Assignment 3

Mahdi Tabatabaei 400101515
Github Repository

Deep Learning

Dr. Fatemizadeh

December 30, 2024

Deep Learning

Assignment 3

Mahdi Tabatabaei 400101515
Github Repository

Question 1 (40 Points)

Determine what the following network computes. (To be more precise, determine the output function calculated by the unit at the final time step.) The size of other outputs is not important. All biases are zero. Assume the inputs are integers, and the length of the input sequence is even. Also, consider the activation function to be a sigmoid.

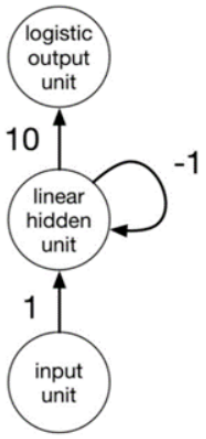


Figure 1: Network of Question 1

Soloution

We aim to determine the output y_T at the final time step T of the given recurrent neural network (RNN).
The hidden state at each time step t is updated as:

$$h_t = -h_{t-1} + x_t,$$

where x_t is the input at time step t, and h_0 is the initial hidden state.
The output at each time step is:

$$y_t = \sigma(10 \cdot h_t),$$

where $\sigma(z)$ is the sigmoid activation function: Using the recurrence relation for h_t, we expand it step by step:

$$\begin{aligned}
h_1 &= -h_0 + x_1, \\
h_2 &= -h_1 + x_2 = -(-h_0 + x_1) + x_2 = h_0 - x_1 + x_2, \\
h_3 &= -h_2 + x_3 = -(h_0 - x_1 + x_2) + x_3 = -h_0 + x_1 - x_2 + x_3.
\end{aligned}$$

From this pattern, the general form is:

$$h_t = (-1)^t h_0 + \sum_{i=1}^{t} (-1)^{t-i} x_i.$$

At the final time step T, the output is:

$$y_T = \sigma(10 \cdot h_T),$$

The final output at time T is:

$$y_T = \sigma\left(10 \cdot \left((-1)^T h_0 + \sum_{i=1}^{T} (-1)^{T-i} x_i\right)\right),$$

For even T:

$$y_T = \sigma\left(10 \cdot \left(h_0 + \sum_{i=1}^{T} (-1)^{T-i} x_i\right)\right),$$

Question 2 (40 Points)

Mention two issues that exist when using the one-hot vector for word representation.

Soloution

1. High Dimensionality and Sparsity:
One-hot vectors are extremely high-dimensional, where the size of the vector equals the vocabulary size. These vectors are **sparse** (mostly zeros), which makes computation and storage very inefficient, especially as the vocabulary grows.

2. No Similarity Information:
One-hot encoding does not capture any relationships or similarity between words. All words are treated as orthogonal (perpendicular) to each other, even if they are semantically similar. This means there is no information about relationships between words like "Deep" and "Network" (which are related concepts), as they are represented as equally distant as any unrelated words.

3. Hard-Coded Representation:
Adding a new word requires introducing a new dimension to the vector. This makes the representation rigid and difficult to scale for large or dynamic vocabularies.

Question 3 (110 Points)

Given the following recurrent neural network diagram, answer the questions below. Note that for simplicity, all values (inputs, weights, and outputs) are scalar values. Also, assume that all activation functions are sigmoid functions.

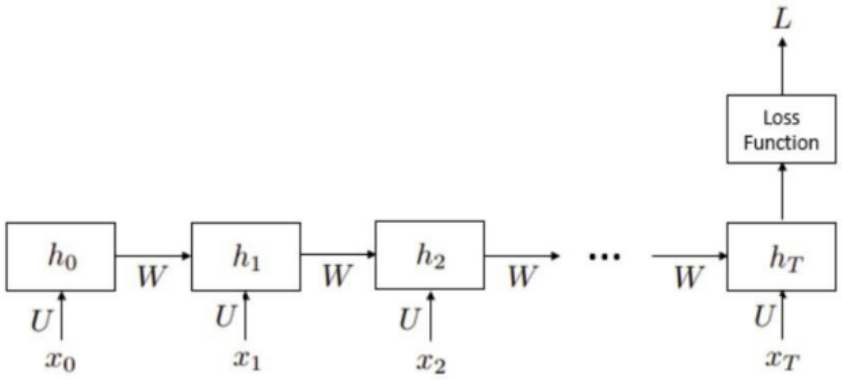


Figure 2: Network Architecture

(a) Write the gradient of h_t with respect to h_{t+1}, i.e., $\frac{\partial L}{\partial h_t}$ in terms of $\frac{\partial L}{\partial h_{t+1}}$ $(1 \leq t \leq T-1)$.

Soloution

Given the recurrent neural network (RNN), the hidden state h_t is computed as:

$$h_t = \sigma(Wh_{t-1} + Ux_t),$$

At time step $t+1$, the hidden state h_{t+1} depends on h_t through the relation:

$$h_{t+1} = \sigma(Wh_t + Ux_{t+1}).$$

The gradient of the loss L with respect to h_t can be computed using the chain rule:

$$\frac{\partial L}{\partial h_t} = \frac{\partial L}{\partial h_{t+1}} \cdot \frac{\partial h_{t+1}}{\partial h_t}$$

From the recurrence relation $h_{t+1} = \sigma(Wh_t + Ux_{t+1})$, the partial derivative of h_{t+1} with respect to h_t is:

$$\frac{\partial h_{t+1}}{\partial h_t} = diag(\sigma'(Wh_t + Ux_{t+1})) \cdot W,$$

where σ' is the derivative of the sigmoid activation function, and it is given by:

$$\sigma'(z) = \sigma(z) \cdot (1 - \sigma(z)).$$

Substituting $\frac{\partial h_{t+1}}{\partial h_t}$ into the chain rule, we obtain:

$$\frac{\partial L}{\partial h_t} = \frac{\partial L}{\partial h_{t+1}} \cdot diag(\sigma'(Wh_t + Ux_{t+1})) \cdot W,$$

$$\frac{\partial L}{\partial h_t} = \frac{\partial L}{\partial h_{t+1}} \cdot diag(\sigma(Wh_t + Ux_{t+1}) \odot (\mathbf{1} - \sigma(Wh_t + Ux_{t+1}))) \cdot W,$$

which $\odot$ is point-wise product.

(b) Using the relationship from part (a), write the gradient h_t in a chain-like form with respect to the gradient h_T.

Soloution

From part (a), the partial derivative $\frac{\partial h_{t+1}}{\partial h_t}$ is given as:

$$\frac{\partial h_{t+1}}{\partial h_t} = \text{diag}(\sigma'(Wh_t + Ux_{t+1})) \cdot W,$$

By applying the chain rule recursively, we can write $\frac{\partial L}{\partial h_t}$ in terms of $\frac{\partial L}{\partial h_T}$ as:

$$\frac{\partial L}{\partial h_t} = \frac{\partial L}{\partial h_T} \prod_{k=t}^{T-1} \frac{\partial h_{k+1}}{\partial h_k}.$$

Substitute $\frac{\partial h_{k+1}}{\partial h_k}$ into the equation:

$$\frac{\partial L}{\partial h_t} = \frac{\partial L}{\partial h_T} \prod_{k=t}^{T-1} \left(\text{diag}(\sigma'(Wh_k + Ux_{k+1})) \cdot W\right).$$

We now want to describe and analyze methods for preventing the vanishing and exploding gradient problems.

(a) One important method for preventing the vanishing and exploding gradients is to initialize the network weights correctly. Describe how to set the maximum value of W such that the gradients neither vanish nor explode. (Hint: Find the upper bound for the gradient h_t.)

Soloution

$$\frac{\partial L}{\partial h_t} = \frac{\partial L}{\partial h_T} \prod_{k=t}^{T-1} \left(\text{diag}(\sigma'(Wh_k + Ux_{k+1})) \cdot W\right) = \frac{\partial L}{\partial h_T} \cdot W^T \cdot \prod_{k=t}^{T-1} \left(\sigma'(Wh_k + Ux_{k+1})\right).$$

To maximize $\sigma'(z)$, note that $\sigma(z)$ is always in the range $[0, 1]$. Let $y = \sigma(z)$. Then:

$$\sigma'(z) = y(1 - y).$$

We now maximize $y(1 - y)$. This is a standard quadratic expression, and its maximum occurs when $y = \frac{1}{2}$. Substituting $y = \frac{1}{2}$:

$$max(\sigma'(z)) = \frac{1}{2} \cdot \left(1 - \frac{1}{2}\right) = \frac{1}{4}.$$

The absolute value of the expression is:

$$\left|\frac{\partial L}{\partial h_T}\cdot W^T\cdot\prod_{k=t}^{T-1}(\sigma'(Wh_k+Ux_{k+1}))\right|\leq\left|\frac{\partial L}{\partial h_T}\right|\cdot\|W^T\|\cdot\prod_{k=t}^{T-1}|\sigma'(Wh_k+Ux_{k+1})|\,,$$

$$|\sigma'(Wh_k+Ux_{k+1})|\leq\frac{1}{4}.$$

For the product over $(T-t)$ terms:

$$\left|\frac{\partial L}{\partial h_T}\right|\cdot\|W^T\|\cdot\prod_{k=t}^{T-1}|\sigma'(Wh_k+Ux_{k+1})|\leq\left|\frac{\partial L}{\partial h_T}\right|\cdot\|W^T\|\cdot\left(\frac{1}{4}\right)^{T-t}.$$

$$\frac{\partial L}{\partial h_t}\leq\left|\frac{\partial L}{\partial h_T}\right|\cdot\|W\|\cdot\left(\frac{1}{4}\right)^{T-t}.$$

(b) Another method to prevent the vanishing gradient problem is to use skip-connections. The following diagram shows that at each time step t, h_t is connected to h_{t+2} as well as h_{t+1}. Now, rewrite the gradient of h_t with respect to h_{t+2} and explain why this approach effectively reduces the vanishing gradient problem $(1\leq t\leq T-2)$.

Soloution

$$h_t=\sigma(Wh_{t-1}+Mh_{t-2}+Ux_t),$$

$$h_{t+1}=\sigma(Wh_t+Mh_{t-1}+Ux_{t+1}).$$

$$h_{t+2}=\sigma(Wh_{t+1}+Mh_t+Ux_{t+2}).$$

The gradient of the loss L with respect to h_t can be computed using the chain rule:

$$\frac{\partial L}{\partial h_t}=\frac{\partial L}{\partial h_{t+2}}\cdot\frac{\partial h_{t+2}}{\partial h_{t+1}}\cdot\frac{\partial h_{t+1}}{\partial h_t}$$

$$\frac{\partial h_{t+1}}{\partial h_t}=diag(\sigma'(Wh_t+Mh_{t-1}+Ux_{t+1}))\cdot W,$$

$$\frac{\partial h_{t+2}}{\partial h_{t+1}}=diag(\sigma'(Wh_t+Mh_{t-2}+Ux_{t+1}))\cdot W,$$

$$\frac{\partial L}{\partial h_t}=\frac{\partial L}{\partial h_{t+2}}\cdot(diag(\sigma'(Wh_t+Mh_{t-2}+Ux_{t+1}))\cdot W)^2$$

Skip connections mitigate gradient vanishing by providing a direct path for gradients to flow back, bypassing intermediate layers. In networks without skip connections, gradients diminish as they propagate through layers due to repeated multiplication by small derivatives. With skip connections, the output of a layer becomes $x_i=f(x_{i-1},W_i)+x_{i-1}$, and the gradient is $\frac{\partial\mathcal{L}}{\partial x_{i-1}}=\frac{\partial\mathcal{L}}{\partial x_i}\cdot\frac{\partial f}{\partial x_{i-1}}+\frac{\partial\mathcal{L}}{\partial x_i}$, ensuring the gradient remains strong.

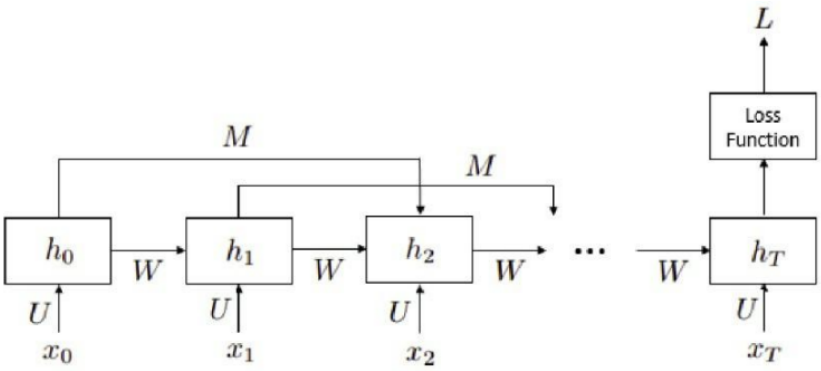


Figure 3: Network Architecture

(c) One of the ways to prevent gradient explosion is gradient clipping. This can be done in two ways: clipping by value and clipping by norm. Explain these two methods separately. Also, explain the advantages of clipping by norm over clipping by value.

Soloution

Clipping by Value: In this method, if any gradient component exceeds a pre-specified threshold value (either positive or negative), it is clipped to that threshold. For example, if the threshold is set to 1.0, any gradient greater than 1.0 is set to 1.0, and any gradient smaller than -1.0 is set to -1.0. This method is simple and ensures no single gradient component becomes excessively large.

Clipping by Norm: This method involves normalizing the entire gradient vector if its norm exceeds a pre-specified threshold. Specifically, if the norm of the gradient vector is larger than the threshold, the vector is scaled down so that its norm matches the threshold. This ensures that the overall gradient magnitude is controlled while preserving the direction of the gradient vector.

Advantages of Clipping by Norm:

- **Preserves Gradient Direction:** Clipping by value affects each gradient component individually, which can distort the gradient's direction. Clipping by norm scales the gradient as a whole, preserving its original direction.
- **Better for Optimization:** Since optimization often relies on the gradient direction rather than its absolute value, clipping by norm is generally more effective for stable convergence.
- **Uniform Scaling:** Clipping by norm ensures that all gradient components are scaled proportionally, avoiding biases introduced by selective clipping in clipping by value.

Question 4 (70 Points)

In this question, we want to familiarize ourselves with concepts in sequence-to-sequence (Seq2Seq) models, their advantages, and their disadvantages. In this question, we will study the concept of **teacher forcing**. To generate a sequence, we can consider a raw strategy where we generate token $t+1$ at time $t+1$ by encoding time t output as the input at time $t+1$. However, this has some problems.

(a) First, explain what these problems are, and then explain the **teacher forcing** method and how it resolves these issues.

Soloution

The problem with the basic approach of generating sequences is that the model uses its own output at each step as the input for the next step during testing (this is called *autoregressive generation*). If the model makes a mistake early on, that mistake can grow larger as the sequence progresses because future steps are based on the incorrect outputs. This can lead to poor overall performance.
Teacher Forcing is a method to address this problem during training. Instead of using the model's own predictions as inputs for the next steps, it uses the actual correct outputs (ground truth) from the training data. This helps the model learn more effectively by always being guided by the correct context, avoiding the buildup of errors during training.

(b) The main problem of **teacher forcing** is **exposure bias**. Explain this issue.

Soloution

The main problem with **Teacher Forcing** is something called **exposure bias**. During training, the model always uses the correct output (ground truth) from the training data as input for the next step. However, during testing, the model does not have access to these correct outputs and must rely on its own predictions. This creates a mismatch between the training and testing environments.
If the model makes a small mistake during testing, it may not know how to recover from it, because it was never trained to handle its own incorrect predictions. This leads to a cascading effect where errors accumulate as the sequence progresses.

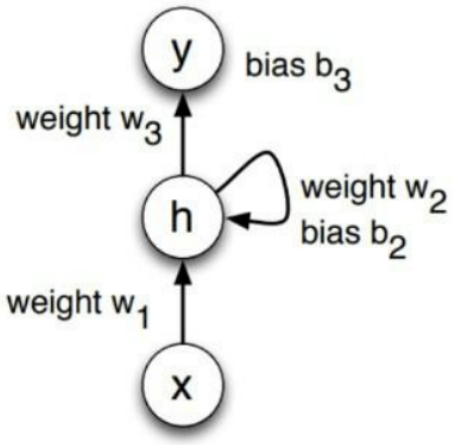
(c) One of the solutions to the **exposure bias** issue is a technique called **scheduled sampling**. Explain this technique and describe how it reduces the effect of exposure bias.

Soloution

One way to address the **exposure bias** problem is by using a technique called **Scheduled Sampling**. In this method, during training, the model is gradually exposed to its own predictions instead of always relying on the ground truth. As training progresses, there is an increasing probability that the model will use its own predictions instead of the ground truth. This gradual transition helps the model learn how to handle its own errors and prepares it for the testing phase, where it must rely solely on its own predictions. By doing this, Scheduled Sampling reduces the mismatch between training and testing environments, thereby mitigating the effects of exposure bias.

Question 5 (70 Points)

Consider the recurrent network shown below. Set the weights and biases in such a way that the network output remains 1 as long as the input sequence contains ones, and the network output changes to 0 when the input changes to 0. For example, the network output for the input sequence $x = 1111010000$ is $y = 1111000000$.



Soloution

The problem involves a Recurrent Neural Network (RNN) that processes a binary input sequence x_t and outputs y_t, where:

$$y_t = 1 \quad \text{if all prior inputs (including the current one) are 1,}$$

$$y_t = 0 \quad \text{otherwise .}$$

The hidden state h_t is defined as:

$$h_t = h_{t-1} \cdot x_t,$$

where:

$$h_t = \begin{cases} 1 & \text{if all inputs up to time } t \text{ are 1,} \\ 0 & \text{otherwise.} \end{cases}$$

The output is defined as:

$$y_t = h_t.$$

At $t = 0$, the initial hidden state is:

$$h_0 = 1.$$

The hidden state update equation can be expressed as:

$$h_t = \sigma(W_2 h_{t-1} + W_1 x_t + b_2),$$

where $\sigma(z)$ is:

$$\sigma(z) = \begin{cases} 1 & \text{if } z \geq 0.5, \\ 0 & \text{otherwise.} \end{cases}$$

The output equation is:

$$y_t = W_3 h_t + b_3.$$

To meet the problem's requirements:

- $W_h = 1$: The hidden state from the previous step contributes fully.
- $W_x = 1$: The current input contributes fully.
- $b_2 = -1.5$: Ensures that $h_t = 1$ only when $h_{t-1} = 1$ and $x_t = 1$.
- $W_y = 1$: The hidden state directly determines the output.
- $b_y = 0$: No additional bias is needed for the output.

The following table shows the computation of h_t and y_t for $x_t = 111101$:

Time Step	x_t	$h_t = h_{t-1} \cdot x_t$	$y_t = h_t$
1	1	$1 \cdot 1 = 1$	1
2	1	$1 \cdot 1 = 1$	1
3	1	$1 \cdot 1 = 1$	1
4	1	$1 \cdot 1 = 1$	1
5	0	$1 \cdot 0 = 0$	0
6	1	$0 \cdot 1 = 0$	0

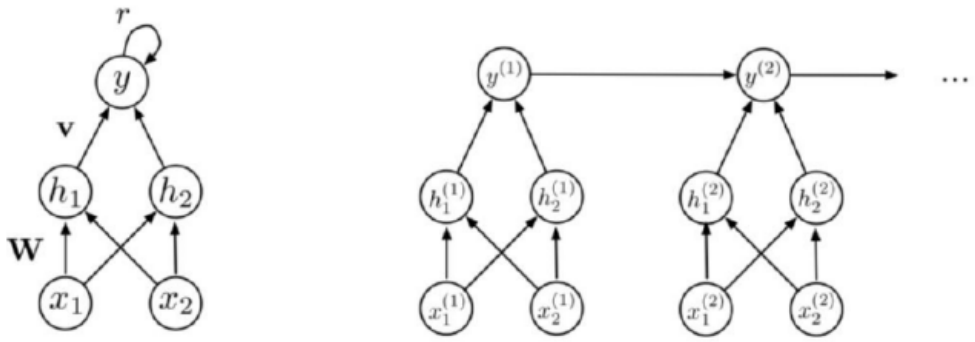
The output sequence is:

$$y = 111100.$$

This configuration satisfies the problem requirements.

Question 6 (70 Points)

Consider the following recurrent network. Assume that this network receives two sequences of zeros and ones and returns the number 1 if the two sequences are equal, and the number 0 otherwise.



The matrix W is a 2×2 matrix, and $\mathbf{v}$, $\mathbf{b}$ are 2-dimensional vectors, and c, r, and c_0 are scalars. Determine these parameters so that the network operates as described. (Hint: $y^{(t)}$, the output at time t, indicates whether the two sequences have been equal up to that point. The first hidden layer indicates whether the two inputs at time t were both zero, and the second hidden layer indicates whether the two inputs at time t were both one.)

$$h^{(t)} = \phi\left(Wx^{(t)} + \mathbf{b}\right)$$

$$y^{(t)} = \begin{cases} \phi\left(\mathbf{v}^\top h^{(t)} + ry^{(t-1)} + c\right) & \text{for } t > 1 \\ \phi\left(\mathbf{v}^\top h^{(t)} + c_0\right) & \text{for } t = 1 \end{cases}$$

$$\phi(z) = \begin{cases} 1 & \text{if } z > 0 \\ 0 & \text{if } z \leq 0 \end{cases}$$

Soloution

The inputs are two binary sequences, denoted by $x_1^{(t)}$ and $x_2^{(t)}$, where $x_1^{(t)}$ and $x_2^{(t)}$ represent the values of the two sequences at time t. The output $y^{(t)}$ indicates whether the sequences have been identical up to time t:

$$y^{(t)} = \begin{cases} 1 & \text{if the two sequences are identical up to time } t, \\ 0 & \text{otherwise.} \end{cases}$$

The hidden layers $h^{(t)}$ are defined as follows:

$$h_1^{(t)} = 1 \quad \text{if } x_1^{(t)} = 0 \text{ and } x_2^{(t)} = 0, \quad \text{otherwise } h_1^{(t)} = 0.$$

$$h_2^{(t)} = 1 \quad \text{if } x_1^{(t)} = 1 \text{ and } x_2^{(t)} = 1, \quad \text{otherwise } h_2^{(t)} = 0.$$

The activations for the hidden layer are determined by:

$$h^{(t)} = \phi(Wx^{(t)} + \mathbf{b}),$$

where $x^{(t)} = \begin{bmatrix} x_1^{(t)} \\ x_2^{(t)} \end{bmatrix}$, W is a 2×2 matrix, and $\mathbf{b}$ is a 2-dimensional vector.

We define:

$$W = \begin{bmatrix} -1 & -1 \\ 1 & 1 \end{bmatrix}, \quad \mathbf{b} = \begin{bmatrix} 1 \\ -1 \end{bmatrix}.$$

For $h_1^{(t)}$, the input $Wx^{(t)} + \mathbf{b} = -x_1^{(t)} - x_2^{(t)} + 1$ is positive only when $x_1^{(t)} = 0$ and $x_2^{(t)} = 0$.

For $h_2^{(t)}$, the input $Wx^{(t)} + \mathbf{b} = x_1^{(t)} + x_2^{(t)} - 1$ is positive only when $x_1^{(t)} = 1$ and $x_2^{(t)} = 1$.

The output layer $y^{(t)}$ depends on $y^{(t-1)}$, the previous output, and $h^{(t)}$, the current hidden layer activations. We define:

$$y^{(t)} = \begin{cases} \phi\left(\mathbf{v}^\top h^{(t)} + r y^{(t-1)} + c\right) & \text{for } t > 1, \\ \phi\left(\mathbf{v}^\top h^{(t)} + c_0\right) & \text{for } t = 1. \end{cases}$$

The parameters for the output layer are chosen as follows:

$$\mathbf{v} = \begin{bmatrix} 1 \\ 1 \end{bmatrix}, \quad r = 1, \quad c = -1, \quad c_0 = 0.$$

The step function $\phi(z)$ is defined as:

$$\phi(z) = \begin{cases} 1 & \text{if } z > 0, \\ 0 & \text{if } z \leq 0. \end{cases}$$

These parameters ensure that:

- The hidden layer $h^{(t)}$ correctly identifies whether $x_1^{(t)} = x_2^{(t)}$.
- The output $y^{(t)}$ propagates the result of the comparison across time steps, starting from $t = 1$.

The final parameters are:

$$W = \begin{bmatrix} -1 & -1 \\ 1 & 1 \end{bmatrix}, \quad \mathbf{b} = \begin{bmatrix} 1 \\ -1 \end{bmatrix}, \quad \mathbf{v} = \begin{bmatrix} 1 \\ 1 \end{bmatrix},$$

$$r = 1, \quad c = -1, \quad c_0 = 0.$$

This configuration ensures that the network outputs 1 if the two sequences are identical and 0 otherwise.