Recurrent Neural Networks

Remembering Past Data

Dr. Mauricio Toledo-Acosta

Diplomado Ciencia de Datos con Python

Table of Contents

Recurrent Neural Networks

- **Recurrent neural networks are designed for sequential data** like text sentences or time-series.

Recurrent Neural Networks

- **Recurrent neural networks are designed for sequential data** like text sentences or time-series.
- The input is of the form $x_1, \ldots, x_n$, where x_t is a d-dimensional point received at the time-stamp t.

Recurrent Neural Networks

- **Recurrent neural networks are designed for sequential data** like text sentences or time-series.
- The input is of the form $x_1, \ldots, x_n$, where x_t is a d-dimensional point received at the time-stamp t.
- **Successive data are dependent on past data**. Therefore, it is helpful to receive a particular input x_t only after the earlier inputs have already been received and converted into a hidden state.

Recurrent Neural Networks

- **Recurrent neural networks are designed for sequential data** like text sentences or time-series.
- The input is of the form $x_1, \ldots, x_n$, where x_t is a d-dimensional point received at the time-stamp t.
- **Successive data are dependent on past data**. Therefore, it is helpful to receive a particular input x_t only after the earlier inputs have already been received and converted into a hidden state.
- The traditional type of feed-forward network in which all inputs feed into the first layer does not achieve this goal.

Recurrent Neural Networks

- RNNs allows the input x_t to interact directly with the hidden state created from the inputs at previous time stamps.

Recurrent Neural Networks

- RNNs allows the input x_t to interact directly with the hidden state created from the inputs at previous time stamps.
- The basic architecture of the recurrent neural network is illustrated below

Recurrent Neural Networks

- RNNs allows the input x_t to interact directly with the hidden state created from the inputs at previous time stamps.
- The basic architecture of the recurrent neural network is illustrated below
- There is an input x_t at each time-stamp, and a hidden state h_t that changes at each time stamp as new data points arrive.

Recurrent Neural Networks

- RNNs allows the input x_t to interact directly with the hidden state created from the inputs at previous time stamps.
- The basic architecture of the recurrent neural network is illustrated below
- There is an input x_t at each time-stamp, and a hidden state h_t that changes at each time stamp as new data points arrive.
- Each time-stamp also has an output value y_t. For example,

Recurrent Neural Networks

- RNNs allows the input x_t to interact directly with the hidden state created from the inputs at previous time stamps.
- The basic architecture of the recurrent neural network is illustrated below
- There is an input x_t at each time-stamp, and a hidden state h_t that changes at each time stamp as new data points arrive.
- Each time-stamp also has an output value y_t. For example,
    - In a time-series setting, the output y_t is the forecasted prediction of x_{t+1}.

Recurrent Neural Networks

- RNNs allows the input x_t to interact directly with the hidden state created from the inputs at previous time stamps.
- The basic architecture of the recurrent neural network is illustrated below
- There is an input x_t at each time-stamp, and a hidden state h_t that changes at each time stamp as new data points arrive.
- Each time-stamp also has an output value y_t. For example,
    - In a time-series setting, the output y_t is the forecasted prediction of x_{t+1}.
    - In the text-setting, we are predicting the next word.

Recurrent Neural Networks

- RNNs allows the input x_t to interact directly with the hidden state created from the inputs at previous time stamps.
- The basic architecture of the recurrent neural network is illustrated below
- There is an input x_t at each time-stamp, and a hidden state h_t that changes at each time stamp as new data points arrive.
- Each time-stamp also has an output value y_t. For example,
    - In a time-series setting, the output y_t is the forecasted prediction of x_{t+1}.
    - In the text-setting, we are predicting the next word.
    - In some applications, we do not output y_t at each time stamp, but only at the end of the sequence (sentiment analysis).

Examples of RNN

- Fully recurrent neural networks (FRNN)

Examples of RNN

- Fully recurrent neural networks (FRNN)
- Independently RNN (IndRNN)

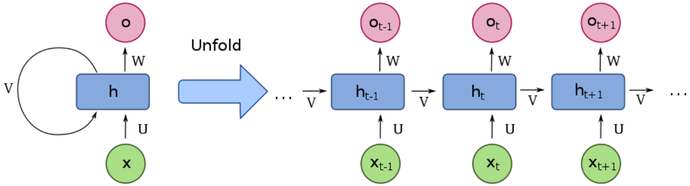
Examples of RNN

- Fully recurrent neural networks (FRNN)
- Independently RNN (IndRNN)
- Long short-term memory (LSTM)

Examples of RNN

- Fully recurrent neural networks (FRNN)
- Independently RNN (IndRNN)
- Long short-term memory (LSTM)
- Gated recurrent units (GRUs)

General Architecture of a RNN



The vanishing gradient problem

- The vanishing gradient problem is encountered when training artificial neural networks with gradient-based learning methods.

The vanishing gradient problem

- The vanishing gradient problem is encountered when training artificial neural networks with gradient-based learning methods.
- In these methods, during each iteration of training each of the neural network's weights receives an update proportional to the partial derivative of the error function.

The vanishing gradient problem

- The vanishing gradient problem is encountered when training artificial neural networks with gradient-based learning methods.
- In these methods, during each iteration of training each of the neural network's weights receives an update proportional to the partial derivative of the error function.
- In some cases, the gradient will be very small, preventing the weight from changing its value.

The vanishing gradient problem

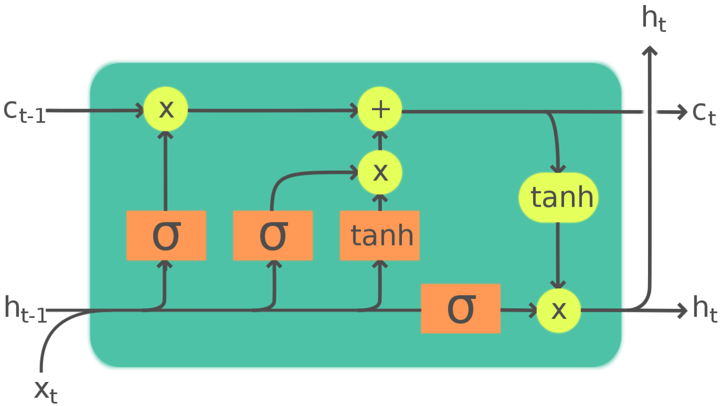
- The vanishing gradient problem is encountered when training artificial neural networks with gradient-based learning methods.
- In these methods, during each iteration of training each of the neural network's weights receives an update proportional to the partial derivative of the error function.
- In some cases, the gradient will be very small, preventing the weight from changing its value.
- For example, the hyperbolic tangent function have gradients in the range $(0, 1]$, therefore, the gradient decreases exponentially with the number of layers.

The vanishing gradient problem

- The vanishing gradient problem is encountered when training artificial neural networks with gradient-based learning methods.
- In these methods, during each iteration of training each of the neural network's weights receives an update proportional to the partial derivative of the error function.
- In some cases, the gradient will be very small, preventing the weight from changing its value.
- For example, the hyperbolic tangent function have gradients in the range $(0, 1]$, therefore, the gradient decreases exponentially with the number of layers.
- With activation functions whose derivatives can take on larger values, one risks encountering the related opposite exploding gradient problem.

Table of Contents

Long short-term memory (LSTM)



LSTM

- Unlike standard feedforward neural networks, **LSTM has feedback connections**. LSTMs can process not only single data points, but also entire sequences of data.

LSTM

- Unlike standard feedforward neural networks, **LSTM has feedback connections**. LSTMs can process not only single data points, but also entire sequences of data.
- **A LSTM unit is composed of a cell, an input gate, an output gate and a forget gate**. The cell remembers values over arbitrary time intervals and the three gates regulate the flow of information into and out of the cell.

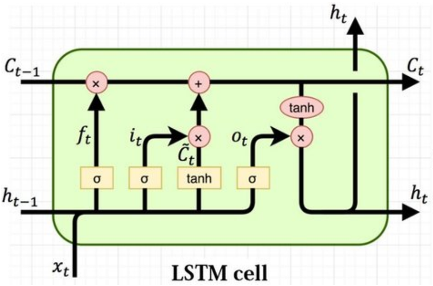
LSTM

- Unlike standard feedforward neural networks, **LSTM has feedback connections**. LSTMs can process not only single data points, but also entire sequences of data.
- **A LSTM unit is composed of a cell, an input gate, an output gate and a forget gate**. The cell remembers values over arbitrary time intervals and the three gates regulate the flow of information into and out of the cell.
- LSTM networks are well-suited to classifying, and making predictions based on time series data.

LSTM

- Unlike standard feedforward neural networks, **LSTM has feedback connections**. LSTMs can process not only single data points, but also entire sequences of data.
- **A LSTM unit is composed of a cell, an input gate, an output gate and a forget gate**. The cell remembers values over arbitrary time intervals and the three gates regulate the flow of information into and out of the cell.
- LSTM networks are well-suited to classifying, and making predictions based on time series data.
- **LSTMs were developed to deal with the vanishing gradient problem** that can be encountered when training traditional RNNs.

LSTM



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

$\mathbf{c_t}$: Cell state, *memoría de la célula*

$\mathbf{f_t}$: Forget gate, *¿qué tanto olvidamos de c_t*

$\mathbf{o_t}$: Output gate

$\mathbf{i_t}$: Input gate

$\mathbf{i_t} * \mathbf{\tilde{C}_t}$: *La contribución de esta entrada a la memoria*

$\mathbf{h_t}$: Hidden-state vector, output

LSTM: Unfolded