ARTIFICIAL NEURAL NETWORKS AND DEEP LEARNING

Recurrent Neural Network

Joshua Llano

Universitat Politècnica de Catalunya

Table of contents

Introduction

What are Recurrent Neural Networks?

- A recurrent neural network (RNN) is a type of artificial neural network which uses sequential data or time series data.
- These deep learning algorithms are commonly used for ordinal or temporal problems.
- Examples: language translation, natural language processing (NLP), speech recognition, and image captioning
- They are incorporated into popular applications such as Siri, voice search, and Google Translate.

What are Recurrent Neural Networks?

- They are distinguished by their "memory" as they take information from prior inputs to influence the current output.
- The output of recurrent neural networks depend on the prior elements within the sequence.
- RNN account for the position of each element in the sequence. They use that information to predict the next word in the sequence.

What are Recurrent Neural Networks?

- Feed-forward NN allow data to flow only in one direction: from input to output (top-down architecture).
- RNN have signals traveling in both directions by using feedback loops in the network.

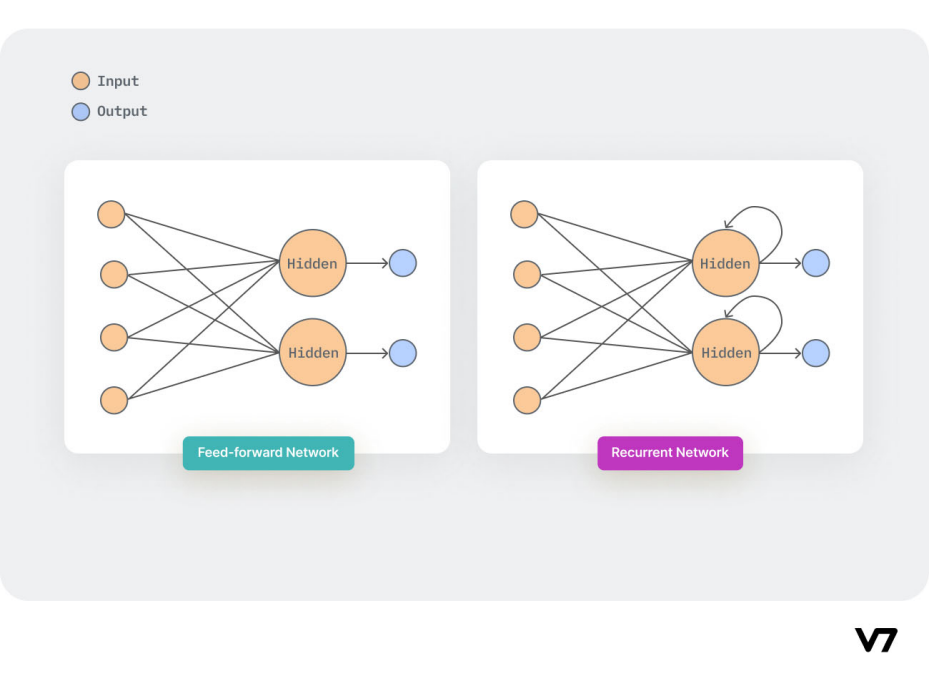


Figure 1: Feed-forward vs Recurrent

What are Recurrent Neural Networks?

- A simplified way of representing the RNN is by unfolding/unrolling the RNN over the input sequence.
- For example, if we feed a sentence as input to the RNN that has 10 words, the network would be unfolded such that it has 10 neural network layers.

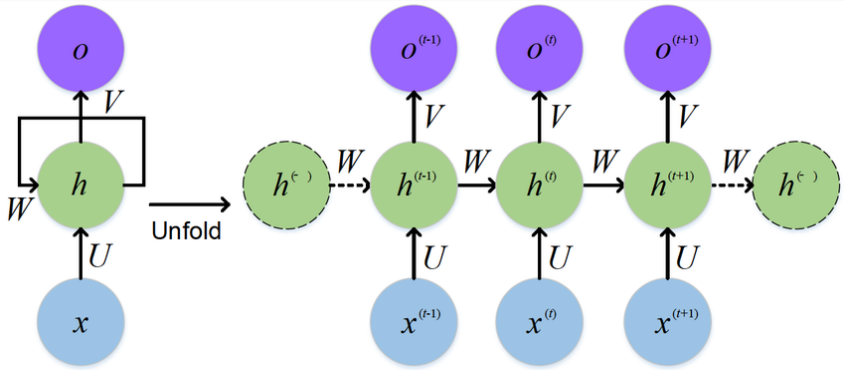


Figure 2: Unfolding the Neural Network architecture over time

Basic RNN Structure

Basic RNN Structure

- The recurrent structure of RNNs is designed to handle sequential data and capture dependencies over time.
- RNNs maintain hidden states that capture information about previous inputs in the sequence.

Basic RNN Structure

- **Input Layer**: The input layer represents the initial input at each time step in the sequence. It could be a word in a sentence, a data point in a time series, etc.
- **Hidden Layer (Recurrent Connection)**: The hidden layer of an RNN is where the recurrent connections occur. At each time step, the hidden layer processes i) the current input and ii) takes into account the information from the previous time step.
  This hidden state serves as a *memory* that helps the network capture sequential dependencies.

Basic RNN Structure

- **Hidden Layer (Recurrent Connection)**: Mathematically, the hidden state h_t at time step t is computed as a function of the current input x_t and the previous hidden state h_{t-1}:

$$h_t = f(W_{hh}h_{t-1} + W_{hx}x_t + b_h)$$

  where:

  - W_{hh} is the weight matrix for the recurrent connections,
  - W_{hx} is the weight matrix for the input connections,
  - b_h is the bias term,
  - f is an activation function.

- The key idea is that the hidden state is updated at each time step, and it retains information about the entire sequence seen so far. This allows RNNs to model sequential patterns.

Basic RNN Structure

- **Output layer**: It produces the prediction or output based on the information in the hidden state. This output can be used for classification, regression, or other tasks.

$$y_t = g(W_{yh} h_t + b_y)$$

where:

  - W_{yh} is the weight matrix for the output connections,
  - b_y is the bias term,
  - g is an activation function.

Basic RNN Structure

- At any given time t, the current output is a combination of input at x(t) and x(t-1).
- The output is fetched back to the network.

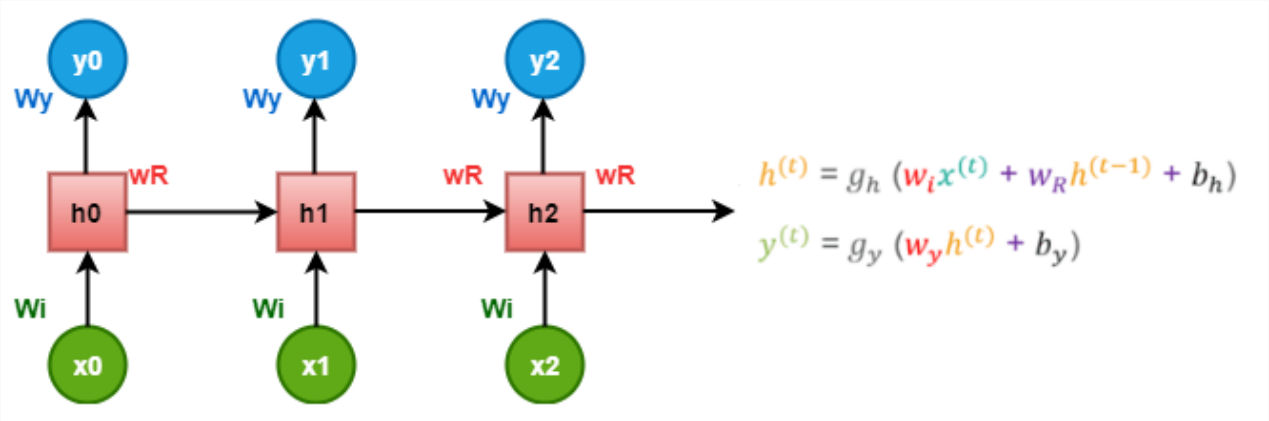


Figure 3: Fully connected RNN. "x" is the input layer, "h" is the hidden layer, and "y" is the output layer. W_y, W_i, and W_R are the network parameters.

Types of RNN

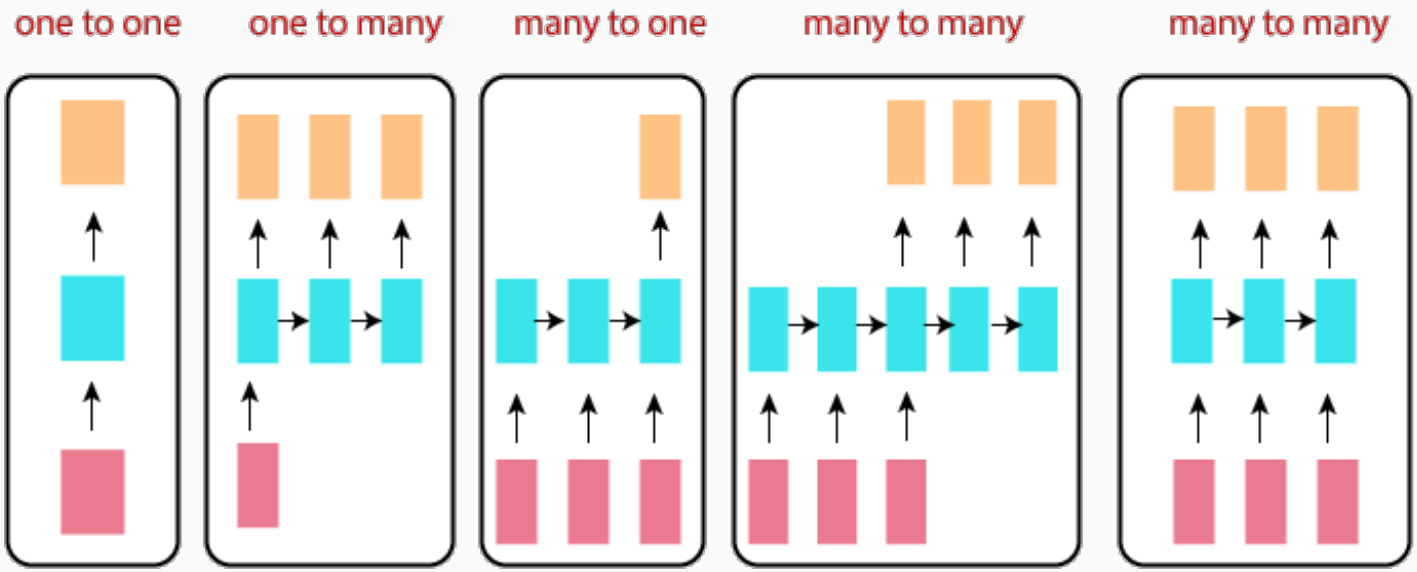


Figure 4: Types of RNN. Number of inputs (red) and number of outputs (orange).

Types of RNN

- **One to one**: single input and a single output (fixed input and output sizes). It acts as a traditional neural network. Ex. *Image Classification*.
- **One to many**: multiple outputs when given a single input. It deals with a fixed size of information as input and outputs a sequence of data. Ex. *Music Generation* and *Image Captioning*.
- **Many to one**: a single output is required from multiple input units. It takes a sequence of information as input and outputs a fixed size of the output. Ex. *Sentiment Analysis*.
- **Many to many**: generate a sequence of output data from a sequence of input units. Ex. *Machine Translation*. It can be Equal Unit Size or Unequal Unit Size

Training and Optimization

Training RNNs

- RNNs use a backpropagation algorithm for training. It is applied for each timestamp. It is known as Back-propagation by Time (BTT).
- Gradients of the loss function can become very small (or large) with respect to the parameters as they propagate through time.

Figure 5: RNNs can suffer from the problem of **vanishing or exploding gradients**, which can make the network difficult to train effectively.

Architectures

Challenges

- There can be scenarios where learning from the immediately preceding data in a sequence is insufficient. Consider a case where you are trying to predict a sentence from another sentence that was introduced a while back in a book or article.
- Remembering the immediately preceding data and the earlier ones is crucial.
- A RNN, owing to the parameter sharing mechanism, uses the same weights at every time step. Thus back propagation makes the gradient either explodes or vanishes, and the neural network doesn't learn much from the data, which is far from the current position.
- LSTM and GRU are types of RNN that are designed to handle the vanishing gradient problem that can occur in standard RNNs.

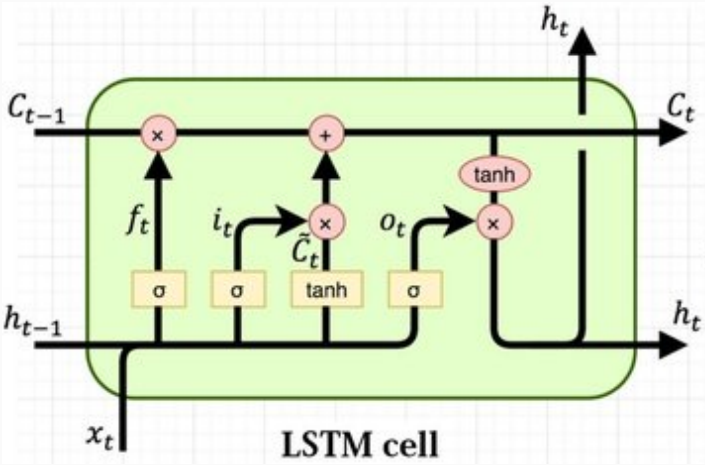
Long Short-Term Memory (LSTM)

- It introduces gating mechanisms that control the flow of information through the network: the input gate, the forget gate, and the output gate.
- These gates allow the LSTM network to selectively remember or forget information from the input sequence, which makes it more effective for long-term dependencies.
- The **input gate** finds which value from input should be used to modify the memory.
- The **forget gate** learns what details to be discarded from the block.
- The **output gate** discovers the input and the memory of the block is used to decide the output.
- A sigmoidal function is used as the gate activation function. Range [0-1].

Advantages of LSTMs

- **Mitigation of gradient vanishing:** LSTMs are designed to retain information for long periods of time, allowing them to learn long-term dependencies.
- **Handling long sequences:** They can capture complex patterns in long sequences, which is essential for tasks such as machine translation, text generation, and more.
- **Flexibility:** LSTMs can be used in different network configurations, including many-to-many (text translation), one-to-many (text generation from one word), and many-to-one (sequence classification).

LSTM scheme



$$i_t = \sigma\left(x_t U^i + h_{t-1} W^i\right)$$
$$f_t = \sigma\left(x_t U^f + h_{t-1} W^f\right)$$
$$o_t = \sigma\left(x_t U^o + h_{t-1} W^o\right)$$
$$\tilde{C}_t = \tanh\left(x_t U^g + h_{t-1} W^g\right)$$
$$C_t = \sigma\left(f_t * C_{t-1} + i_t * \tilde{C}_t\right)$$
$$h_t = \tanh(C_t) * o_t$$

Figure 6: LSTM has three gates: input gate, forget gate and output gate.

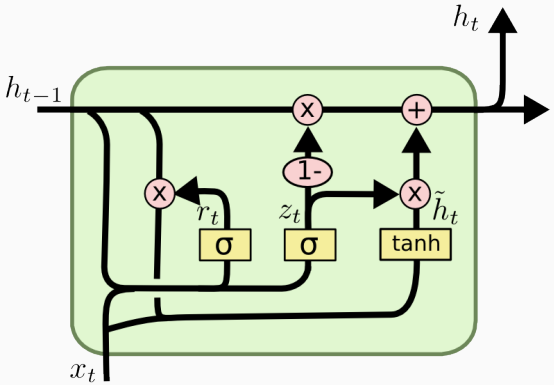
Gated Recurrent Units (GRU)

- GRU uses update and reset gate. Basically, these are two vectors which decide what information should be passed to the output.
- They can be trained to keep long-term information without washing it through time or remove information which is irrelevant to the prediction.
- The **update gate** is responsible for determining the amount of previous information that needs to pass along the next state.
- The **reset gate** is used from the model to decide how much of the past information is needed to neglect.

Advantages of GRUs

- **Simpler architecture:** GRUs have fewer parameters than LSTMs, which makes them faster to train and less susceptible to overfitting.
- **Computational efficiency:** Due to their simpler architecture, GRUs are more computationally efficient and can be faster in terms of training and prediction time.
- **Competitive performance:** Despite their simplicity, GRUs often achieve comparable performance to LSTMs on many sequential learning tasks.

GRU scheme



$$z_t = \sigma\left(W_z \cdot [h_{t-1}, x_t]\right)$$

$$r_t = \sigma\left(W_r \cdot [h_{t-1}, x_t]\right)$$

$$\tilde{h}_t = \tanh\left(W \cdot [r_t * h_{t-1}, x_t]\right)$$

$$h_t = (1 - z_t) * h_{t-1} + z_t * \tilde{h}_t$$

Figure 7: GRU has two gates: reset gate and update gate.

Conclusion

Summary

- RNNs are a specialized class of neural networks used to process sequential data.
- Modeling sequential data requires persisting the data learned from the previous instances.
- RRN implements Parameter Sharing by using same weights at every time stamp so as to accommodate varying lengths of the sequential data for which it makes use of feedback loops.
- A main limitation of RNN is that the gradient either explodes or vanishes; The network doesn't learn much from the data which is far away from the current position.
- RNNs have short term memory problem. To overcome this problem specialized versions of RNN are created like LSTM, GRU.
- RNN processes inputs in a strict temporal order. This means current input has context of previous inputs but not the future.