[linecolor=conceptblue, backgroundcolor=blue!5, linewidth=2pt, topline=false, rightline=false, leftline=true, bottomline=false, innertopmargin=5pt]concept

[linecolor=examplegreen, backgroundcolor=green!5, linewidth=2pt, topline=false, rightline=false, leftline=true, bottomline=false, innertopmargin=5pt]example

[linecolor=notered, backgroundcolor=red!5, linewidth=2pt, topline=false, rightline=false, leftline=true, bottomline=false, innertopmargin=5pt]important

CS550/DSL501: Machine Learning (2023–24–M)
Assignment-II

Full name: Ayush Kumar Mishra **ID: 12240340**

Question 1

Consider a sequence of daily temperatures recorded over a month, where the temperatures exhibit some seasonality and trends. You decide to use an LSTM model to predict the temperature for the next day based on the previous seven days' temperatures.

Part A

Mathematically, describe how the input to your LSTM model will be structured. If the temperatures for the last seven days are represented as:

$$\{T_t, T_{t-1}, T_{t-2}, T_{t-3}, T_{t-4}, T_{t-5}, T_{t-6}\}$$

each input sequence to the model is a vector containing these temperature values.

Part B

Given that LSTMs have memory cells and gates (input, forget, and output gates), explain how these components could theoretically help your model learn long-term dependencies in this temperature data. Describe the mathematical operations performed in these gates, and discuss why they are essential for this type of sequential data.

The operations for each gate are as follows:

- **Forget Gate:**

$$f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f)$$

This gate determines which information to discard from the cell state, which helps in maintaining relevant information over time.

- **Input Gate:**

$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i)$$
$$\tilde{c}_t = \tanh(W_c \cdot [h_{t-1}, x_t] + b_c)$$

The input gate decides what new information to store in the cell state, allowing the model to learn new temperature patterns.

- **Output Gate:**

$$o_t = \sigma(W_o \cdot [h_{t-1}, x_t] + b_o)$$
$$h_t = o_t * \tanh(c_t)$$

This gate controls what information to output from the cell state, which helps in generating predictions based on relevant features.

- **Bidirectional LSTM:** This approach processes the sequence in both forward and backward directions, which could help in better understanding rapid fluctuations.
- **Custom Loss Function:** Implementing a loss function that places more weight on prediction errors during sudden weather changes might improve model accuracy in these cases.

Question 2

Part a) Forward and Backward Pass Calculations

Given that the actual output of $y(O_6)$ is 1 and learning rate is 0.9, we perform another forward pass.

Forward Pass

The forward pass computes the output values using:

$$a_j = \sum_j (w_{i,j} \cdot x_i) \tag{1}$$

Where a_j is the net input to neuron j, $w_{i,j}$ is the weight connecting input i to neuron j, and x_i is the i-th input value.

$$y_j = F(a_j) = \frac{1}{1 + e^{-a_j}} \tag{2}$$

Computing outputs for y_4, y_5, and y_6:

$$\begin{aligned} a_4 &= (w_{14} \cdot x_1) + (w_{24} \cdot x_2) + (w_{34} \cdot x_3) + \theta_4 \\ &= (0.2 \cdot 1) + (0.4 \cdot 0) + (-0.5 \cdot 1) + (-0.4) = -0.7 \\ O(H_4) = y_4 = f(a_4) &= \frac{1}{1 + e^{0.7}} = 0.332 \end{aligned}$$

$$\begin{aligned} a_5 &= (w_{15} \cdot x_1) + (w_{25} \cdot x_2) + (w_{35} \cdot x_3) + \theta_5 \\ &= (-0.3 \cdot 1) + (0.1 \cdot 0) + (0.2 \cdot 1) + (0.2) = 0.1 \\ O(H_5) = y_5 = f(a_5) &= \frac{1}{1 + e^{-0.1}} = 0.525 \end{aligned}$$

$$\begin{aligned} a_6 &= (w_{46} \cdot H_4) + (w_{56} \cdot H_5) + \theta_6 \\ &= (-0.3 \cdot 0.332) + (-0.2 \cdot 0.525) + 0.1 = -0.105 \\ O(O_6) = y_6 = f(a_6) &= \frac{1}{1 + e^{0.105}} = 0.474 \end{aligned}$$

Backward Pass

For output units:

$$\delta_j = o_j(1 - o_j)(t_j - o_j) \tag{3}$$

For hidden units:

$$\delta_j = o_j(1 - o_j) \sum_k \delta_k w_{kj} \tag{4}$$

Computing error gradients:

$$\begin{aligned}\delta_6 &= y_6(1 - y_6)(y_{target} - y_6) \\ &= 0.474 \cdot (1 - 0.474) \cdot (1 - 0.474) = 0.1311\end{aligned}$$

$$\begin{aligned}\delta_5 &= y_5(1 - y_5)w_{56} \cdot \delta_6 \\ &= 0.525 \cdot (1 - 0.525) \cdot (-0.2 \cdot 0.1311) = -0.0065\end{aligned}$$

$$\begin{aligned}\delta_4 &= y_4(1 - y_4)w_{46} \cdot \delta_6 \\ &= 0.332 \cdot (1 - 0.332) \cdot (-0.3 \cdot 0.1311) = -0.0087\end{aligned}$$

Weight Updates

$$\Delta w_{ji} = \eta \delta_j o_i \tag{5}$$

Computing new weights:

$$\begin{aligned}\Delta w_{46} &= \eta \delta_6 y_4 = 0.9 \cdot 0.1311 \cdot 0.332 = 0.03917 \\ w_{46}(new) &= \Delta w_{46} + w_{46}(old) = 0.03917 + (-0.3) = -0.261\end{aligned}$$

$$\begin{aligned}\Delta w_{14} &= \eta \delta_4 x_1 = 0.9 \cdot -0.0087 \cdot 1 = -0.0078 \\ w_{14}(new) &= \Delta w_{14} + w_{14}(old) = -0.0078 + 0.2 = 0.192\end{aligned}$$

New Forward Pass

Computing outputs with updated weights:

$$\begin{aligned}a_4 &= (0.192 \cdot 1) + (0.4 \cdot 0) + (-0.508 \cdot 1) + (-0.408) = -0.724 \\ O(H_4) = y_4 &= f(a_4) = \frac{1}{1 + e^{0.724}} = 0.327\end{aligned}$$

$$\begin{aligned}a_5 &= (-0.306 \cdot 1) + (0.1 \cdot 0) + (0.194 \cdot 1) + (0.194) = 0.082 \\ O(H_5) = y_5 &= f(a_5) = \frac{1}{1 + e^{-0.082}} = 0.520\end{aligned}$$

$$\begin{aligned}a_6 &= (-0.261 \cdot 0.327) + (-0.138 \cdot 0.520) + 0.218 = 0.061 \\ O(O_6) = y_6 &= f(a_6) = \frac{1}{1 + e^{-0.061}} = 0.515\end{aligned}$$

Part b Activation Functions

Given the net input calculation:

$$\begin{aligned} y_{in} &= b + \sum_{i=1}^{n} x_i w_i \\ &= b + x_1 w_1 + x_2 w_2 + x_3 w_3 \\ &= 0.35 + 0.8 \times 0.1 + 0.6 \times 0.3 + 0.4 \times (-0.2) \\ &= 0.35 + 0.08 + 0.18 - 0.08 = 0.53 \end{aligned}$$

Binary Sigmoid Activation Function

$$y = f(y_{in}) = \frac{1}{1 + e^{-y_{in}}} = \frac{1}{1 + e^{-0.53}} = 0.625 \tag{6}$$

Bipolar Sigmoid Activation Function

$$y = f(y_{in}) = \frac{2}{1 + e^{-y_{in}}} - 1 = \frac{2}{1 + e^{-0.53}} - 1 = 0.259 \tag{7}$$

Question 3: Ensemble Learning with AdaBoost

Core Concept: AdaBoost (Adaptive Boosting)
AdaBoost is an ensemble learning algorithm that combines multiple weak classifiers to create a strong classifier. Each weak classifier is assigned a weight based on its performance, and misclassified examples receive higher weights in subsequent iterations.

Problem Setup

Given Parameters:

[leftmargin=*]Three trained classifiers: h_1, h_2, and h_3 Weight coefficients:

$$\begin{aligned} \alpha_1 &= 0.2 \quad \text{(First classifier)} \\ \alpha_2 &= -0.3 \quad \text{(Second classifier)} \\ \alpha_3 &= -0.2 \quad \text{(Third classifier)} \end{aligned}$$

Predictions for test example x:

$$\begin{aligned} h_1(x) &= +1 \quad \text{(Positive class)} \\ h_2(x) &= +1 \quad \text{(Positive class)} \\ h_3(x) &= -1 \quad \text{(Negative class)} \end{aligned}$$

Solution Process

Step 1: Understanding the Final Classifier Formula

$$H(x) = \text{sign}\left(\sum_{i=1}^{3} \alpha_i h_i(x)\right)$$

This formula represents the weighted majority vote of all weak classifiers.

Step 2: Calculating the Weighted Sum

$$\begin{aligned}\sum_{i=1}^{3} \alpha_i h_i(x) &= \alpha_1 h_1(x) + \alpha_2 h_2(x) + \alpha_3 h_3(x) \\ &= (0.2 \cdot (+1)) + (-0.3 \cdot (+1)) + (-0.2 \cdot (-1)) \\ &= 0.2 - 0.3 + 0.2 \\ &= 0.1\end{aligned}$$

Step 3: Applying the Sign Function

$$H(x) = \text{sign}(0.1) = +1$$

Since $0.1 > 0$, the sign function returns $+1$

Key Insights:

[leftmargin=*]The positive result (0.1) indicates a slight lean towards the positive class Despite two classifiers having negative weights, the final prediction is positive The combination of weights and predictions shows the adaptive nature of AdaBoost

Question 4: Advanced Attention Mechanisms

Core Concept: Attention in Neural Networks
Attention mechanisms allow neural networks to selectively focus on different parts of the input, similar to human visual attention. They are crucial for tasks like summarization where context-dependent understanding is essential.

Part (a): Attention Score Calculation

Mathematical Definition:

$$e_{ij} = v^T \tanh(W_1 h_i + W_2 s_j) \tag{8}$$

Component Breakdown:

[leftmargin=*]h_i: i-th encoder hidden state s_j: j-th decoder state W_1, W_2: Trainable weight matrices v: Trainable vector tanh: Non-linear activation function

Process Flow:

[leftmargin=*]**Transform States:** Apply W_1 to h_i and W_2 to s_j **Add States:** Combine transformed states **Apply Non-linearity:** Use tanh activation **Project to Scalar:** Multiply with v^T

Part (b): Attention Weight Computation

Softmax Normalization:

$$\alpha_{ij} = \frac{\exp(e_{ij})}{\sum_{k=1}^{n} \exp(e_{ik})} \tag{9}$$

Properties:

[leftmargin=*]Normalized weights: $\sum_{i=1}^{n} \alpha_{ij} = 1$ Range: $0 \leq \alpha_{ij} \leq 1$ Interpretable as probabilities

Part (c): Multi-Head Attention

Multi-Head Architecture:

$$\text{head}_i = \text{Attention}(QW_i^Q, KW_i^K, VW_i^V) \tag{10}$$

Final Output:

$$\text{MultiHead}(Q, K, V) = \text{Concat}(\text{head}_1, \ldots, \text{head}_h)W^O \tag{11}$$

Advantages of Multi-Head Attention:

[leftmargin=*]**Parallel Processing:**

- Multiple attention patterns computed simultaneously
- Each head can specialize in different aspects
- Efficient computation through parallelization

- **Enhanced Feature Capture:**
  - Different heads focus on different feature subspaces
  - Combines local and global information
  - Better handling of complex dependencies
- **Improved Model Capacity:**
  - Increased representational power
  - Better handling of diverse patterns
  - More robust feature extraction

Key Implementation Considerations:

[leftmargin=*]Choose appropriate number of attention heads Balance between computational cost and model capacity Ensure proper initialization of weight matrices Consider using residual connections

Conclusion

This concludes the assignment. Thank you for your attention.

- *End of Assignment*