Gated Recurrent Unit (GRU) Networks

Overview

GRUs are a type of Recurrent Neural Network (RNN) designed to efficiently capture dependencies in sequential data, such as time series, text, or speech. GRUs help solve the vanishing gradient problem found in traditional RNNs by using gating mechanisms to control information flow.

Mathematical Formulation

At each time step t, the GRU takes an input vector $\mathbf{x}_t$ and the previous hidden state $\mathbf{h}_{t-1}$ to compute the current hidden state $\mathbf{h}_t$.

Step 1: Update Gate $\mathbf{z}_t$

The update gate decides how much of the previous hidden state to keep:

$$\mathbf{z}_t = \sigma(\mathbf{W}_z \mathbf{x}_t + \mathbf{U}_z \mathbf{h}_{t-1} + \mathbf{b}_z)$$

- σ is the sigmoid function.
- $\mathbf{W}_z, \mathbf{U}_z$ are weight matrices.
- $\mathbf{b}_z$ is a bias vector.

Step 2: Reset Gate $\mathbf{r}_t$

The reset gate decides how much of the previous state to forget:

$$\mathbf{r}_t = \sigma(\mathbf{W}_r \mathbf{x}_t + \mathbf{U}_r \mathbf{h}_{t-1} + \mathbf{b}_r)$$

Step 3: Candidate Hidden State $\tilde{\mathbf{h}}_t$

Compute a candidate hidden state $\tilde{\mathbf{h}}_t$ that uses the reset gate to control the influence of the previous hidden state:

$$\tilde{\mathbf{h}}_t = \tanh(\mathbf{W}_h \mathbf{x}_t + \mathbf{U}_h (\mathbf{r}_t \odot \mathbf{h}_{t-1}) + \mathbf{b}_h)$$

- $\odot$ is element-wise multiplication.

Step 4: Final Hidden State $\mathbf{h}_t$

Combine the previous hidden state and candidate hidden state using the update gate:

$$\mathbf{h}_t = (1 - \mathbf{z}_t) \odot \mathbf{h}_{t-1} + \mathbf{z}_t \odot \tilde{\mathbf{h}}_t$$

Explanation of Each Step

1. **Update Gate $\mathbf{z}_t$**
   Controls how much information from the past state $\mathbf{h}_{t-1}$ should be carried forward to the current state. If z_t is close to 1, the network updates the hidden state mostly based on the candidate $\tilde{\mathbf{h}}_t$. If close to 0, it keeps the old hidden state.
2. **Reset Gate $\mathbf{r}_t$**
   Controls how much of the past information to forget when calculating the candidate hidden state. A reset gate near 0 "forgets" the previous hidden state, focusing only on the current input.
3. **Candidate Hidden State $\tilde{\mathbf{h}}_t$**
   This is a temporary state calculated based on the current input and the reset-modified previous state.
4. **Final Hidden State $\mathbf{h}_t$**
   Combines the old hidden state and the candidate hidden state based on the update gate.

Numerical Example (Simple Case)

Setup

- Let input $\mathbf{x}_t = [0.5]$ (scalar for simplicity)
- Previous hidden state $\mathbf{h}_{t-1} = [0.1]$
- Weights and biases (all scalar for simplicity):

$$W_z = 0.8, \quad U_z = 0.1, \quad b_z = 0.0$$

$$W_r = 0.5, \quad U_r = 0.2, \quad b_r = 0.0$$

$$W_h = 0.9, \quad U_h = 0.3, \quad b_h = 0.0$$

Step 1: Calculate update gate z_t

$$z_t = \sigma(W_z x_t + U_z h_{t-1} + b_z) = \sigma(0.8 \times 0.5 + 0.1 \times 0.1 + 0) = \sigma(0.4 + 0.01) = \sigma(0.41)$$

Sigmoid function:

$$\sigma(0.41) = \frac{1}{1 + e^{-0.41}} \approx 0.601$$

Step 2: Calculate reset gate r_t

$$r_t = \sigma(W_r x_t + U_r h_{t-1} + b_r) = \sigma(0.5 \times 0.5 + 0.2 \times 0.1 + 0) = \sigma(0.25 + 0.02) = \sigma(0.27)$$

$$\sigma(0.27) = \frac{1}{1 + e^{-0.27}} \approx 0.567$$

Step 3: Calculate candidate hidden state $\tilde{h}_t$

First compute reset applied hidden state:

$$r_t \times h_{t-1} = 0.567 \times 0.1 = 0.0567$$

Then,

$$\tilde{h}_t = \tanh(W_h x_t + U_h (r_t h_{t-1}) + b_h) = \tanh(0.9 \times 0.5 + 0.3 \times 0.0567 + 0)$$

$$= \tanh(0.45 + 0.017) = \tanh(0.467) \approx 0.436$$

Step 4: Calculate final hidden state h_t

$$h_t = (1 - z_t) \times h_{t-1} + z_t \times \tilde{h}_t = (1 - 0.601) \times 0.1 + 0.601 \times 0.436$$

$$= 0.399 \times 0.1 + 0.601 \times 0.436 = 0.0399 + 0.262 = 0.302$$

Variable	Value
Zt (update gate)	0.601
Rt (reset gate)	0.567
h~t (candidate hidden)	0.436
Ht (new hidden state)	0.302