Recurrent Neural Network (RNN) Backpropagation

1. Forward Pass in RNN (Brief Recap)

- Input sequence: x_1, x_2, x_3, x_4
- At each time step t, hidden state o_t is computed:

$$\begin{aligned} o_1 &= f(x_1, w) \\ o_2 &= f(x_2, w + o_1 w) \\ o_3 &= f(x_3, w + o_2 w) \\ o_4 &= f(x_4, w + o_3 w) \end{aligned}$$

- Final output $\hat{y}_4 = f(o_4, w'')$, passed through **Softmax** for classification.

---

2. Loss and Gradient Calculation

- Loss function:

$$\text{Loss} = |\hat{y}_4 - y_4|$$

- Objective: **minimize the loss** by updating weights w (shared across all time steps) and output weight w''.

---

3. Backpropagation Through Time (BPTT)

Step 1: Start from Output Layer

- Compute gradient of loss w.r.t output:

$$\frac{\partial L}{\partial \hat{y}_4}$$

- Then use **chain rule** to compute gradient of loss w.r.t output weight w'':

$$\frac{\partial L}{\partial w''} = \frac{\partial L}{\partial \hat{y}_4} \cdot \frac{\partial \hat{y}_4}{\partial w''}$$

Step 2: Move Backward to Hidden Layers

- Compute gradient for **hidden weight** w using chain rule:

$$\frac{\partial L}{\partial w} = \frac{\partial L}{\partial \hat{y}_4} \cdot \frac{\partial \hat{y}_4}{\partial o_4} \cdot \frac{\partial o_4}{\partial w}$$

- Similarly, for earlier time steps (e.g., weight update at x_3), again apply chain rule:

$$\frac{\partial L}{\partial w} = \frac{\partial L}{\partial \hat{y}_4} \cdot \frac{\partial \hat{y}_4}{\partial o_4} \cdot \frac{\partial o_4}{\partial o_3} \cdot \frac{\partial o_3}{\partial w}$$

Note:

- This process continues **for all time steps**, backward from the output layer to the earliest input.

4. Key Problems in Simple RNNs

(a) Vanishing Gradient Problem

- Cause: The **derivative of sigmoid activation function** lies in (0, 1).
- During backpropagation, repeatedly multiplying small values causes gradients to **shrink toward zero**.
- Result: **No meaningful weight updates** — network stops learning.
- Important: Happens **not because of many hidden layers**, but due to **many time steps** (long sequences).

(b) Exploding Gradient Problem

- Cause: If activation function (like ReLU) produces large derivatives (>1), gradients can **grow very large**.
- Result: Extremely large weight updates; training becomes **unstable** or **diverges**.

---

5. Consequence of Both Problems

- In both cases, the RNN **fails to reach global minima** in the loss landscape:
  - **Vanishing gradients**: No learning (flat slope).
  - **Exploding gradients**: Overshooting minima.

6. Activation Function Impacts

Activation	**Range**	**Problem**
Sigmoid	(0, 1)	Vanishing gradient
Tanh	(-1, 1)	Vanishing (less severe)
ReLU	[0, ∞)	Exploding gradient

🧠 Conclusion

- **Backpropagation in RNN** uses the **chain rule** repeatedly over time steps.

- Problems like **vanishing/exploding gradients** arise due to the nature of repeated multiplication of derivatives over time, not due to depth of layers.
- This is why **LSTM** and **GRU** were developed — to solve these gradient issues.