Assignment 4

Ehsan Merrikhi 400101967
github repository

Deep Learning

Dr. Fatemizadeh

December 22, 2024

Deep Learning

Assignment 4

Ehsan Merrikhi 400101967
github repository

Contents

Question 1 (40 Points)

Determine what the following network computes. More precisely, determine the function calculated by the output unit at the final time step. The sizes of the outputs do not matter. All biases are zero. Assume that the inputs are integers and the length of the input sequence is even. Also, consider the activation function to be the sigmoid function.

Soloution

Input Unit and Linear Hidden Unit

Let the input sequence be $x_1, x_2, \ldots, x_T$, where T is the length of the sequence. Denote the state of the linear hidden unit at time step t as h_t.

$$h_t = x_t - h_{t-1}.$$

Assume $h_0 = 0$ (initial state of the hidden unit).

$$h_1 = x_1 - 0 = x_1, \quad h_2 = x_2 - h_1 = x_2 - x_1$$

We can generalize:

$$h_T = x_T - x_{T-1} + x_{T-2} - \cdots \pm x_1.$$

Logistic Output Unit

At the final time step, the output of the logistic unit is:

$$y = \sigma(10 \cdot h_T) \xrightarrow{\text{Substituting } h_T} \sigma\left(10 \cdot (x_T - x_{T-1} + x_{T-2} - \cdots \pm x_1)\right).$$

Final Computation

The network computes a sigmoid-transformed, weighted sum of the alternating sum of the input sequence:

$$y = \frac{1}{1 + e^{-10 \cdot (x_T - x_{T-1} + x_{T-2} - \cdots \pm x_1)}}.$$

This is the function calculated by the output unit at the final time step.

Question 2 (40 Points)

List two of the problems that exist when using one-hot vectors to represent words.

Soloution

High Dimensionality

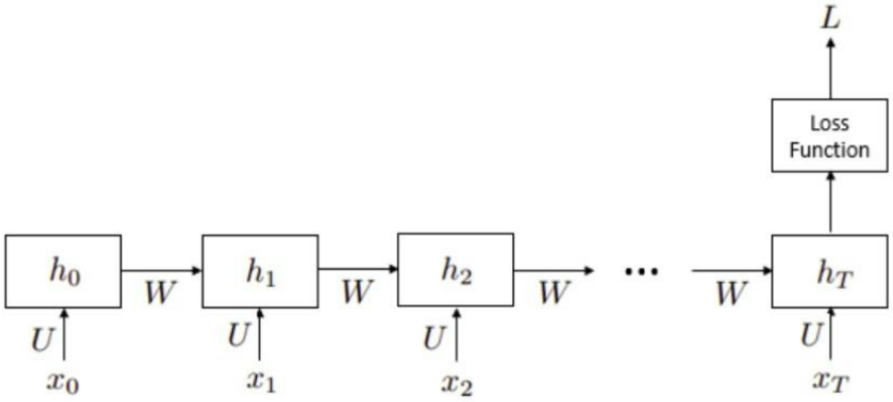
One-hot vectors require a dimensionality equal to the size of the vocabulary. For large vocabularies, this leads to very high-dimensional vectors, which can be computationally inefficient and consume significant memory resources.

Lack of Semantic Information

One-hot encoding does not capture any semantic relationships between words. All words are represented as orthogonal vectors, meaning that similar words do not have similar representations. This limitation hinders the model's ability to understand and leverage the meanings and relationships between different words.

Question 3 (110 Points)

Given the following recurrent neural network, answer the questions. Note that for simplicity, all values—meaning the inputs, weights, and outputs—are numerical. Also, assume that all activation functions are sigmoid functions.



Part 1.

First, express $\frac{\partial l}{\partial h_t}$ in terms of $\frac{\partial l}{\partial h_{t+1}}$.

Soloution

To express the gradient $\frac{\partial l}{\partial h_t}$ in terms of $\frac{\partial l}{\partial h_{t+1}}$, we utilize the chain rule within the framework of a recurrent neural network (RNN).
Assume the hidden state at time $t+1$ is defined as:

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

The loss l depends on all hidden states h_t. To determine $\frac{\partial l}{\partial h_t}$, we consider both its direct and indirect dependencies through h_{t+1}.
Applying the chain rule:

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

$$\frac{\partial h_{t+1}}{\partial h_t} = W^T \cdot \sigma'(a_t)$$

where $a_t = Wh_t + Ux_{t+1}$ and $\sigma'(a_t)$ is the derivative of the sigmoid function evaluated at a_t.
Substituting back into the chain rule:

$$\frac{\partial l}{\partial h_t} = \frac{\partial l}{\partial h_{t+1}} \cdot W^T \cdot \sigma'(a_t)$$

Therefore, the gradient $\frac{\partial l}{\partial h_t}$ is expressed in terms of $\frac{\partial l}{\partial h_{t+1}}$ as follows:

$$\boxed{\frac{\partial l}{\partial h_t} = \frac{\partial l}{\partial h_{t+1}} \cdot W^T \cdot \sigma'(a_t)}$$

Part 2.

Now, using the relationship from the previous section, write the gradient of h_0 in terms of the gradient of h_T in a chain-like manner. Next, we aim to introduce and analyze methods to prevent gradient vanishing and exploding.

Soloution

From the previous subsection, we have:

$$\frac{\partial l}{\partial h_t} = \frac{\partial l}{\partial h_{t+1}} \cdot W^T \cdot \sigma'(a_t)$$

where $a_t = Wh_t + Ux_{t+1}$.

Expanding the Gradient from h_T to h_0

Apply the recursive relationship iteratively from $t = T - 1$ down to $t = 0$:

$$\frac{\partial l}{\partial h_0} = \frac{\partial l}{\partial h_1} \cdot W^T \cdot \sigma'(a_0)$$

$$\frac{\partial l}{\partial h_1} = \frac{\partial l}{\partial h_2} \cdot W^T \cdot \sigma'(a_1)$$

$$\vdots$$

$$\frac{\partial l}{\partial h_{T-1}} = \frac{\partial l}{\partial h_T} \cdot W^T \cdot \sigma'(a_{T-1})$$

$$\implies \frac{\partial l}{\partial h_0} = \frac{\partial l}{\partial h_T} \cdot \prod_{k=0}^{T-1} \left(W^T \cdot \sigma'(a_k)\right)$$

Interpretation of the Gradient Expression

1. **Gradient Vanishing:** If the eigenvalues of $W^T \cdot \sigma'(a_k)$ are less than 1, the gradients diminish exponentially as they propagate back through time, making it difficult for the network to learn long-range dependencies.

2. **Gradient Exploding:** Conversely, if the eigenvalues are greater than 1, the gradients can grow exponentially, causing instability in the training process.

Methods to Prevent Gradient Vanishing and Exploding

To mitigate these issues, several strategies can be employed:

1. **Gradient Clipping:** Prevents large gradients by limitting gradient size.

2. **Advanced Architectures (LSTM/GRU):** Use gating to stabilize gradients.

3. **Proper Weight Initialization:** Helps Maintains gradient scale.

4. **Non-saturating Activation Functions (e.g., ReLU):** Preserves gradient strength.

5. **Regularization Techniques (e.g., Dropout):** Enhances training stability.

Part 3.

One of the important methods to prevent gradient vanishing and exploding is proper initialization of the network's weights. Explain the maximum initial value of W that ensures, regardless of the input, gradient explosion does not occur from the very beginning. Hint: Find an upper bound for the gradient of h_0.

Soloution

To prevent gradient explosion, we must bound the gradient of h_0 ($\frac{\partial l}{\partial h_0}$). Using the chain rule, the gradient propagation from h_T to h_0 is:

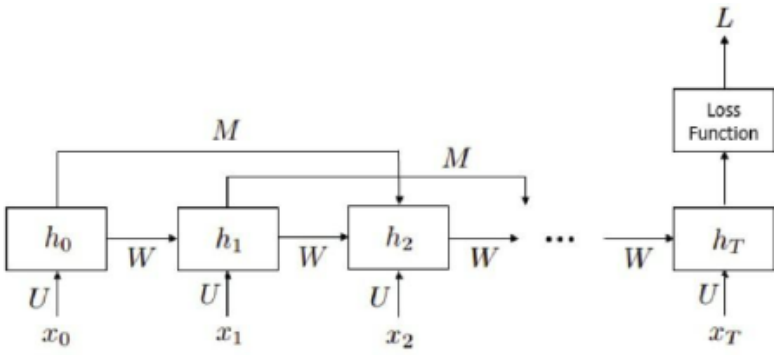
$$\frac{\partial l}{\partial h_0} = \frac{\partial l}{\partial h_T} \cdot \prod_{k=0}^{T-1} \left(W^T \cdot \sigma'(a_k)\right)$$

The sigmoid function is defined as $\sigma(z) = \frac{1}{1+e^{-z}}$. Its derivative is $\sigma'(z) = \sigma(z)(1-\sigma(z))$, which reaches its maximum value of $\frac{1}{4}$ when $\sigma(z) = 0.5$. Therefore, $\sigma'(a_k) \leq \frac{1}{4}$ for all a_k. To ensure the product does not exceed 1, the spectral norm of W must satisfy:

$$\|W\| \leq 4$$

Part 4.

One of the methods to prevent gradient vanishing is the use of **skip connections**. Consider the following structure, where each h_t is connected not only to h_{t+1} but also to h_{t+2}. Now, rewrite the gradient of h_t in terms of the gradients of h_{t+1} and h_{t+2}, and explain why this approach significantly reduces the effect of gradient vanishing. (For $1 \leq t \leq T-2$).



Soloution

The gradient of h_t with respect to the loss l can be expressed as:

$$\frac{\partial l}{\partial h_t} = \frac{\partial l}{\partial h_{t+1}} \cdot W^T \cdot \sigma'(a_t) + \frac{\partial l}{\partial h_{t+2}} \cdot M^T \cdot \sigma'(a_t)$$

1. **Multiple Gradient Paths:** Connecting h_t to both h_{t+1} and h_{t+2} provides two routes for gradients to flow during backpropagation. This redundancy ensures that even if one path weakens, the other can still carry significant gradient information.

2. **Reduced Multiplicative Effect:** Additional connections shorten the paths that gradients must traverse, decreasing the number of multiplicative factors that can diminish the gradients. This helps in maintaining stronger gradients throughout the network.

3. **Enhanced Gradient Flow:** Skip connections create more direct pathways for gradients, facilitating their flow from later to earlier time steps. This preservation of gradient strength is crucial for learning long-term dependencies.

Part 5.

One of the methods to prevent gradient explosion is gradient clipping. This method is divided into two approaches: clipping by value and clipping by norm. Explain these two methods separately. Also, explain the advantage of clipping by norm over clipping by value.

Soloution

Gradient clipping is a technique used to prevent the gradients from becoming excessively large, which can lead to numerical instability and hinder the training process. It is divided into two main approaches:

1. **Clipping by Value:** This method restricts each individual gradient component to lie within a predefined range, typically $[-c, c]$, where c is the threshold value. If a gradient component exceeds this range, it is set to the boundary value. For example, if $c = 5$, any gradient component greater than 5 is set to 5, and any component less than -5 is set to -5.

2. **Clipping by Norm:** Instead of limiting individual components, this approach scales the entire gradient vector if its norm exceeds a certain threshold c. Specifically, if the gradient's norm $\|g\|$ is greater than c, the gradient is scaled down by a factor of $c/\|g\|$, ensuring that $\|g\| \leq c$. This maintains the direction of the gradient while controlling its magnitude.

Advantage of Clipping by Norm over Clipping by Value: Clipping by norm is generally preferred because it preserves the direction of the gradient vector, ensuring that the update direction remains consistent. In contrast, clipping by value can distort the gradient direction when multiple components are clipped, potentially leading to suboptimal updates. By scaling the entire gradient uniformly, clipping by norm maintains the integrity of the gradient's direction while effectively controlling its size, thereby providing a more stable and reliable mechanism to prevent gradient explosion.

Question 4 (70 Points)

In this problem, we aim to familiarize ourselves with the concepts involved in sequence generation in Seq2Seq networks and their advantages and disadvantages. In this question, we aim to examine the concept of **teacher forcing**. For sequence generation, we can consider a basic naive strategy: for generating token $t+1$ using the decoder at time $t+1$, the token generated by the network at time t is used as input to the decoder at time $t+1$. However, this approach has its issues.

Part 1.

First, explain what these issues are. Then, describe the **teacher forcing** method and explain how **teacher forcing** resolves these issues.

Soloution

The naive sequence generation strategy encounters several challenges:

1. **Exposure Bias:** There's a mismatch between training and inference phases since the model uses ground truth tokens during training but relies on its own predictions during inference.
2. **Error Accumulation:** Mistakes made in early predictions can compound, leading to increasingly inaccurate sequences.
3. **Training-Inference Discrepancy:** Differences in input sources during training (ground truth) and inference (model predictions) hinder the model's ability to generalize effectively.

Teacher Forcing is a training technique that addresses these issues by providing the actual ground truth token as input for the next time step during training, instead of using the model's own previous prediction. This method offers several advantages:

1. **Consistent Training Inputs:** Aligns the training process with the inference scenario by using accurate context information.
2. **Reduced Error Propagation:** Minimizes the risk of errors compounding over the sequence, leading to more stable predictions.
3. **Improved Convergence:** Enhances the model's ability to learn sequence dependencies effectively, often resulting in faster and more reliable training.

In summary, Teacher Forcing mitigates exposure bias, prevents error accumulation, and bridges the training-inference gap by ensuring the model is trained with accurate context inputs.

Part 2.

The main issue with **teacher forcing** is a problem known as **exposure bias**. Explain this problem.

Soloution

Exposure Bias refers to the discrepancy between training and inference phases in sequence generation models. During training with teacher forcing, the model receives the ground truth tokens as inputs for each time step, ensuring accurate context. However, during inference, the model relies on its own previously generated tokens, which may contain errors. This mismatch can lead to the model being less robust to its own mistakes, as it was never exposed to them during training. Consequently, errors can accumulate over the sequence, degrading the quality of the generated output and hindering the model's ability to generalize effectively in real-world scenarios.

Part 3.

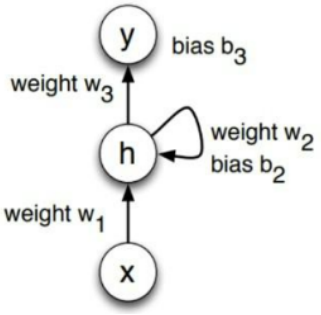
One of the solutions to the **exposure bias** problem is the **scheduled sampling** technique. Explain this technique and describe how it helps reduce the effect of **exposure bias**.

Soloution

Scheduled Sampling is a training strategy that gradually transitions the decoder's input from ground truth tokens to the model's own predictions. During training, at each time step, the model probabilistically decides whether to use the actual target token or its previously generated token as input for generating the next token. The probability of using the model's prediction typically increases over time, allowing the model to become more reliant on its own outputs as training progresses.
This approach helps reduce **exposure bias** by exposing the model to its own prediction errors during training, thereby bridging the gap between training and inference phases. By learning to handle its own generated tokens, the model becomes more robust to inaccuracies and variations in the input it receives during inference, leading to more reliable and coherent sequence generation.

Question 5 (70 Points)

Consider a recurrent network as described below. Set the weights and biases such that, for any sequence of numbers as input to the network, the output of the network remains constant until the input changes to zero. When the input changes to zero, the output of the network should also become zero and remain zero. For example, for an input sequence of 1110101, the output of the network should be 1110000.



Soloution

Hidden State Update

$$h_t = \sigma(w_1 \cdot x_t + w_2 \cdot h_{t-1} + b_2) \quad \& \quad y_t = \sigma(w_3 \cdot h_t + b_3).$$

Design of Weights and Biases

To meet the requirements:
1. **Maintain Output When $x_t = 1$:** If $x_t = 1$, the hidden state h_t should depend on h_{t-1} in such a way that it retains the previous value. This is achieved by setting $w_2 \approx 1$ and ensuring $w_1 \cdot x_t + b_2$ stabilizes h_t near its previous value h_{t-1}.
2. **Reset Output When $x_t = 0$:** If $x_t = 0$, the contribution of $w_1 \cdot x_t$ becomes 0. The bias b_2 and weight w_2 should drive h_t toward 0.
3. **Output y_t:** The output y_t directly reflects h_t with appropriate scaling by w_3 and offset by b_3, ensuring that y_t matches h_t.

Final Configuration

The weights and biases are set as follows:

- $w_1 = 5$: Strong influence from the input x_t.
- $w_2 = 1$: Ensures h_t retains its previous value when $x_t = 1$.
- $b_2 = -5$: Ensures h_t decays to 0 when $x_t = 0$.
- $w_3 = 10$: Amplifies h_t for the output y_t.
- $b_3 = -5$: Adjusts the output range of y_t to match the binary behavior (close to 0 or 1).

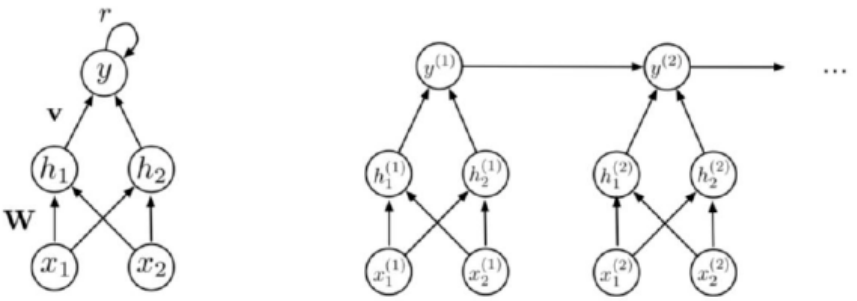
Equations for the Network

1. **Hidden State Update:**

$$h_t = \sigma(5 \cdot x_t + 1 \cdot h_{t-1} - 5) \quad \& \quad y_t = \sigma(10 \cdot h_t - 5).$$

Question 6 (70 Points)

Consider a recurrent network as shown below. Assume that this network receives two sequences of zeros and ones. If the two sequences are identical, it outputs one; otherwise, it outputs zero.



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

$$y^{(t)} = \begin{cases} \phi\left(\mathbf{v}^\top \mathbf{h}^{(t)} + r y^{(t-1)} + c\right) & \text{for } t > 1 \\ \phi\left(\mathbf{v}^\top \mathbf{h}^{(t)} + c_0\right) & \text{for } t = 1, \end{cases} \qquad \phi(z) = \begin{cases} 1 & \text{if } z > 0 \\ 0 & \text{if } z \leq 0 \end{cases}$$

Matrix W is a 2×2 matrix, and b and v are two-dimensional vectors. c, r, and c_0 are scalar values. Set them in such a way that the network achieves the defined functionality. **Hint**: The output $y(t)$ at any moment indicates whether the two sequences have been identical 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.

Soloution

Define the Weight Matrix W

We design W such that:
- The first hidden unit activates when both inputs are zero.
- The second hidden unit activates when both inputs are one.

$$W = \begin{bmatrix} -1 & -1 \\ 1 & 1 \end{bmatrix}$$

Set the Bias Vector b

Biases are used to control the activation thresholds of the hidden units.
Set b as:

$$b = \begin{bmatrix} 1 \\ -1 \end{bmatrix}$$

- **First Hidden Unit Bias ($b_1 = 1$):** The unit activates when both inputs are zero.
- **Second Hidden Unit Bias ($b_2 = -1$):** The unit activates when both inputs are one.

Define the Output Weights v *and Bias* r

To output one only when both hidden units are activated, set v and r as:

$$v = \begin{bmatrix} 1 & 1 \end{bmatrix}, \quad r = -1.5$$

Set the Thresholds c *and* c_0

- Activation Threshold for Hidden Units ($c = 0.5$)
- Output Threshold ($c_0 = 0$)

Operational Logic

- **First Hidden Unit ($h_1(t)$)**

$$h_1(t) = \sigma(-x_1(t) - x_2(t) + 1)$$

Activates when $x_1(t) = 0$ and $x_2(t) = 0$.
- **Second Hidden Unit ($h_2(t)$):**

$$h_2(t) = \sigma(x_1(t) + x_2(t) - 1)$$

Activates when $x_1(t) = 1$ and $x_2(t) = 1$.
- **Output Layer Activation**

$$y(t) = \sigma(v^T h(t) + r) = \sigma(h_1(t) + h_2(t) - 1.5)$$

- **If both $h_1(t) = 1$ and $h_2(t) = 1$**

$$y(t) = \sigma(1 + 1 - 1.5) = \sigma(0.5) \approx 0.622 > 0.5 \quad \Rightarrow \quad y(t) = 1$$

- **Otherwise**

$$y(t) < 0.5 \quad \Rightarrow \quad y(t) = 0$$

Summary of Parameters

$$W = \begin{bmatrix} -1 & -1 \\ 1 & 1 \end{bmatrix}, \quad b = \begin{bmatrix} 1 \\ -1 \end{bmatrix}, \quad v = \begin{bmatrix} 1 & 1 \end{bmatrix}, \quad r = -1.5, \quad c = 0.5, \quad c_0 = 0$$