Gated Recurrent

## Getting Started with Gated Recurrent Units (GRU)

Gated Recurrent Units (GRUs) are a type of RNN introduced by Cho et al. in 2014. The core idea behind GRUs is to use **gating mechanisms** to selectively update the hidden state at each time step allowing them to remember important information while discarding irrelevant details. GRUs aim to simplify the LSTM architecture by merging some of its components and focusing on just two main gates: the **update gate** and the **reset gate**.

## Core Structure of GRUs

The GRU consists of **two main gates**:

1. **Update Gate** (zt$z_t$): This gate decides how much information from previous hidden state should be retained for the next time step.
2. **Reset Gate** (rt$r_t$): This gate determines how much of the past hidden state should be forgotten.

These gates allow GRU to control the flow of information in a more efficient manner compared to traditional RNNs which solely rely on hidden state.

## Equations for GRU Operations

The internal workings of a GRU can be described using following equations:

**1. Reset gate**:

rt=σ(Wr·[ht−1,xt])$r_t=\sigma(W_r\cdot[h_{t-1},x_t])$

The reset gate determines how much of the previous hidden state ht−1$h_{t-1}$ should be forgotten.

**2. Update gate**:

zt=σ(Wz·[ht−1,xt])$z_t=\sigma(W_z\cdot[h_{t-1},x_t])$

The update gate controls how much of the new information xt$x_t$ should be used to update the hidden state.

**3. Candidate hidden state**:

ht'=tanh(Wh·[rt·ht−1,xt])$h_t'=\tanh(W_h\cdot[r_t\cdot h_{t-1},x_t])$

This is the potential new hidden state calculated based on the current input and the previous hidden state.

**4. Hidden state**:

ht=(1–zt)·ht−1+zt·ht′$h_t=(1-z_t)\cdot h_{t-1}+z_t\cdot h_t'$

The final hidden state is a weighted average of the previous hidden state ht−1$h_{t-1}$ and the candidate hidden state ht′$h_t'$ based on the update gate zt$z_t$.

## How GRUs Solve the Vanishing Gradient Problem

Like LSTMs, GRUs were designed to address the **vanishing gradient problem** which is common in traditional RNNs. GRUs help mitigate this issue by using gates that regulate the flow of gradients during training ensuring that important information is preserved and that gradients do not shrink excessively over time. By using these gates, GRUs maintain a balance between remembering important past information and learning new, relevant data.

## GRU vs LSTM

GRUs are more computationally efficient because they combine the forget and input gates into a single update gate. GRUs do not maintain an internal cell state as LSTMs do, instead they store information directly in the hidden state making them simpler and faster.

| Feature | LSTM (Long Short-Term Memory) | GRU (Gated Recurrent Unit) |
|---|---|---|
| **Gates** | 3 (Input, Forget, Output) | 2 (Update, Reset) |
| **Cell State** | Yes it has cell state | No (Hidden state only) |
| **Training Speed** | Slower due to complexity | Faster due to simpler architecture |
| **Computational Load** | Higher due to more gates and parameters | Lower due to fewer gates and parameters |
| **Performance** | Often better in tasks requiring long-term memory | Performs similarly in many tasks with less complexity |