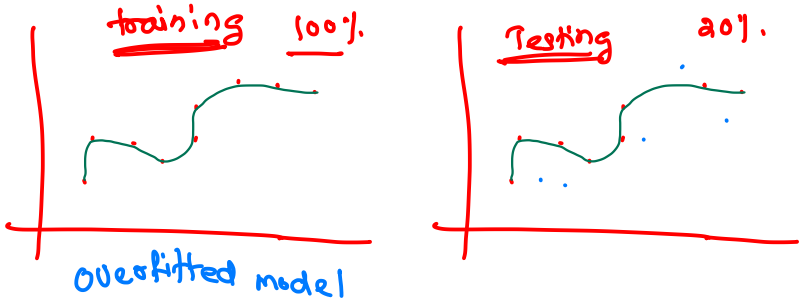
Machine Learning


RNN
ANN

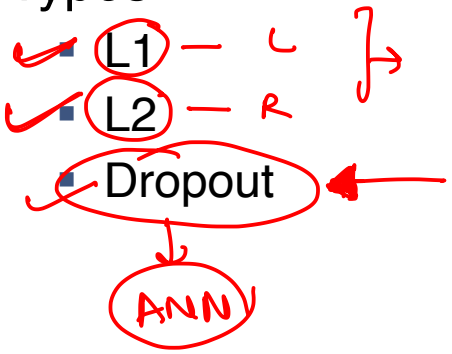
Problem With Overfitting

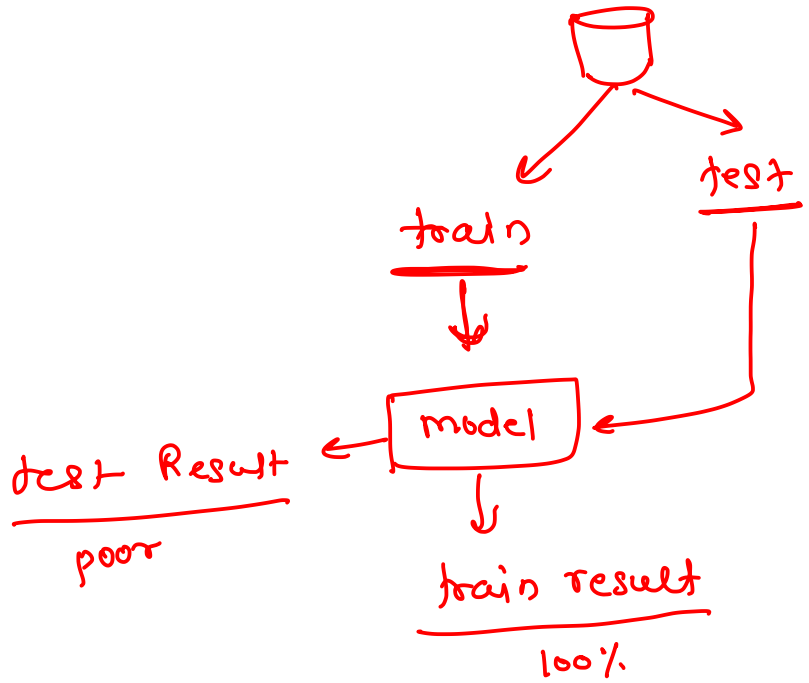
- Large neural nets trained on relatively small datasets can overfit the training data.
- This has the effect of the model learning the statistical noise in the training data, which results in poor performance when the model is evaluated on new data, e.g. a test dataset.
- Generalization error increases due to overfitting.
- One approach to reduce overfitting is to fit all possible different neural networks on the same dataset and to average the predictions from each model. This is not feasible in practice, and can be approximated using a small collection of different models, called an ensemble.
- A problem even with the ensemble approximation is that it requires multiple models to be fit and stored, which can be a challenge if the models are large, requiring days or weeks to train and tune



Problem Solution - Regularization

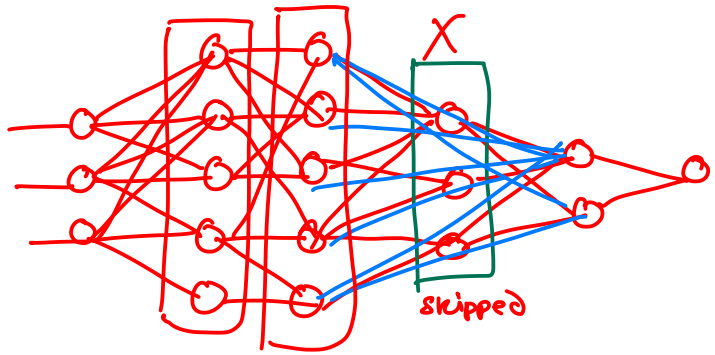
- Generalization error: Performance on inputs not previously seen
- Any modification to a learning algorithm to reduce its generalization error but not its training error
- Reduce generalization error even at the expense of increasing training error
- Types
  - L1 — L
  - L2 — R
  - Dropout





Problem Solution - Dropout

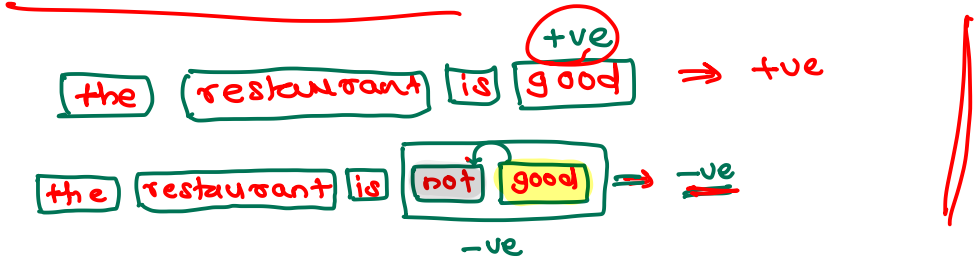
- Dropout is a regularization method that approximates training a large number of neural networks with different architectures in parallel.
- During training, some number of layer outputs are randomly ignored or "*dropped out*."
- This has the effect of making the layer look-like and be treated-like a layer with a different number of nodes and connectivity to the prior layer
- In effect, each update to a layer during training is performed with a different "*view*" of the configured layer.
- *By dropping a unit out, we mean temporarily removing it from the network, along with all its incoming and outgoing connections*



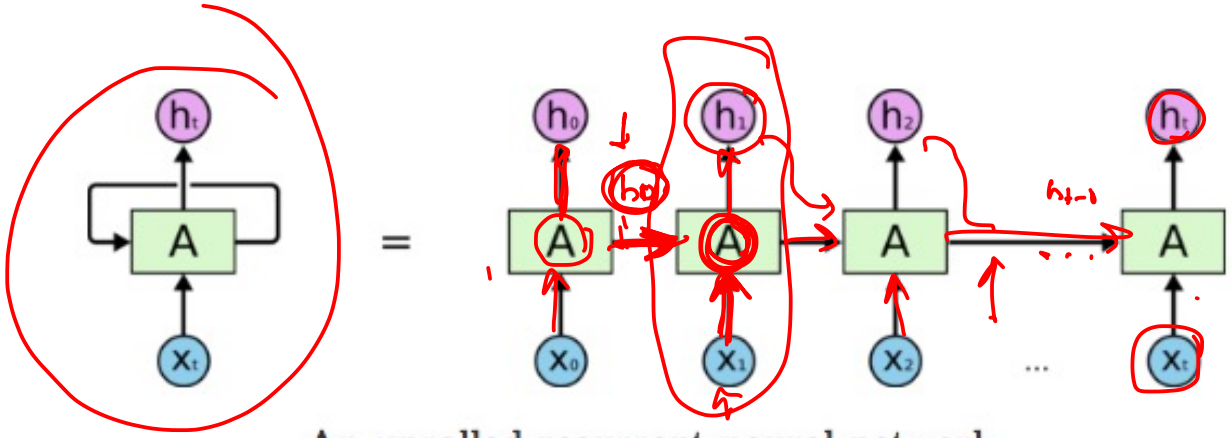
RNN

Overview

- Recurrent Neural Network is a generalization of feedforward neural network that has an internal memory
- RNN is recurrent in nature as it performs the same function for every input of data while the output of the current input depends on the past one computation
- After producing the output, it is copied and sent back into the recurrent network
- For making a decision, it considers the current input and the output that it has learned from the previous input
- Unlike feedforward neural networks, RNNs can use their internal state (memory) to process sequences of inputs
- This makes them applicable to tasks such as unsegmented, connected handwriting recognition or speech recognition



RNN



An unrolled recurrent neural network.

RNN

- First, it takes the X(0) from the sequence of input and then it outputs h(0) which together with X(1) is the input for the next step
- So, the h(0) and X(1) is the input for the next step
- Similarly, h(1) from the next is the input with X(2) for the next step and so on
- This way, it keeps remembering the context while training

RNN

- The formula for current state is

$$h_t = f(h_{t-1}, x_t)$$

output generated in the layer before

- Activation Function

$$h_t = \tanh (W_{hh} h_{t-1} + W_{xh} x_t)$$

output of last layer

input of this layer

ANN

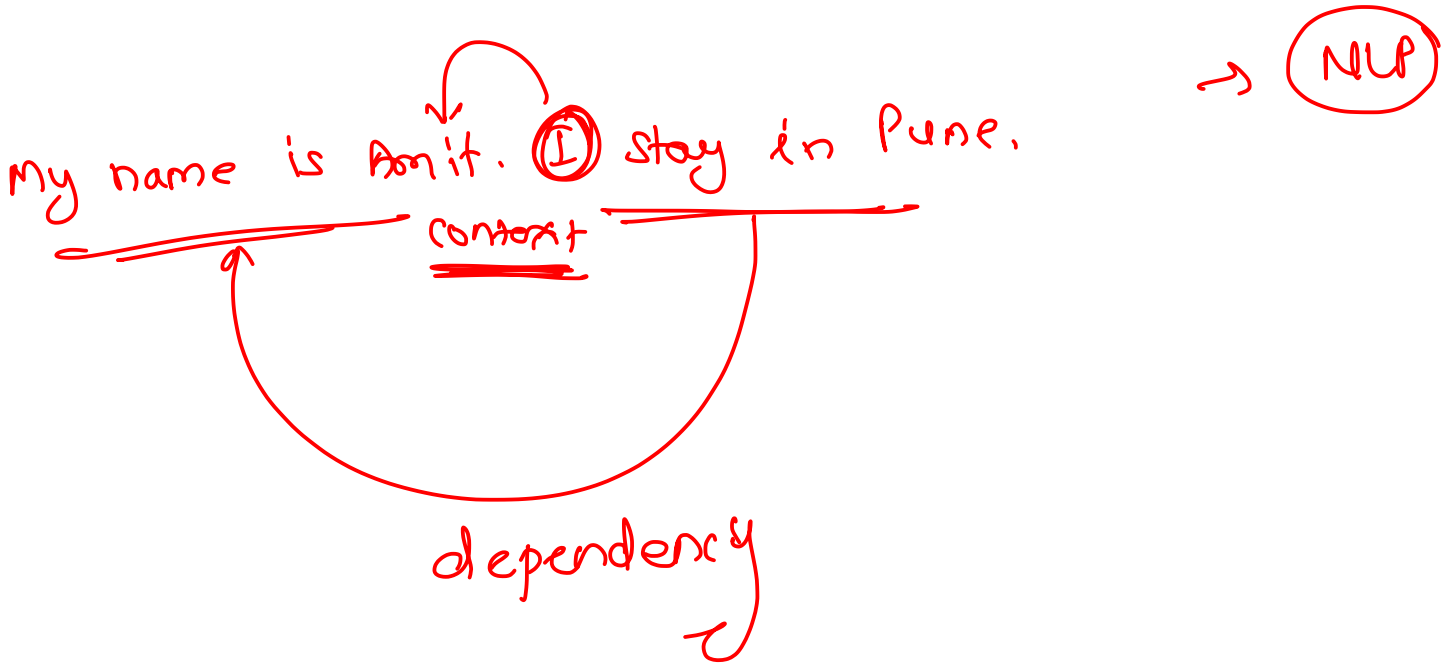
$\Rightarrow \tanh(w_i x_i)$

- Output

$$y_t = W_{hy} h_t$$

Advantages of Recurrent Neural Network

- **RNN** can model sequence of data so that each sample can be assumed to be dependent on previous ones
- Recurrent neural network are even used with convolutional layers to extend the effective pixel neighbourhood



Disadvantages of Recurrent Neural Network

- Gradient vanishing and exploding problems
- Training an RNN is a very difficult task → training of one layer is dependent of previous layer
- It cannot process very long sequences if using *tanh* or *relu* as an activation function

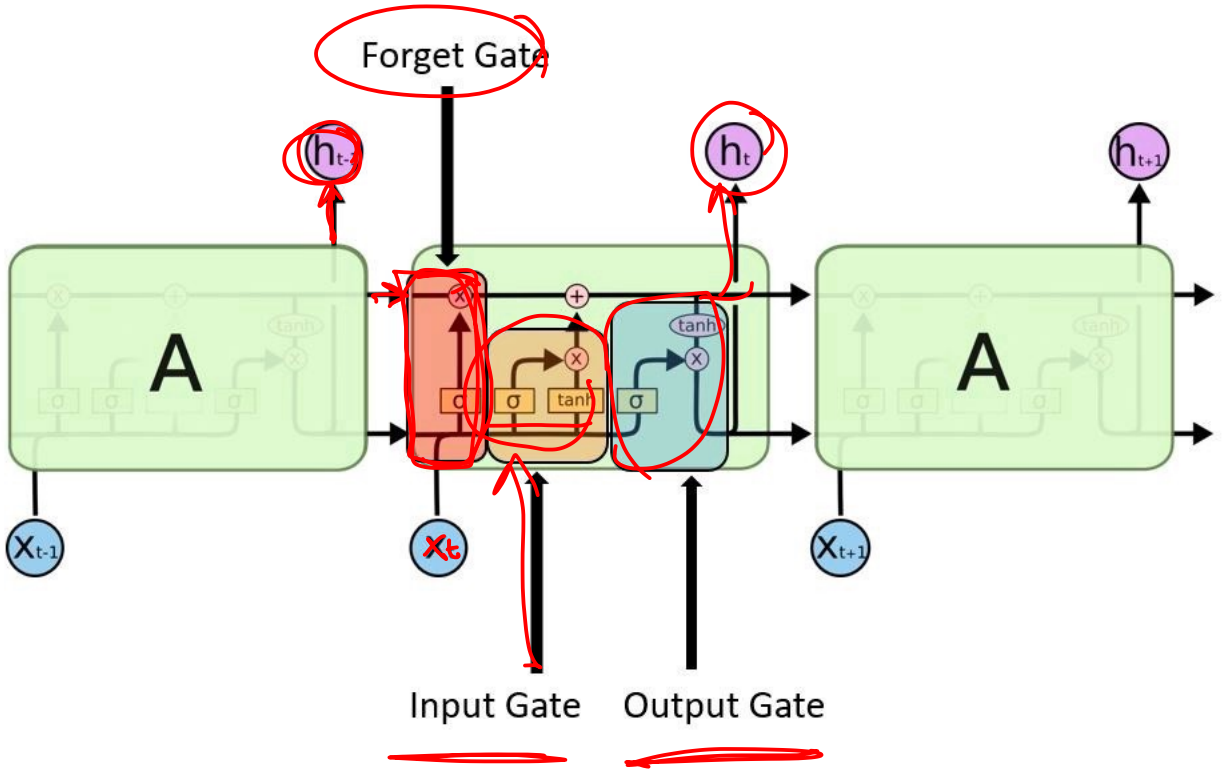
Problems with a standard RNN

- Simplest RNN model has a major drawback, called **vanishing gradient problem,** which prevents it from being accurate
- In a nutshell, the problem comes from the fact that at each time step during training we are using the same weights to calculate *y_t*. That multiplication is also done during back-propagation. The further we move backwards, the bigger or smaller our error signal becomes. This means that **the network experiences difficulty in memorising words from far away in the sequence** and makes predictions based on only the most recent ones.
- That is why more powerful models like LSTM and GRU come in hand. Solving the above issue, they have become the accepted way of implementing recurrent neural networks.

LSTM

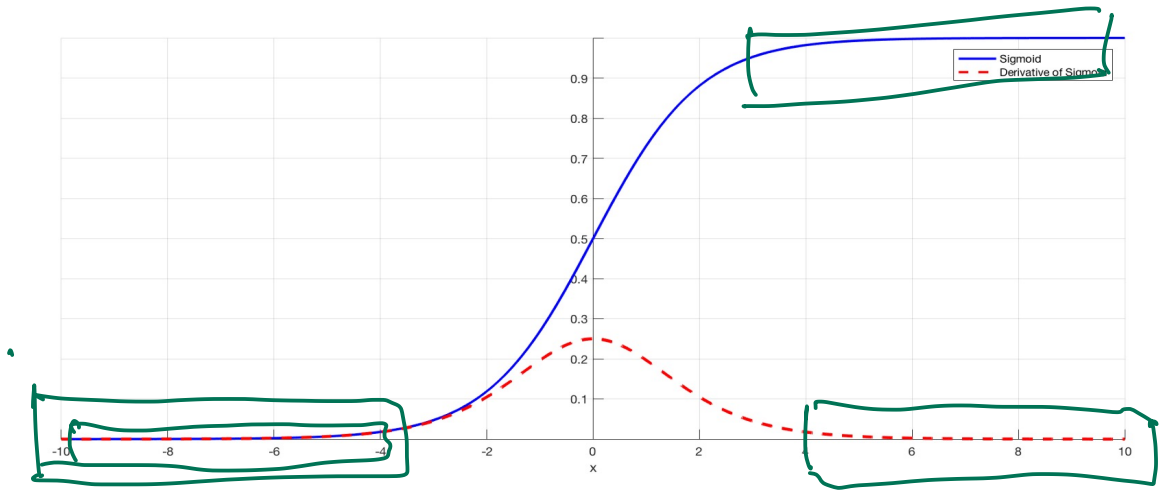
- Long Short-Term Memory (LSTM) networks are a modified version of recurrent neural networks, which makes it easier to remember past data in memory
- The vanishing gradient problem of RNN is resolved here
- LSTM is well-suited to classify, process and predict time series given time lags of unknown duration
- It trains the model by using back-propagation
- In an LSTM network, three gates are present
  - **Input gate** — discover which value from input should be used to modify the memory. **Sigmoid** function decides which values to let through 0,1. and tanh function gives weightage to the values which are passed deciding their level of importance ranging from-1 to 1
  - **Forget gate** — discover what details to be discarded from the block. It is decided by the sigmoid function. it looks at the previous state(ht-1) and the content input(Xt) and outputs a number between 0(*omit this*)and 1(*keep this*)for each number in the cell state Ct–1
  - **Output gate** — the input and the memory of the block is used to decide the output. **Sigmoid** function decides which values to let through 0,1. and tanh function gives weightage to the values which are passed deciding their level of importance ranging from-1 to 1 and multiplied with output of **Sigmoid.**

LSTM



Vanishing gradient problem

- **Problem**
  - As more layers using certain activation functions are added to neural networks, the gradients of the loss function approaches zero, making the network hard to train
- **Reason**
  - Certain activation functions, like the sigmoid function, squishes a large input space into a small input space between 0 and 1
  - Therefore, a large change in the input of the sigmoid function will cause a small change in the output. Hence, the derivative becomes small
  - As an example, Image is the sigmoid function and its derivative. Note how when the inputs of the sigmoid function becomes larger or smaller (when lxl becomes bigger), the derivative becomes close to zero.



Why it's significant ?

- For shallow network with only a few layers that use these activations, this isn't a big problem
- However, when more layers are used, it can cause the gradient to be too small for training to work effectively
- Gradients of neural networks are found using backpropagation
- Simply put, backpropagation finds the derivatives of the network by moving layer by layer from the final layer to the initial one
- By the chain rule, the derivatives of each layer are multiplied down the network (from the final layer to the initial) to compute the derivatives of the initial layers
- However, when *n* hidden layers use an activation like the sigmoid function, *n* small derivatives are multiplied together. Thus, the gradient decreases exponentially as we propagate down to the initial layers.
- A small gradient means that the weights and biases of the initial layers will not be updated effectively with each training session. Since these initial layers are often crucial to recognizing the core elements of the input data, it can lead to overall inaccuracy of the whole network.

Gated RNN 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

How do GRUs work?

- As mentioned above, GRUs are improved version of standard recurrent neural network. But what makes them so special and effective?
- To solve the vanishing gradient problem of a standard RNN, GRU uses, so-called, **update gate and reset gate**. Basically, these are two vectors which 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 remove information which is irrelevant to the prediction.