketch for at least one stride)
- Use zero-padding

Discuss the results: why is zero-padding good/bad?