Neural Networks: Long Short Term Memory Network

Andrzej Kordecki

Neural Networks (ML.ANK385 and ML.EM05): Lecture 13
Division of Theory of Machines and Robots
Institute of Aeronautics and Applied Mechanics
Faculty of Power and Aeronautical Engineering
Warsaw University of Technology

Table of Contents

Long Short Term Memory Network

Recurrent Neural networks

Recurrent neural networks (RNN) are a family of neural networks for processing sequential data.

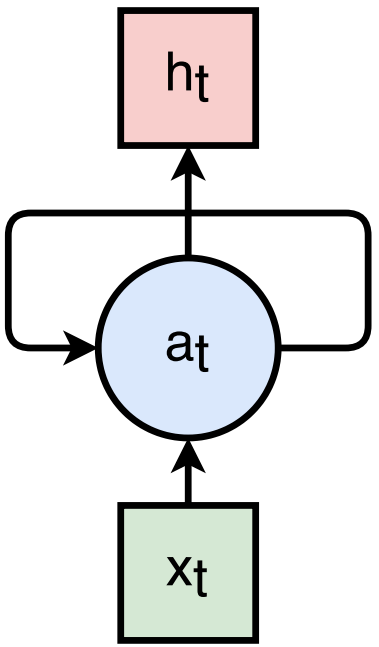
- RNN is a neural network that is specialized for processing a sequence of values $x(1), x(2), ..., x(\tau)$.
- RNN parameter sharing makes it possible to extend and apply the model to examples of different forms (e.g. different lengths) and generalize across them.

Examples of application: text analysis, genomes analysis, handwriting recognition, numerical times series data analysis emanating from sensors.

Recurrent Neural networks

The Recurrent Neural networks:

- RNN are distinguished from feedforward networks by that feedback loop connected to their past state,
- RNN preserve in the hidden state (memory) time and sequence of data information.
- In many areas RNN are similar to feedforward networks. The RNN use backpropagation algorithm in training.



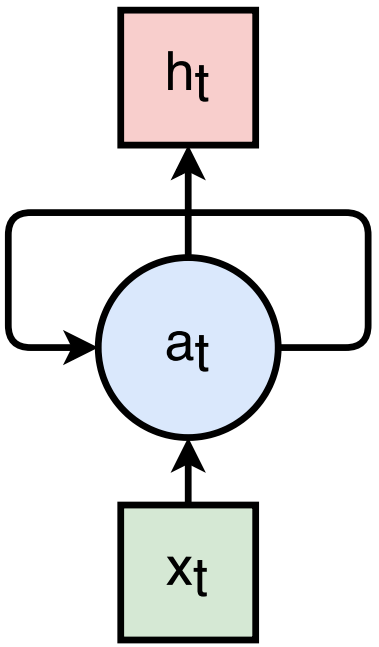
Simple RNN with 1 input, 1 output, and 1 recurrent hidden unit.

Recurrent Neural networks

The process of carrying memory forward mathematically:

The hidden state a_t at time step t:

$$a_t = \sigma(w_{xa}x_t + w_{aa}a_{t-1})$$
$$h_t = w_{ah}a_t$$

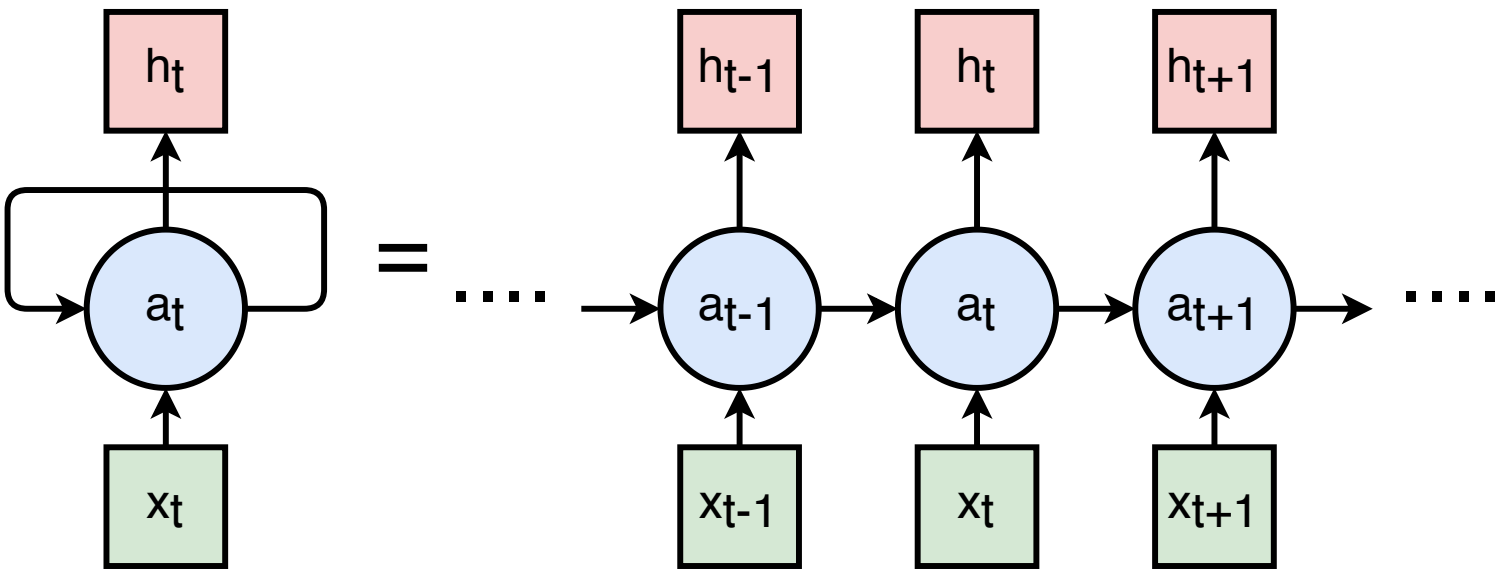


where: x_t - input at the same time step t, w - a weight matrix, σ - activation function, h_{t-1} previous time state.

Equation is recurrent because the definition of a at time t refers back to the same definition at time $t-1$.

Recurrent Neural networks

A useful way to visualise RNNs is to consider the update graph formed by 'unfolding' the network along the input sequence. Unfolding the equation by repeatedly applying the definition in this way has yielded an expression that does not involve recurrence.



Bidirectional Networks

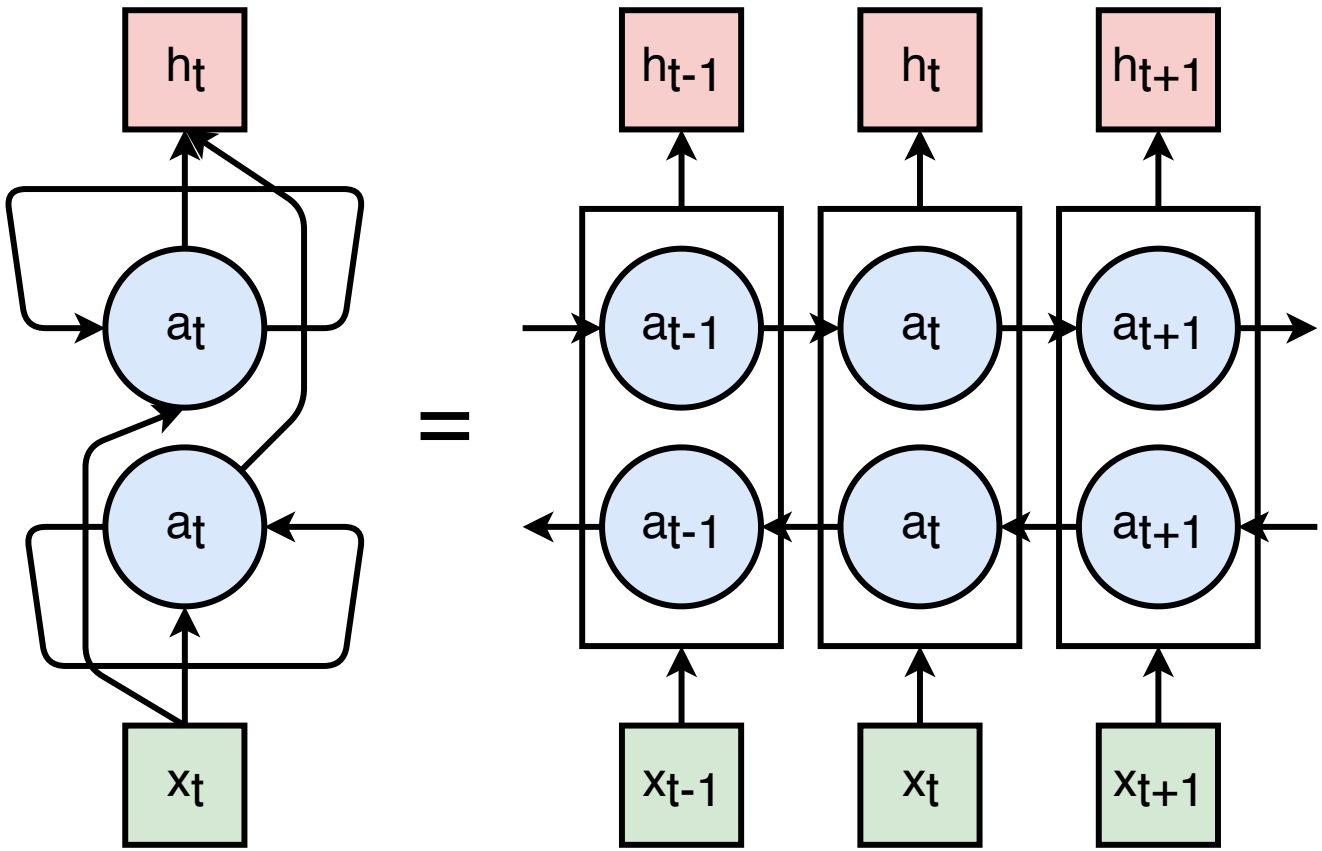
The bidirectional recurent neural networks (BRNN) have access to future as well as past context:

- In many applications we want to output a prediction of which may depend on the whole input sequence.
- The BRNN presents each training sequence forwards and backwards (opposite directions) to two separate recurrent hidden layers, both of which are connected to the same output layer.

The bidirectional RNNs combine an RNN that moves forward through time beginning from the start of the sequence with another RNN that moves backward through time beginning from the end of the sequence.

Bidirectional Networks

An unfolded bidirectional network.



Sequential Data

A RNN is a neural network that works best on sequential data:

- In the sequential data order of the data is very important e.g. the words in text (the order of the words convey context and meaning), or the average temperature for each day (ordered by time).
- The RNNs are especially can detect and capture all the nonlinear relationships in data.
- The sequential are mostly used to predict future values.

What should be the length of the sequence to remember?

Encoder-Decoder

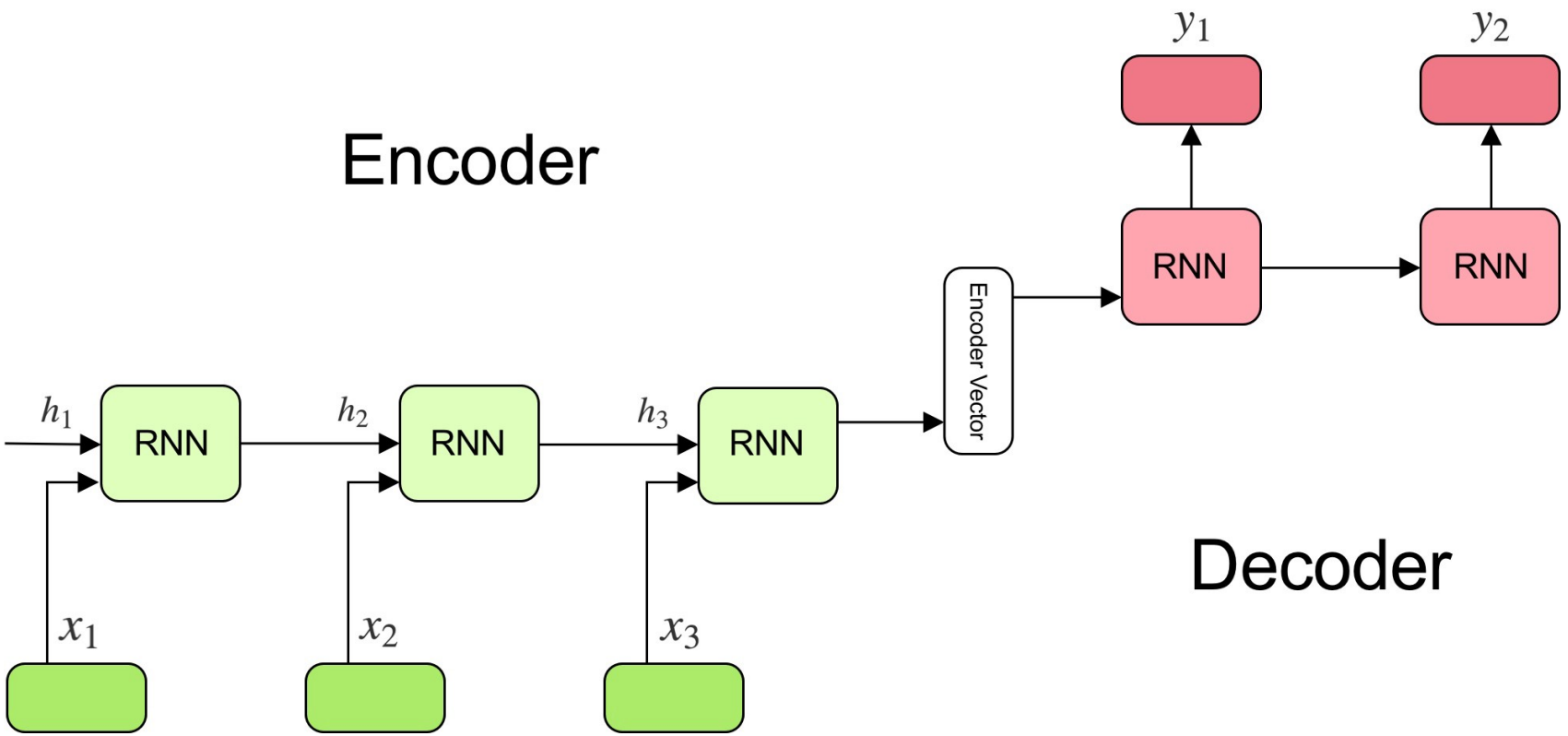
The RNN can be trained to map an input sequence to an output sequence which is not necessarily of the same length. The encoder-decoder or sequence-to-sequence architecture:

- An encoder or reader or input RNN processes the input sequence. The encoder emits the context, usually as a simple function of its final hidden state. We want to produce a representation of this context - feature map.
- A decoder or writer or output RNN is conditioned on that fixed-length vector to generate the output sequence.
- There is no constraint that the encoder must have the same size of hidden layer as the decoder.

In a sequence-to-sequence architecture, the two RNNs are trained jointly to minimize loss function.

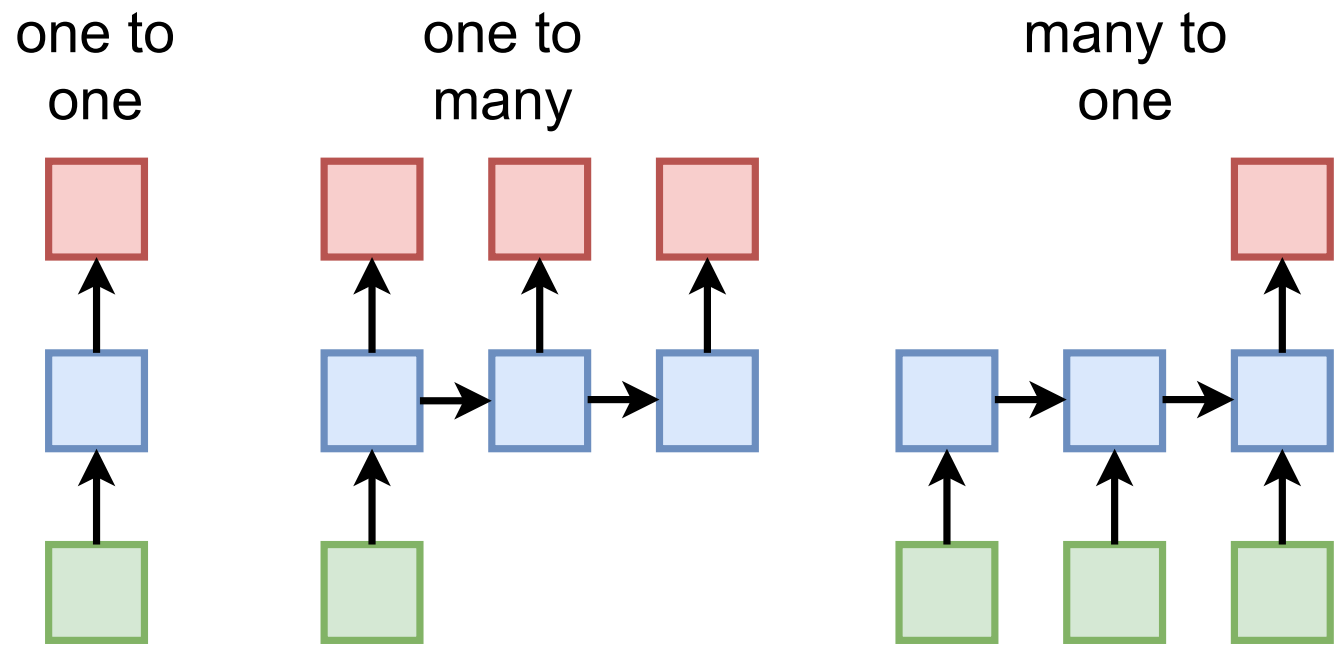
Encoder-Decoder

Example of an encoder-decoder RNN architecture



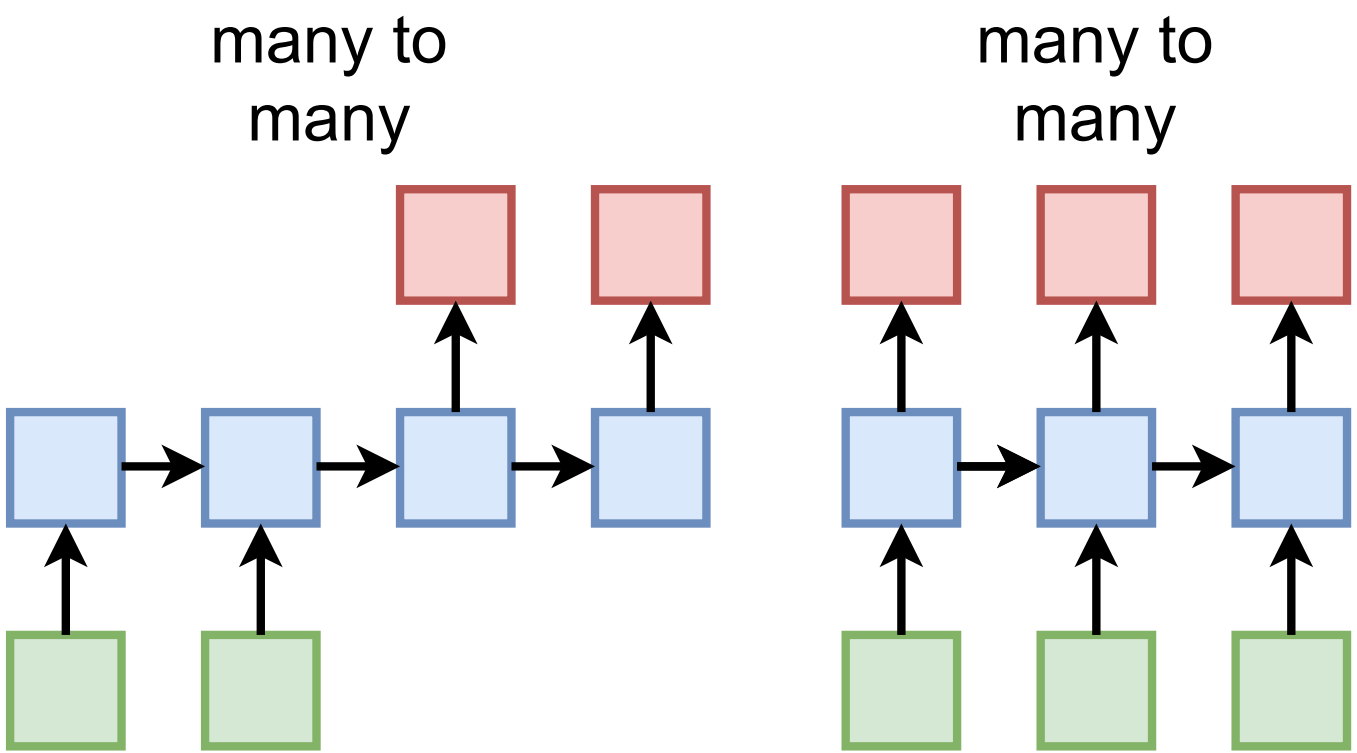
Sequential Data

The RNN work on sequences of data.



Sequential Data

The RNN work on sequences of data.



The Problem of Long-Term Dependencies

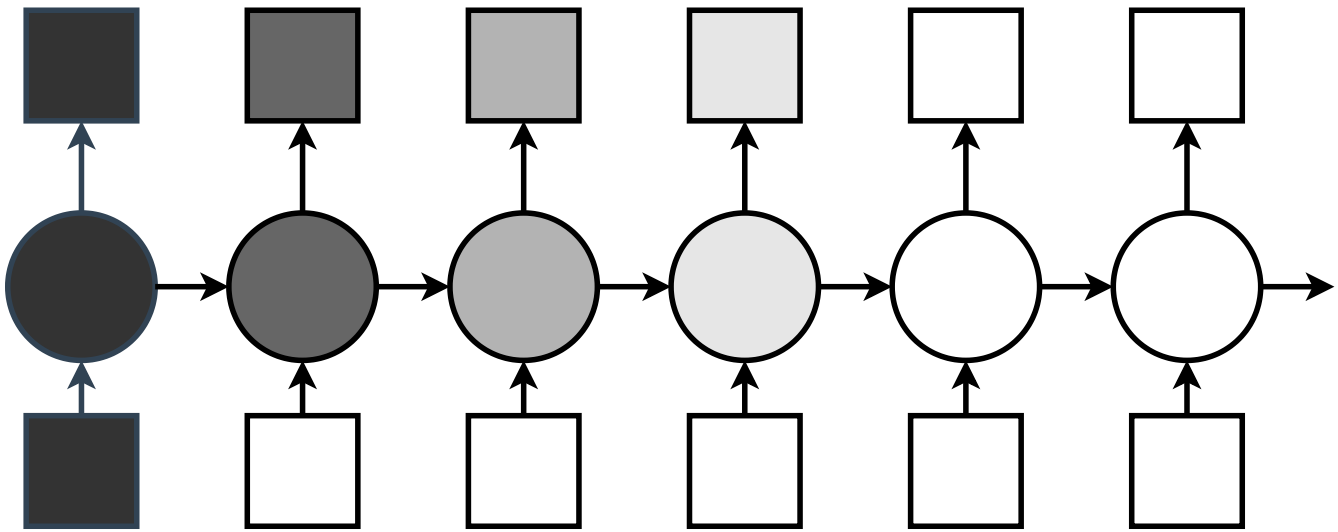
The RNN might be able to connect previous information to the present task:

- Sometimes, we only need to look at recent information to perform the present task. But there are also cases where we need more context from beginning of data.
- Can we use always RNN in such tasks?

In theory, RNNs are absolutely capable of handling such long-term dependencies. But, the problem of Vanishing and Exploding Gradients can easily occur in a deep Recurrent Neural Network. Therefore, in practice, RNNs don't seem to be able to learn them.

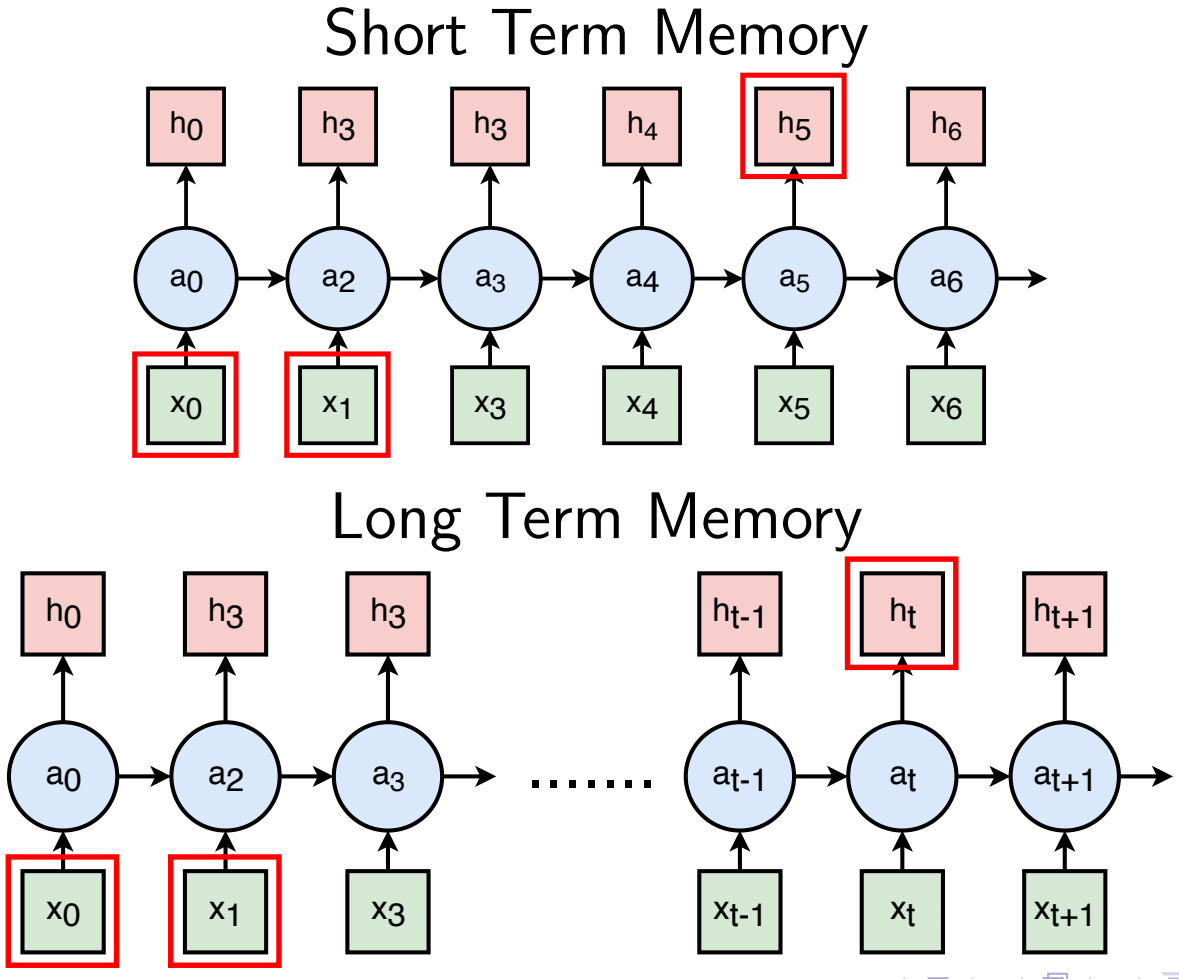
The Problem of Long-Term Dependencies

The sensitivity decays over time as new inputs overwrite the activations of the hidden layer, and the network 'forgets' the first inputs.



The Problem of Long-Term Dependencies

We can divide memory types into:



Long Short Term Memory Network

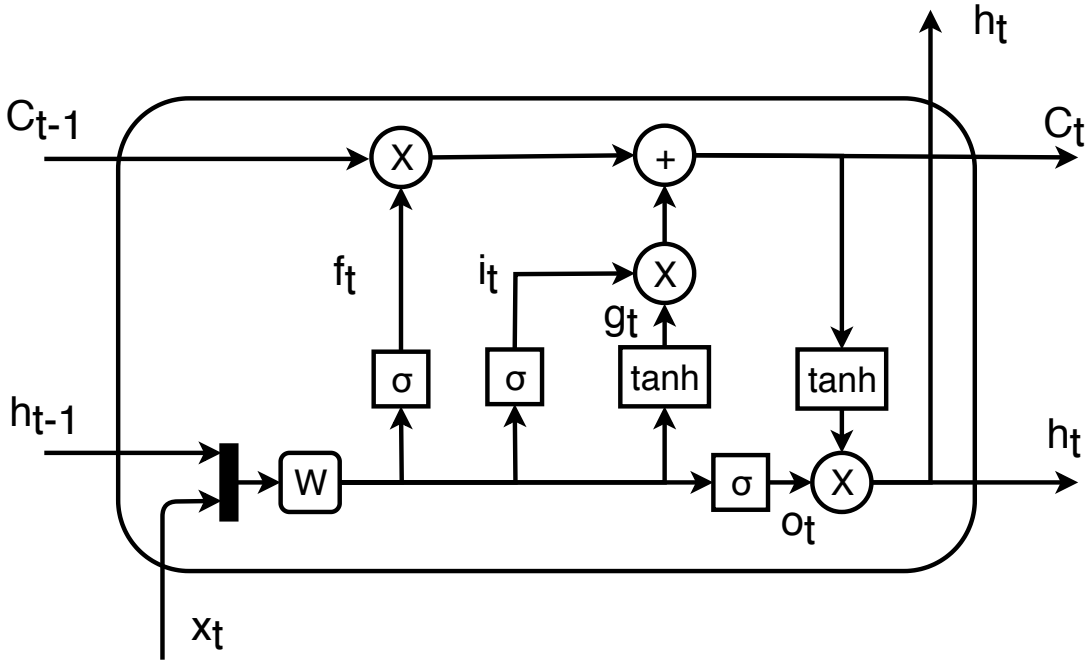
The Long Short Term Memory (LSTM) network is a special kind of RNN, capable of learning long-term dependencies.

- LSTMs have this chain like structure like all RNN, but the repeating module has a different structure.
- In concept, an LSTM recurrent unit tries to "remember" all the past knowledge that the network is seen so far and to "forget" irrelevant data. This is done by introducing different activation function layers called gates for different purposes.

Hochreiter Sepp, Jürgen Schmidhuber, "Long short-term memory", Neural computation 9.8:1735-1780, 1997

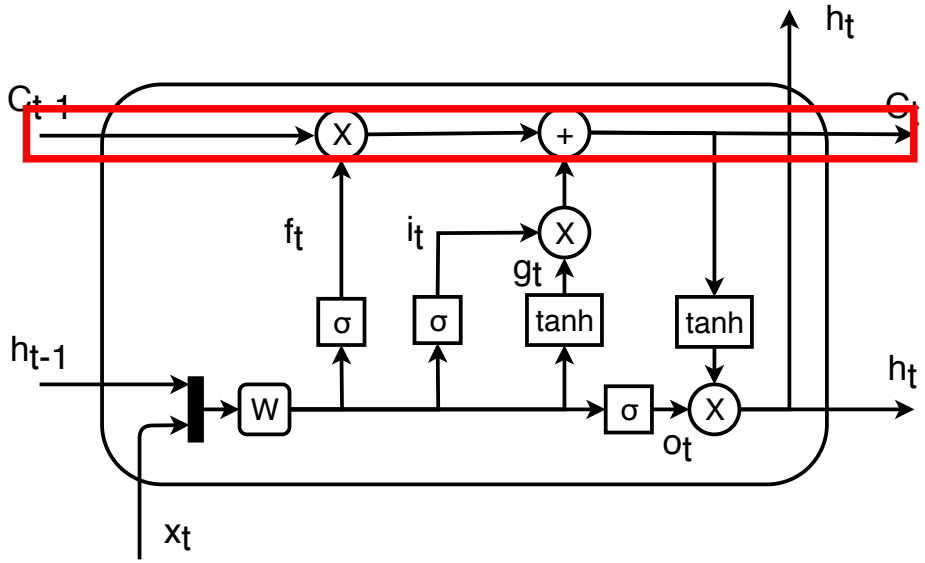
Long Short Term Memory Network

Each LSTM unit contains one or more self-connected memory cells and units that provide continuous analogues of write, read and reset.



Idea Behind LSTM

The key to LSTMs is the cell state, the horizontal line running through the top of the diagram. the Internal Cell State which conceptually describes the information that was chosen to be retained by the previous LSTM recurrent unit.



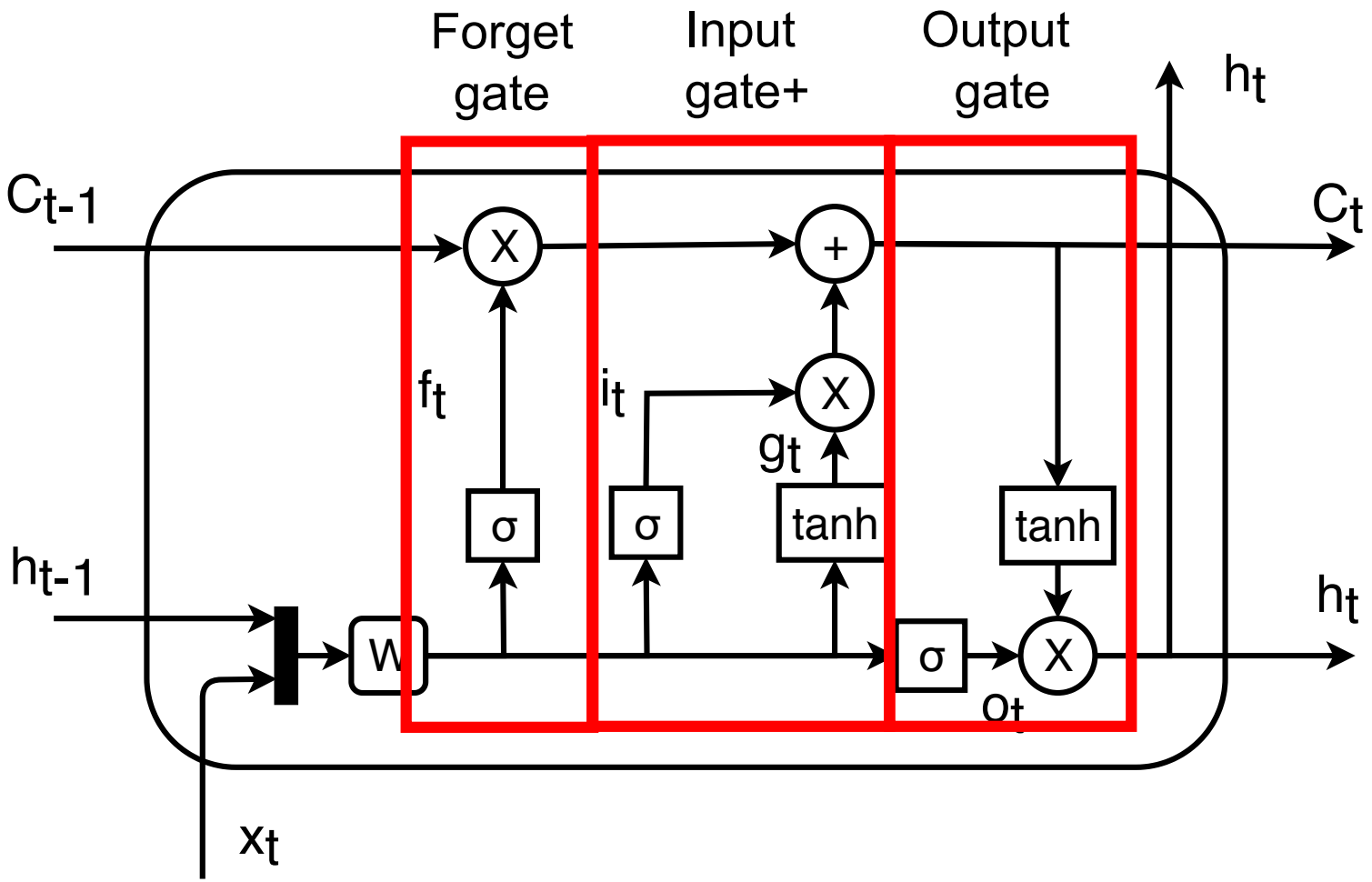
The LSTM does have the ability to remove or add information to the cell state - it is regulated by structures called gates.

Long Short Term Memory Network

The LSTM consists of four different gates for different purposes:

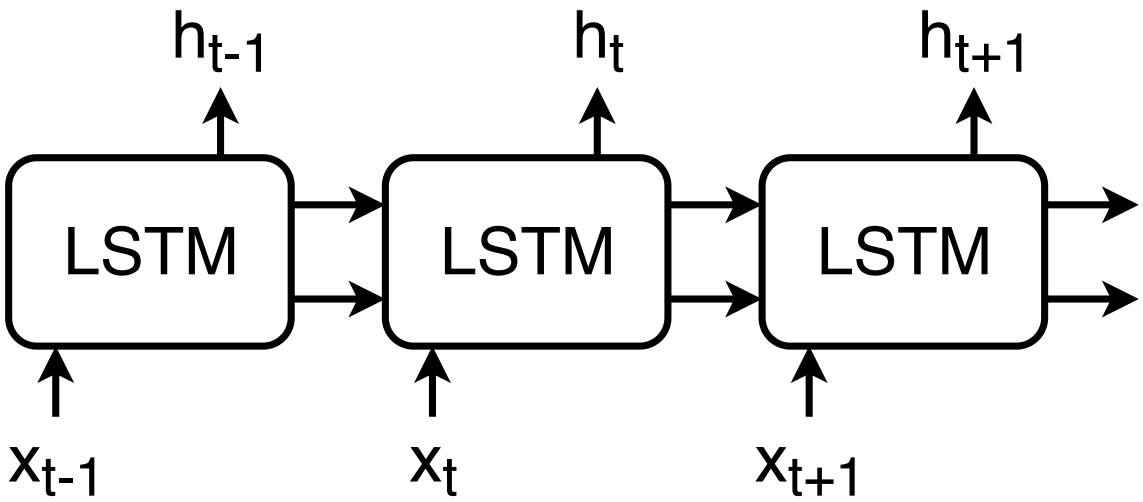
- Forget Gate f - determine to what extent forget the previous data.
- Input Gate i - determine the extent of information to be written onto the Internal Cell State.
- Input Modulation Gate (Gate gate) g - It is often considered as a sub-part of the input gate. It is used to modulate the information that the Input gate will write onto the Internal State Cell by adding non-linearity to the information and making the information Zero-mean.
- Output Gate o: It determines what output to generate from the current Internal Cell State.

Long Short Term Memory Network



Long Short Term Memory Network

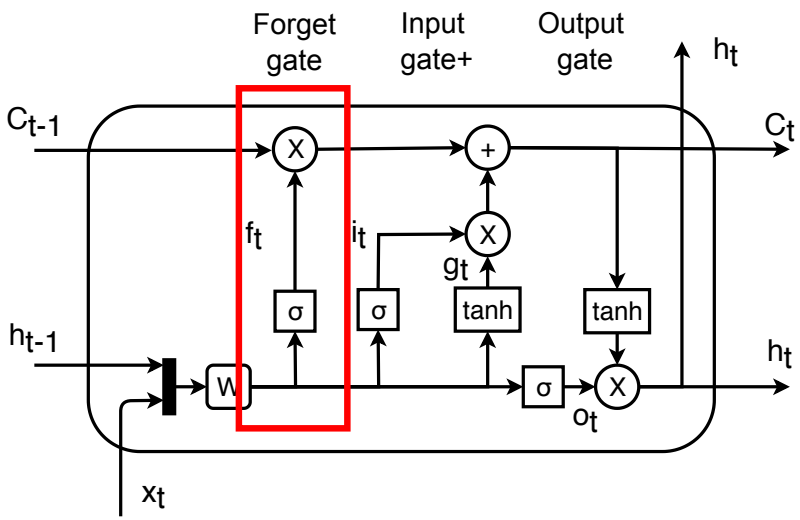
The basic work-flow of a LSTM network is similar to the work-flow of a RNN with only difference being that the Internal Cell State is also passed forward along with the Hidden State.



Unfolded LSTM network.

Long Short Term Memory Network

The first step in LSTM is to decide what information we're going to remove from the cell state. This decision is made by a sigmoid layer called the "forget gate layer."

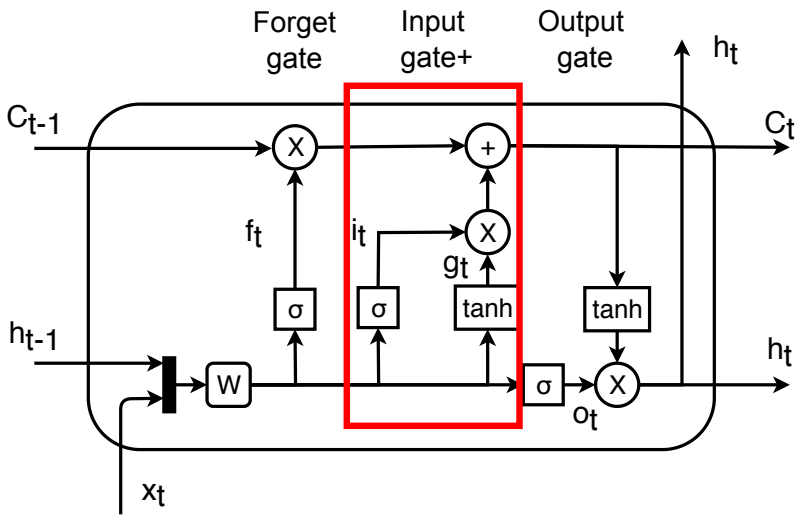


$$f_t = \sigma(W_f[h_{t-1}, x_t] + b_f)$$

The f_t outputs a number between 0 and 1 (usually the logistic function) for each number in the cell state c_{t-1}. A 1 represents information "stay", while a 0 represents "remove".

Long Short Term Memory Network

The next step is to decide what new information we're going to store in the cell state:

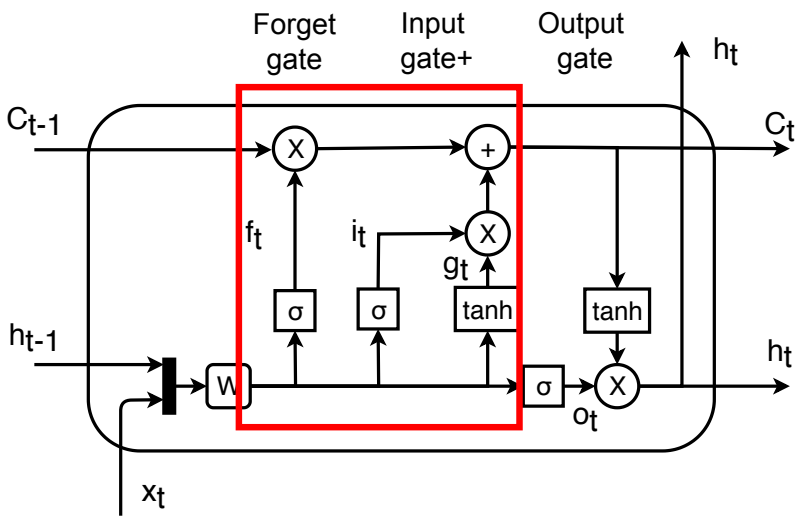


$$i_t = \sigma(W_f[h_{t-1}, x_t] + b_f)$$
$$g_t = \tanh(W_c[h_{t-1}, x_t] + b_c)$$

1. The sigmoid σ layer called the "input gate layer" decides which values we'll update.
2. The tanh layer creates a vector of new candidate values, g_t, that could be added to the state
3. We combine these two to create an update to the state.

Long Short Term Memory Network

To update the old cell state c_{t-1} into the new cell state c_t, we need to "forget" and update information.



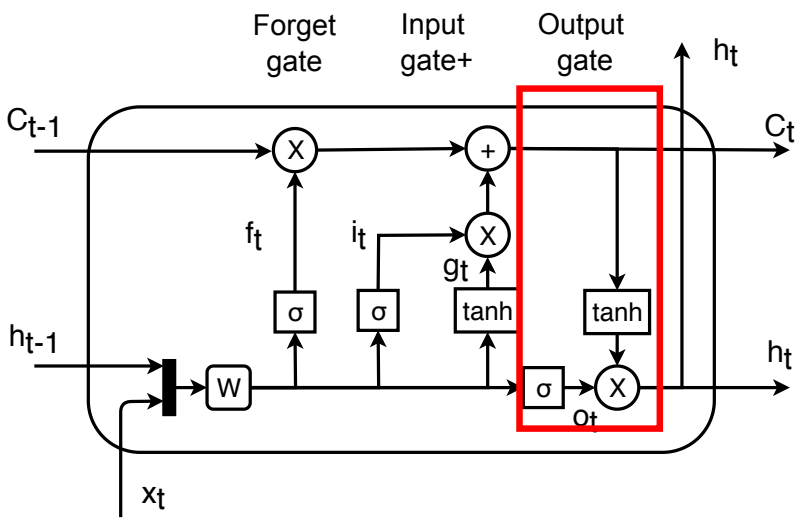
$$c_t = f_t * c_{t-1} + i_t * g_t$$

The operation on cell state have been finished.

Long Short Term Memory Network

The output will be based on our cell state:

1. The sigmoid layer "decides" what parts of the previous state we're going to output,
2. The cell state through tanh (-1 and 1) and multiply it by the output of the sigmoid gate, so that we only output the parts we decided.



$$o_t = \sigma(W_o[h_{t-1}, x_t] + b_o)$$
$$h_t = o_t * \tanh(c_t)$$

Long Short Term Memory Network

Working of an LSTM recurrent unit:

1. Take input the current input, the previous hidden state and the previous internal cell state.
2. Calculate the values of the four different gates by following the below steps:
   - For each gate, calculate the parameterized vectors for the current input and the previous hidden state by element-wise multiplication with the concerned vector with the respective weights for each gate.
   - Apply the respective activation function for each gate element-wise on the parameterized vectors.

Long Short Term Memory Network

Working of an LSTM recurrent unit:

1. Calculate the current internal cell state by first calculating the element-wise multiplication vector of the input gate and the input modulation gate, then calculate the element-wise multiplication vector of the forget gate and the previous internal cell state and then adding the two vectors.

$$c_t = f_t * c_{t-1} + i_t * g_t$$

2. Calculate the current hidden state by first taking the element-wise hyperbolic tangent of the current internal cell state vector and then performing element wise multiplication with the output gate.

$$h_t = o_t * tanh(c_t)$$

Characteristic of LSTM training

Cross entropy loss function:

$$E = \sum_j -t_j log(y_j)$$

Backpropagation algorithm weight update use gradients $\frac{\partial E}{\partial W} = \sum_j \frac{\partial E_j}{\partial W}$:

$$\frac{\partial E_t}{\partial W} = \frac{\partial E_j}{\partial y_j}\frac{\partial y_j}{\partial h_j}\frac{\partial h_j}{\partial c_j}\frac{\partial c_j}{\partial c_{j-1}}\frac{\partial c_{j-1}}{\partial c_{j-2}}.....\frac{\partial c_0}{\partial W}$$

The total error gradient:

$$\frac{\partial E}{\partial W} = \sum_j \frac{\partial E_j}{\partial y_j}\frac{\partial y_j}{\partial h_j}\frac{\partial h_j}{\partial c_j}\frac{\partial c_j}{\partial c_{j-1}}\frac{\partial c_{j-1}}{\partial c_{j-2}}.....\frac{\partial c_0}{\partial W}$$

The LSTM network use Backpropagation through time (BPTT) training method.

Characteristic of LSTM training

How does LSTM solve the problem of vanishing and exploding gradients? The value of the gradients is controlled by the chain of derivatives starting from $\frac{\partial c_t}{\partial c_{t-1}}$:

$$C_t = f_t * C_{t-1} + i_t * g_t$$

$$\frac{\partial c_t}{\partial c_{t-1}} = \frac{\partial c_t}{\partial f}\frac{\partial f}{\partial h_{t-1}}\frac{\partial h_{t-1}}{\partial c_{t-1}} + \frac{\partial c_t}{\partial i}\frac{\partial i}{\partial h_{t-1}}\frac{\partial h_{t-1}}{\partial c_{t-1}} + \frac{\partial c_t}{\partial g_t}\frac{\partial g_t}{\partial h_{t-1}}\frac{\partial h_{t-1}}{\partial c_{t-1}}$$

In the RNN, the term $\frac{\partial a_t}{\partial a_{t-1}}$ after a certain time starts to take values either greater than 1 or less than 1 but always in the same range (cause of the vanishing and exploding gradients problem). In an LSTM, the term $\frac{\partial c_t}{\partial c_{t-1}}$ does not have a fixed pattern and can take any positive value at any time step.

Summary

- RNNs allow a lot of flexibility in architecture design,
- Classic RNNs are simple but don't work very well,
- Backward flow of gradients in RNN can explode or vanish. Exploding is controlled with gradient clipping. Vanishing is controlled with additive interactions (LSTM),

Test

Organization matters

The test will be held on Thursday 27 January at 15:15 (3:15 pm)

The range of topics includes **ALL** lectures except:

- Introduction to the neural networks (1 lecture),
- Python programming (14 lecture),
- Authors and dates associated with the described methods.

Organization matters

The test rules:

- The test consists of two parts: theory questions and calculation tasks. The tasks for each part will be collected separately. There is a short break between test parts.
- Test part 1 - theory. It is **NOT allowed** to use notes, books, articles, presentations, calculator, etc.
- Test part 2 - calculation. It is **allowed** to use notes, books, articles, presentations, calculator. But, still you cannot use: forums or other forms of communications, specialized programs for neural network computations.
- The solution of a calculation task will be rated positively, if all intermediate calculations will be included in the solution.
- Independence of work is required.

Organization matters

The test consists of:

- 7 single-choice questions - each question consists of 2-5 answer options. The sum of points for one-choice tests is 14 points (2 point for every question).
- 2 tasks - solve described problem. Depending on progress of calculation, you can receive up to 11 points (3 points + 8 points). One of the tasks will be about training of the neural network.

The whole test will take 1 hour.

Test

Power engineering - Final grade:

$$G = (G_t + G_p + G_a)/25$$

where: G_t – test points in range 0-25, G_p – project points in range 0-4, G_a – activity points. Grading scale [%]:

- 0 – 50 (0 - 12.5 points), grade: 2
- 51 – 60 (12.6 - 15 points), grade: 3
- 61 – 70 (15.1 - 17.5 points), grade: 3.5
- 71 – 80 (17.6 - 20 points), grade: 4
- 81 – 90 (20.1 - 22.5 points), grade: 4.5
- 91 – 100 (22.6 - 25 points), grade: 5

To pass you have to obtain more than 50% points.

Test

EMARO and Robotics - Final grade:

$$G = 0.7 * (G_t + G_a)/25 + 0.3 * (G_{proj} + G_{prog} + G_{pres})/10$$

where: G_t – test points in range 0-25, G_a – activity points, G_{proj} – project points in range 0-4, G_{prog} – project program points in range 0-4, G_{pres} – presentation points in range -2-2.

Test

EMARO - Grading scale [%]:

- 0 – 59 (0 - 14.7 points), grade: F/FX
- 60 – 64 (14.8 - 16.2 points), grade: E
- 65 – 70 (16.3 - 17.5 points), grade: D
- 71 – 80 (17.6 - 20 points), grade: C
- 81 – 90 (20.1 - 22.5 points), grade: B
- 91 – 100 (22.6 - 25 points), grade: A

Robotics - Grading scale [%]:

- 0 – 50 (0 - 12.5 points), grade: 2
- 51 – 60 (12.6 - 15 points), grade: 3
- 61 – 70 (15.1 - 17.5 points), grade: 3.5
- 71 – 80 (17.6 - 20 points), grade: 4
- 81 – 90 (20.1 - 22.5 points), grade: 4.5
- 91 – 100 (22.6 - 25 points), grade: 5

Example

Which of the following sentences are true?

a. As a result of further growth in computer power neural networks will be able to replace the conventional algorithms.
b. Good knowledge of the problem is essential in the construction of the neural network model.
c. The operation of the trained network is predictable and well-defined.

Options:

1. (b) is true,
2. all are true.
3. none is true.

Example

Which of the following sentences are true?

a. As a result of further growth in computer power neural networks will be able to replace the conventional algorithms.
b. Good knowledge of the problem is essential in the construction of the neural network model.
c. The operation of the trained network is predictable and well-defined.

Options:

1. (b) is true,
2. all are true.
3. none is true.

Answer: (3)

Example

Is the following statement true or false? "In on-line training all data are available at the beginning and used in network training."
Options:

1. True,
2. False.

Example

Is the following statement true or false? "In on-line training all data are available at the beginning and used in network training."
Options:

1. True,
2. False.

Answer: (2)

Example

A neuron has five inputs and the weights are 1, 2, 3, 4, and 5. The activation function is linear with a factor of proportionality of 3. The inputs are (in the same order): 2, 6, 8, 12, 0. The output will be:

1. 28.67,
2. 45,
3. 120,
4. 258,

Example

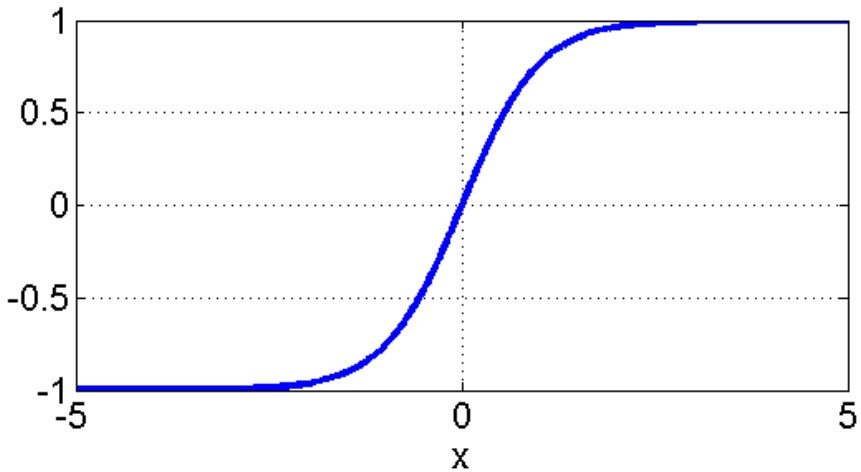
A neuron has five inputs and the weights are 1, 2, 3, 4, and 5. The activation function is linear with a factor of proportionality of 3. The inputs are (in the same order): 2, 6, 8, 12, 0. The output will be:

1. 28.67,
2. 45,
3. 120,
4. **258**,

Answer: (4) $v = 3(1 * 2 + 2 * 6 + 3 * 8 + 4 * 12 + 5 * 0) = 3(2 + 12 + 24 + 48) = 3 * 86 = 258$

Example

What is this activation function?

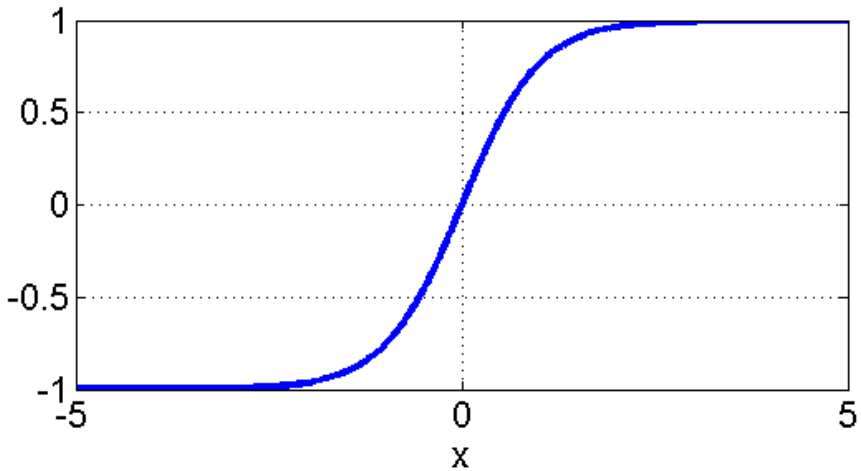


Options:

1. A threshold function,
2. A Gaussian function,
3. A sigmoid function.

Example

What is this activation function?



Options:

1. A threshold function,
2. A Gaussian function,
3. A sigmoid function.

Answer: (3)

Example

What is Sampling?
Answer:

1. Reconstruction of a continuous function,
2. Assignment of the discrete values of a continuous in time function,
3. Method of noise power reduction.

Example

What is Sampling?
Answer:

1. Reconstruction of a continuous function,
2. Assignment of the discrete values of a continuous in time function,
3. Method of noise power reduction.

Answer: (2)

Example

How we can NOT prevent overfitting?
Answer:

1. By limiting the weights values by addition of the appropriate penalty function to the objective function,
2. By using only the data without noise,
3. By stopping the training when the mean square error for data not involved in the training set starts to grow,
4. By using data augmentation methods.

Example

How we can NOT prevent overfitting?
Answer:

1. By limiting the weights values by addition of the appropriate penalty function to the objective function,
2. By using only the data without noise,
3. By stopping the training when the mean square error for data not involved in the training set starts to grow,
4. By using data augmentation methods.

Answer: (2)

Example

How can we improve Radial Basis Function network (RBF):

a. adding additional hidden layers,
b. use smaller number of basis functions than training data points,
c. using kernel functions.

Answer:

1. (a) and (c) are true,
2. (b) is true,
3. (c) is true,

Example

How can we improve Radial Basis Function network (RBF):

a. adding additional hidden layers,
b. use smaller number of basis functions than training data points,
c. using kernel functions.

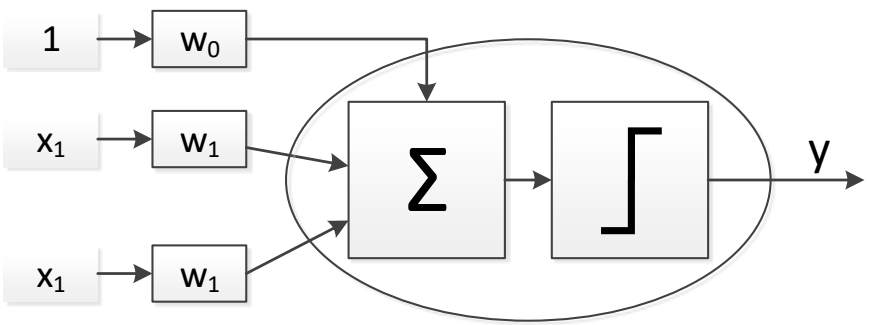
Answer:

1. (a) and (c) are true,
2. (b) is true,
3. (c) is true,

Answer: (2)

Example 1

Determine the bias (w_0) and weights (w_1 and w_2) of one McCulloch-Pitts neuron for implementation of logic operation AND:



x_1	x_2	y
0	0	0
0	1	0
1	0	0
1	1	1

Example 1

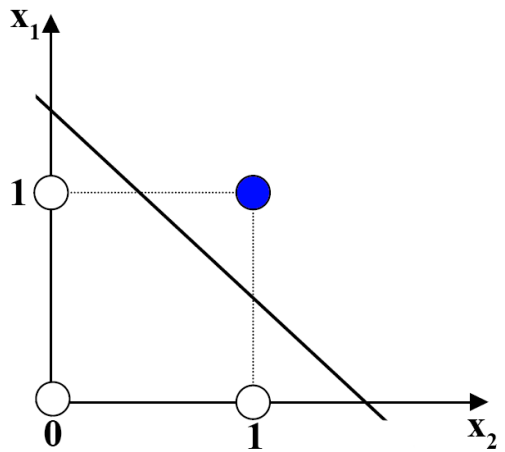
Answer: Solving set of equations:

x_1	x_2	y
0	0	0
0	1	0
1	0	0
1	1	1

$$
\begin{aligned}
w_1 0 + w_2 0 + w_0 &< 0 \\
w_1 0 + w_2 1 + w_0 &< 0 \\
w_1 1 + w_2 0 + w_0 &< 0 \\
w_1 1 + w_2 1 + w_0 &\geq 0
\end{aligned}
$$

$$
\begin{aligned}
w_0 &< 0 \\
w_1 &< -w_0 \\
w_2 &< -w_0 \\
w_1 + w_2 &\geq -w_0
\end{aligned}
$$

Or use decision boundaries:



Example: $w_1 = 1,\ w_2 = 1,\ w_0 = -1.5$

Task: Calculate the SVM loss an specify which examples are correctly classified:

$$L_i = \sum_{j \neq y_i} \max(0, 1 + s_j - s_{y_i})$$

Outputs of neural network for classes plane, car and ship:

Input:	ex01:P	ex02:P	ex03:C	ex04:S	ex05:S	ex06:S
plane	**-0.5**	**4**	1	1	-2	-3
car	1.5	2	**1.5**	0.5	-1	-4
ship	1	2	-1	**-2**	1	5

Description: ex01:P, ex02:P - plane example; ex03:C - car example; ex04:S, ex05:S, ex06:S - ship example.

Solution:

- Sample ex01/plane class

$$L_1 = max(0, (1.5-(-0.5)+1)+max(0, (1-(-0.5)+1) = 5.5$$

The sample was incorrectly classified into the car class.

- Sample ex03/car class

$$L_1 = max(0, (1 - 1.5 + 1) + max(0, (-1 - 1.5 + 1) = 0.5$$

The sample was correctly classified into the car class.

- Sample ex05/ship class

$$L_1 = max(0, (-2 - 1 + 1) + max(0, (-1 - 1 + 1) = 0$$

The sample was correctly classified into the ship class.

Questions