VIETNAM GENERAL CONFEDERATION OF LABOR

TON DUC THANG UNIVERSITY

INFORMATION TECHNOLOGY FACULTY

HOMEWORK 2

DEEP LEARNING

LSTM and GRU

Instructor: **Prof. Lê Anh Cường**

Name: **Đỗ Phạm Quang Hưng**

Student ID: **520K0127**

HO CHI MINH CITY, 2023

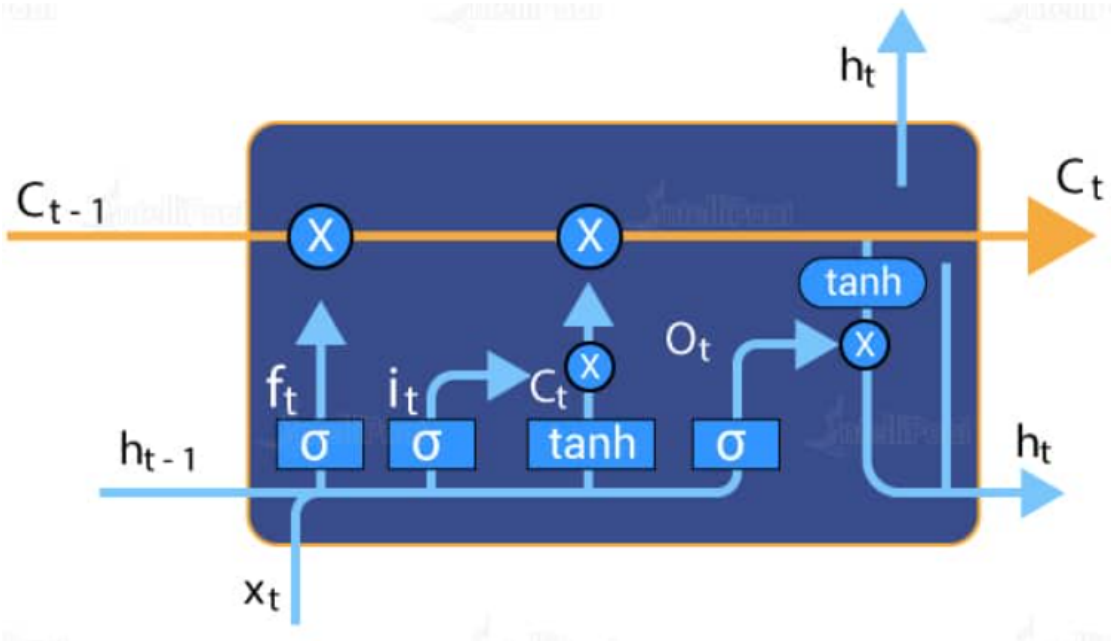
Table of Content

I. Long Short-Term Memory (LSTM)

'What is LSTM?' First, you must be wondering 'What does LSTM stand for?' LSTM stands for long short-term memory networks, used in the field of Deep Learning. It is a variety of **recurrent neural networks (RNNs)** that are capable of learning long-term dependencies, especially in sequence prediction problems. LSTM has feedback connections, i.e., it is capable of processing the entire sequence of data, apart from single data points such as images. This finds application in speech recognition, machine translation, etc. LSTM is a special kind of RNN, which shows outstanding performance on a large variety of problems.

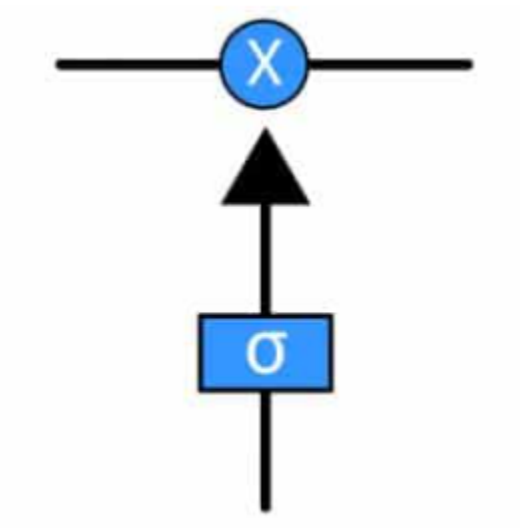
The Logic Behind LSTM

The central role of an LSTM model is held by a memory cell known as a 'cell state' that maintains its state over time. The cell state is the horizontal line that runs through the top of the below diagram. It can be visualized as a conveyor belt through which information just flows, unchanged.



Information can be added to or removed from the cell state in LSTM and is regulated by gates. These gates optionally let the information flow in and out of the cell. It contains a

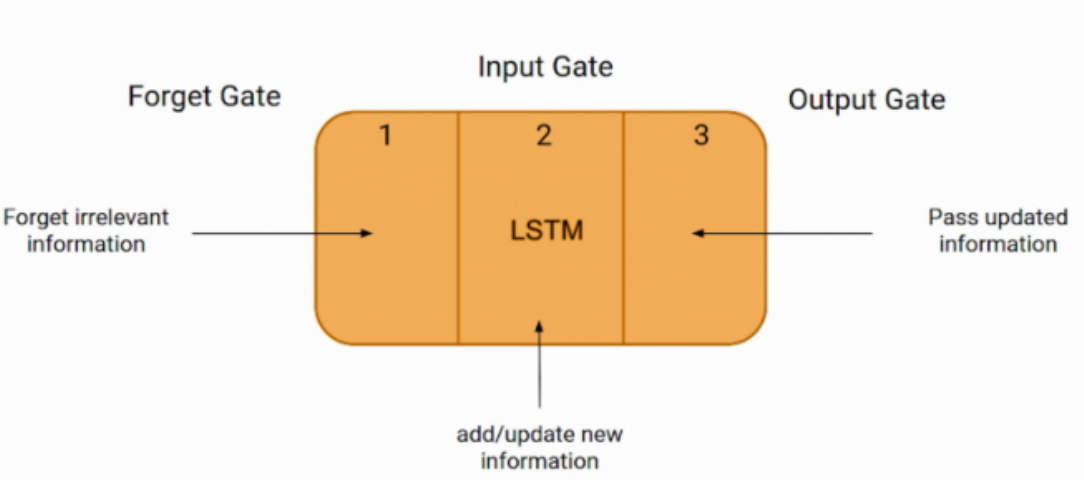
pointwise multiplication operation and a sigmoid neural net layer that assist the mechanism.



The sigmoid layer gives out numbers between zero and one, where zero means ‘nothing should be let through,’ and one means ‘everything should be let through.’

The problem with Recurrent Neural Networks is that they have a short-term memory to retain previous information in the current neuron. However, this ability decreases very quickly for longer sequences. As a remedy for this, the LSTM models were introduced to be able to retain past information even longer.

The problem with Recurrent Neural Networks is that they simply store the previous data in their “short-term memory”. Once the memory in it runs out, it simply deletes the longest retained information and replaces it with new data. The LSTM model attempts to escape this problem by retaining only selected information in short-term memory.



For this purpose, the LSTM architecture consists of a total of three different stages:

1. In the so-called Forget Gate, it is decided which current and previous information are kept and which are thrown out. This includes the hidden status from the previous run and the current status. These values are passed into a sigmoid function, which can only output values between 0 and 1. The value 0 means that all previous information is forgotten and 1 accordingly that all previous information is kept.

Forget Gate:

- $f_t = \sigma (x_t * U_f + H_{t-1} * W_f)$

Let's try to understand the equation, here

- Xt: input to the current timestamp.
- Uf: weight associated with the input
- Ht-1: The hidden state of the previous timestamp
- Wf: It is the weight matrix associated with hidden state

Later, a sigmoid function is applied over it. That will make ft a number between 0 and 1. This ft is later multiplied with the cell state of the previous timestamp as shown below.

2. In the Input Gate, it is decided how valuable the current input is to solve the task. For this purpose, the current input is multiplied by the hidden state and the weight matrix of the last run.

Input Gate:

- $i_t = \sigma(x_t * U_i + H_{t-1} * W_i)$

Here,

- Xt: Input at the current timestamp t
- Ui: weight matrix of input
- Ht-1: A hidden state at the previous timestamp
- Wi: Weight matrix of input associated with hidden state

3. In the Output Gate, the output of the LSTM model is calculated. Depending on the application, it can be, for example, a word that complements the meaning of the sentence.

Output Gate:

- $o_t = \sigma(x_t * U_o + H_{t-1} * W_o)$

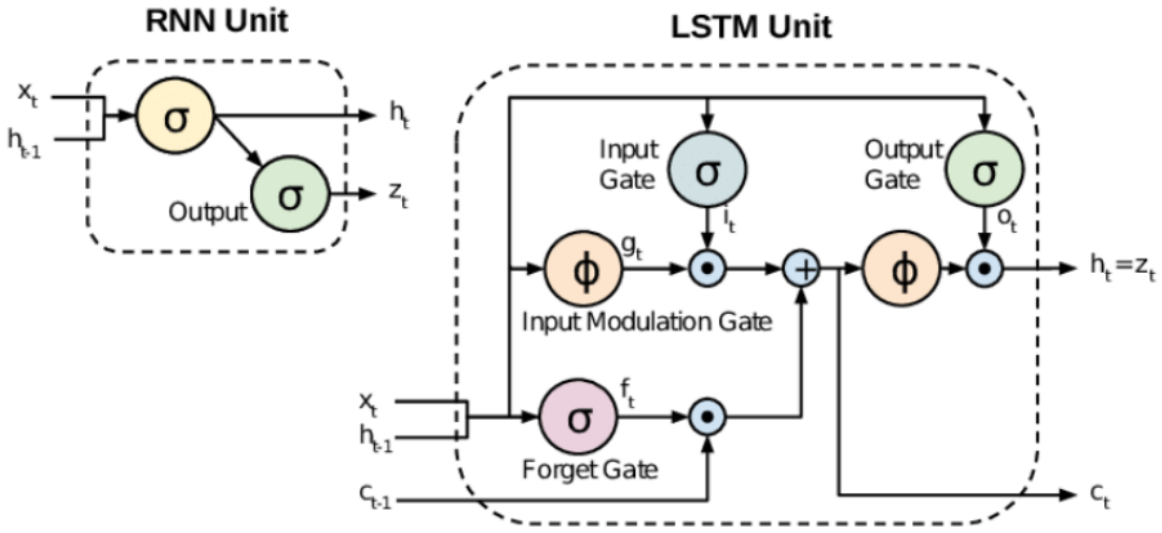
Its value will also lie between 0 and 1 because of this sigmoid function. Now to calculate the current hidden state we will use Ot and tanh of the updated cell state. As shown below.

$$H_t = o_t * \tanh(C_t)$$

It turns out that the hidden state is a function of Long term memory (Ct) and the current output. If you need to take the output of the current timestamp just apply the SoftMax activation on hidden state Ht.

$$\text{Output} = \text{Softmax}(H_t)$$

LSTM vs RNN



The main difference between RNN and LSTM is in terms of which one maintain information in the memory for a long period of time. Here LSTM has an advantage over RNN as LSTM can handle the information in memory for a long period of time as compared to RNN. But the question is what is different in LSTM than RNN by which LSTMs are capable of maintaining long-term temporal dependencies (remembering information for a long period of time).

RNNs have recurrent connections and/or layers

You can describe a recurrent neural network (RNN) or a long short-term memory (LSTM), depending on the context, at different levels of abstraction. For example, you could say that an RNN is any neural network that contains one or more recurrent (or cyclic) connections. Or you could say that layer ll of neural network NN is a recurrent layer, given that it contains units (or neurons) with recurrent connections, but NN may not contain only recurrent layers (for example, it may also be composed of feedforward layers, i.e. layers with units that contain only feedforward connections).

In any case, a recurrent neural network is almost always described as a neural network (NN) and not as a layer (this should also be obvious from the name).

LSTM can refer to a unit, layer, or neural network

On the other hand, depending on the context, the term "LSTM" alone can refer to an

- LSTM unit (or neuron),
- an LSTM layer (many LSTM units), or
- an LSTM neural network (a neural network with LSTM units or layers).

People may also refer to neural networks with LSTM units as LSTMs (plural version of LSTM).

LSTMs are RNNs

An LSTM unit is a recurrent unit, that is, a unit (or neuron) that contains cyclic connections, so an LSTM neural network is a recurrent neural network (RNN).

LSTM units/neurons

The main difference between an LSTM unit and a standard RNN unit is that the LSTM unit is *more sophisticated*. More precisely, it is composed of the so-called **gates** that supposedly regulate better the flow of information through the unit.

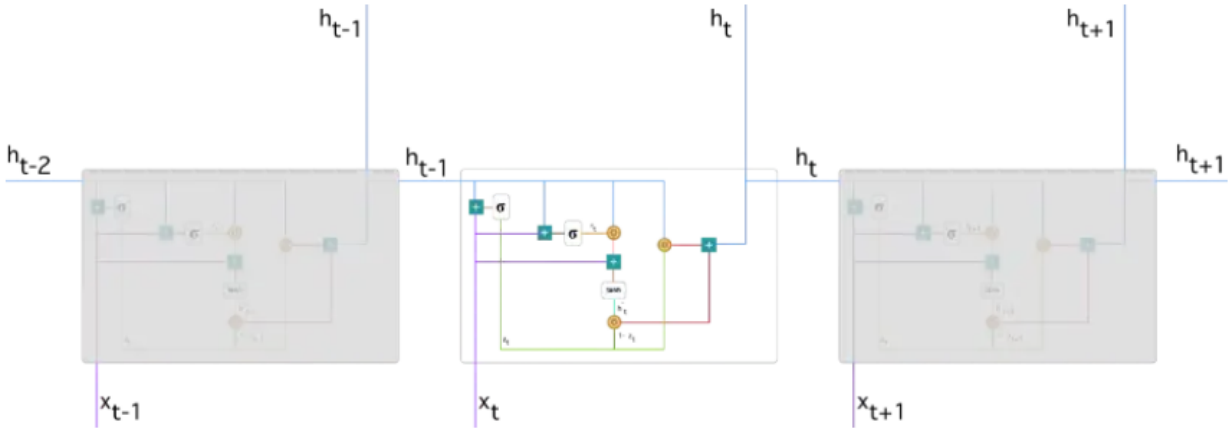
II. Gated Recurrent Unit (GRU)

GRU (Gated Recurrent Unit) aims to solve the **vanishing gradient problem** which comes with a standard recurrent neural network. GRU can also be considered as a variation on the LSTM because both are designed similarly and, in some cases, produce equally excellent results.

As mentioned above, GRUs are improved versions of standard recurrent neural networks. But what makes them so special and effective?

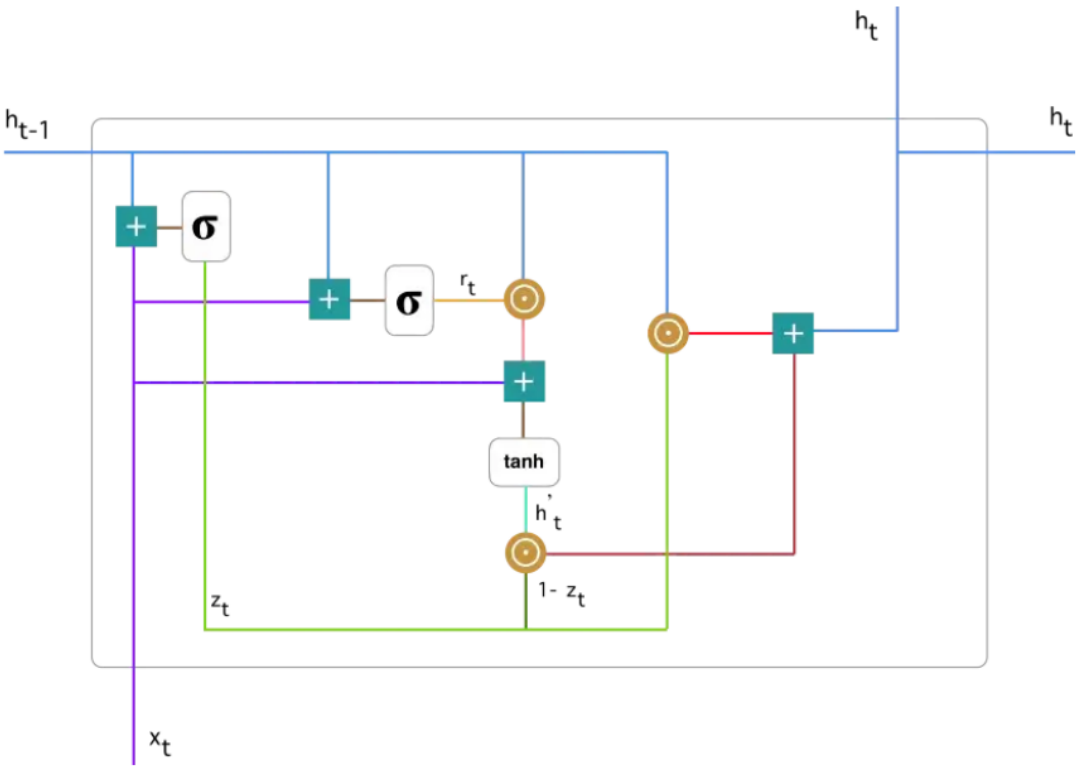
To solve the vanishing gradient problem of a standard RNN, GRU uses, the so-called, **update gate and reset gate**. Basically, these are two vectors that decide what information should be passed to the output. The special thing about them is that they can be trained to keep information from long ago, without washing it through time or removing information that is irrelevant to the prediction.

To explain the mathematics behind that process we will examine a single unit from the following recurrent neural network:



Recurrent neural network with Gated Recurrent Unit

Here is a more detailed version of that single GRU:



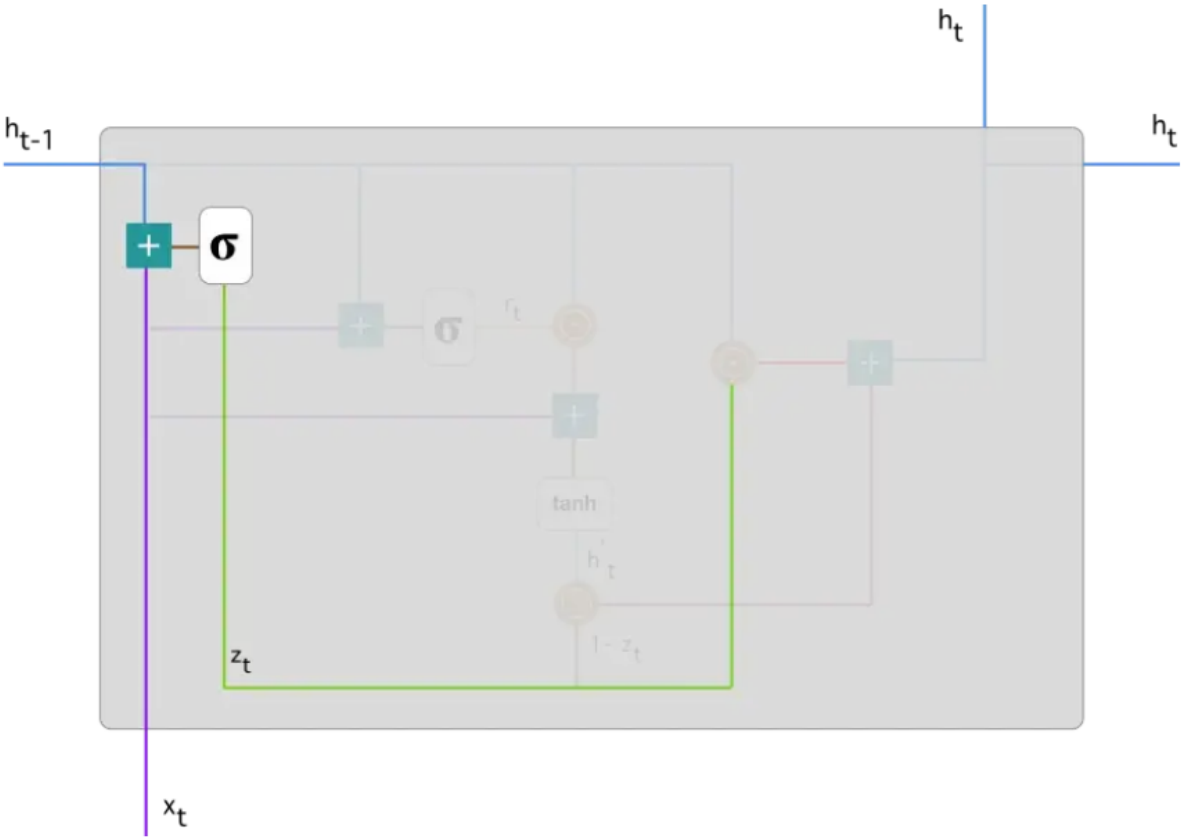
Gated Recurrent Unit

2.1. Update gate

We start with calculating the **update gate z_t for time step t** using the formula:

$$z_t = \sigma(W^{(z)}x_t + U^{(z)}h_{t-1})$$

When *x_t* is plugged into the network unit, it is multiplied by its own weight *W(z)*. The same goes for *h_(t-1)* which holds the information for the previous *t-1* units and is multiplied by its own weight *U(z)*. Both results are added together and a sigmoid activation function is applied to squash the result between 0 and 1.



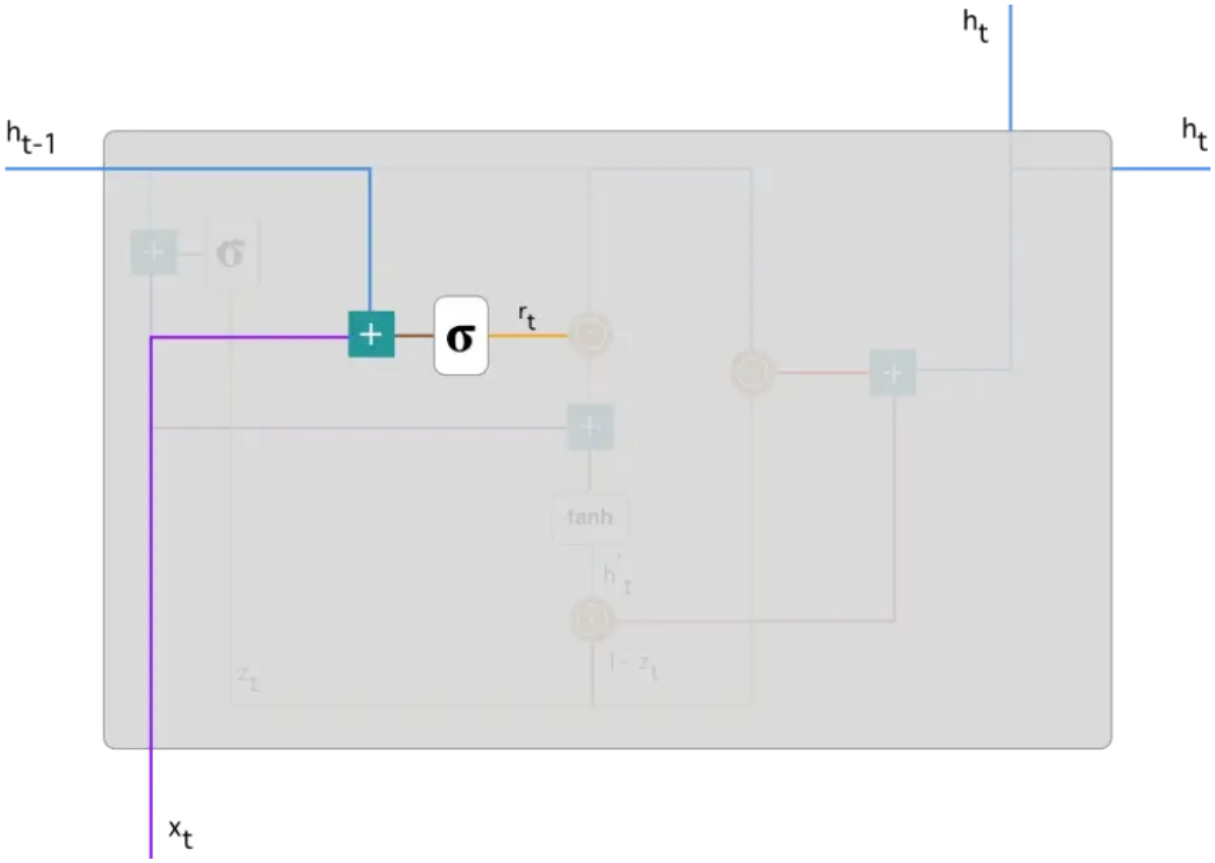
The update gate helps the model to determine how much of the past information (from previous time steps) needs to be passed along to the future. That is really powerful because the model can decide to copy all the information from the past and eliminate the risk of the vanishing gradient problem.

2.2. Reset gate

Essentially, **this gate is used from the model to decide how much of the past information to forget**. To calculate it, we use:

$$r_t = \sigma(W^{(r)}x_t + U^{(r)}h_{t-1})$$

This formula is the same as the one for the update gate. The difference comes in the weights and the gate's usage, which will see in a bit. The schema below shows where the reset gate is:



As before, we plug in h_(t-1) — blue line and x_t — purple line, multiply them with their corresponding weights, sum the results and apply the sigmoid function.

2.3. Current memory content

Let's see how exactly the gates will affect the final output. First, we start with the usage of the reset gate. We introduce a new memory content which will use the reset gate to store the relevant information from the past. It is calculated as follows:

$$h_t^{'} = \tanh(Wx_t + r_t \odot Uh_{t-1})$$

2.4. Final memory at current time step

As the last step, the network needs to calculate h_t — vector which holds information for the current unit and passes it down to the network. In order to do that the update gate is needed. It determines what to collect from the current memory content — h'_t and what from the previous steps — h_(t-1). That is done as follows:

$$h_t = z_t \odot h_{t-1} + (1 - z_t) \odot h_t'$$

1. Apply element-wise multiplication to the update gate *z_t* and *h_(t-1)*.
2. Apply element-wise multiplication to *(1-z_t)* and *h'_t*.
3. Sum the results from step 1 and 2.