Chapter 10: DeepNLP - Recurrent Neural Networks with Math.

Madhu Sanjeevi (Mady) · Follow
Published in Deep Math Machine learning.ai
6 min read · Jan 10, 2018

Listen Share More

we talked about normal neural networks quite a bit, Let's talk about fancy neural networks called **recurrent neural networks.**

Before we talk about what exactly RNN's are, let me first put this "Why RNN's ???" (I am a big fan of **Simon sinek**, so I start with why.)

A neural network usually takes an independent variable **X** (or a set of independent variables) and a dependent variable **y** then it learns the mapping between **X** and **y** (we call this **Training),** Once training is done , we give a new independent variable to predict the dependent variable.

infact that's all most of machine learning(supervised).

but what if the order of data matters????? just imagine what if the order of all independent variables matter???

Let me explain visually (I call this The RecurAnt Theory).

Just assume every ant is an independent variable if one ant goes in a different direction , it does not matter for other ants right?

But what if the order of the ants matters ?

if one ant misses or turns away from the group, it affects the following ants.

So a normal neural network does not follow the order, so when we tackle the real world problems where the order matters , we need Recurrent neural networks. Period.

So which data where the order matter in our ML space ????

1. Natural Language Data where the order of words matter
2. Speech data
3. Time series data
4. Video/Music Sequences data
5. Stock markets data

etc....

so how RNN's solve "the whole order matters thing" data??????

> *Note: I take natural text data as an example to explain RNN's.*

Let's say i am doing sentiment analysis on user reviews on a movie

"This movie is good" → Positive *"This movie is bad"* → negative

We can classify these by using simple model "Bag of words" and we can predict (Positive or Negative) but wait…

what if the review is *"This movie is not good"*

The BOW model may say it's a positive sign but actually it's not.

The RNN understands it and predicts that it's negative.

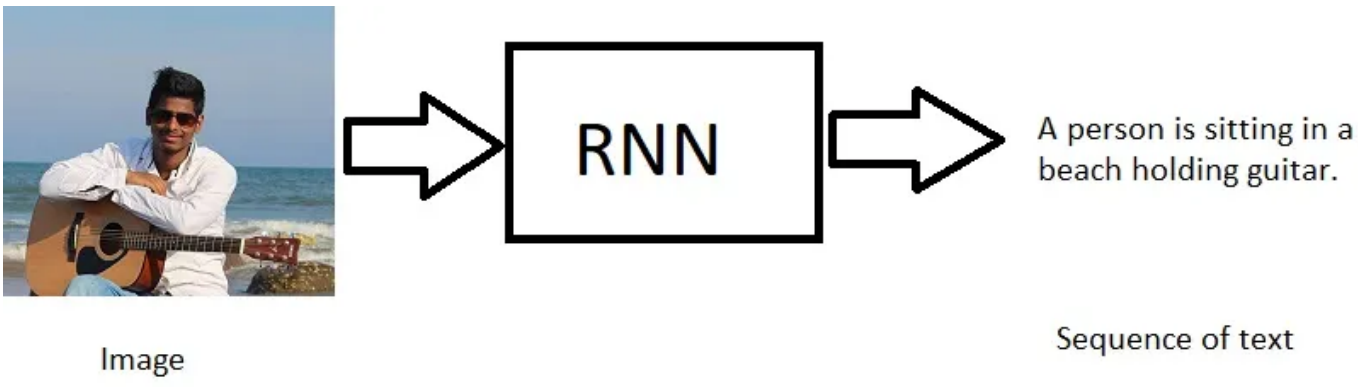
How??????

First let's admit that here the order of the text matters. cool? okay

RNN has the following models

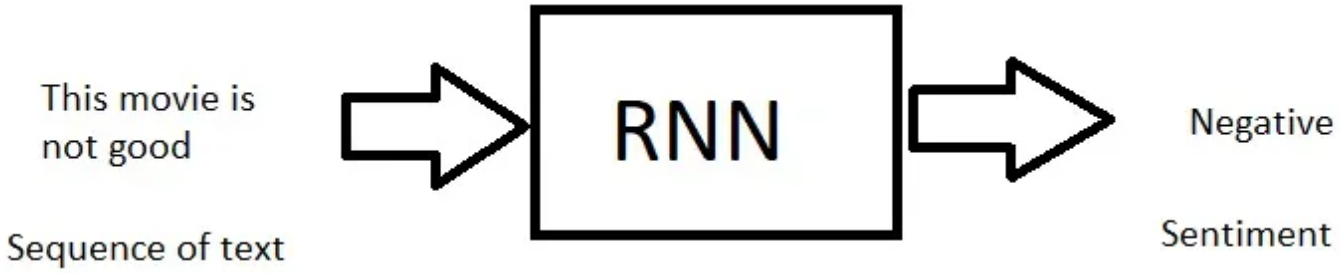
1. One to Many

RNN takes one input lets say an image and generates a sequence of words.



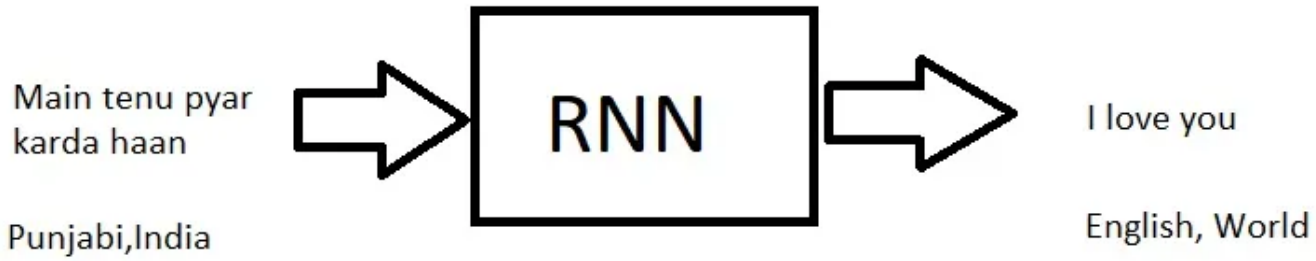
2.Many to One

RNN takes sequence of words as input and generates one output.



3.Many to Many

RNN takes sequence of words as input and generates sequence of words as output. (lets say language translations).



Currently we are focusing on 2nd model “Many to One”.

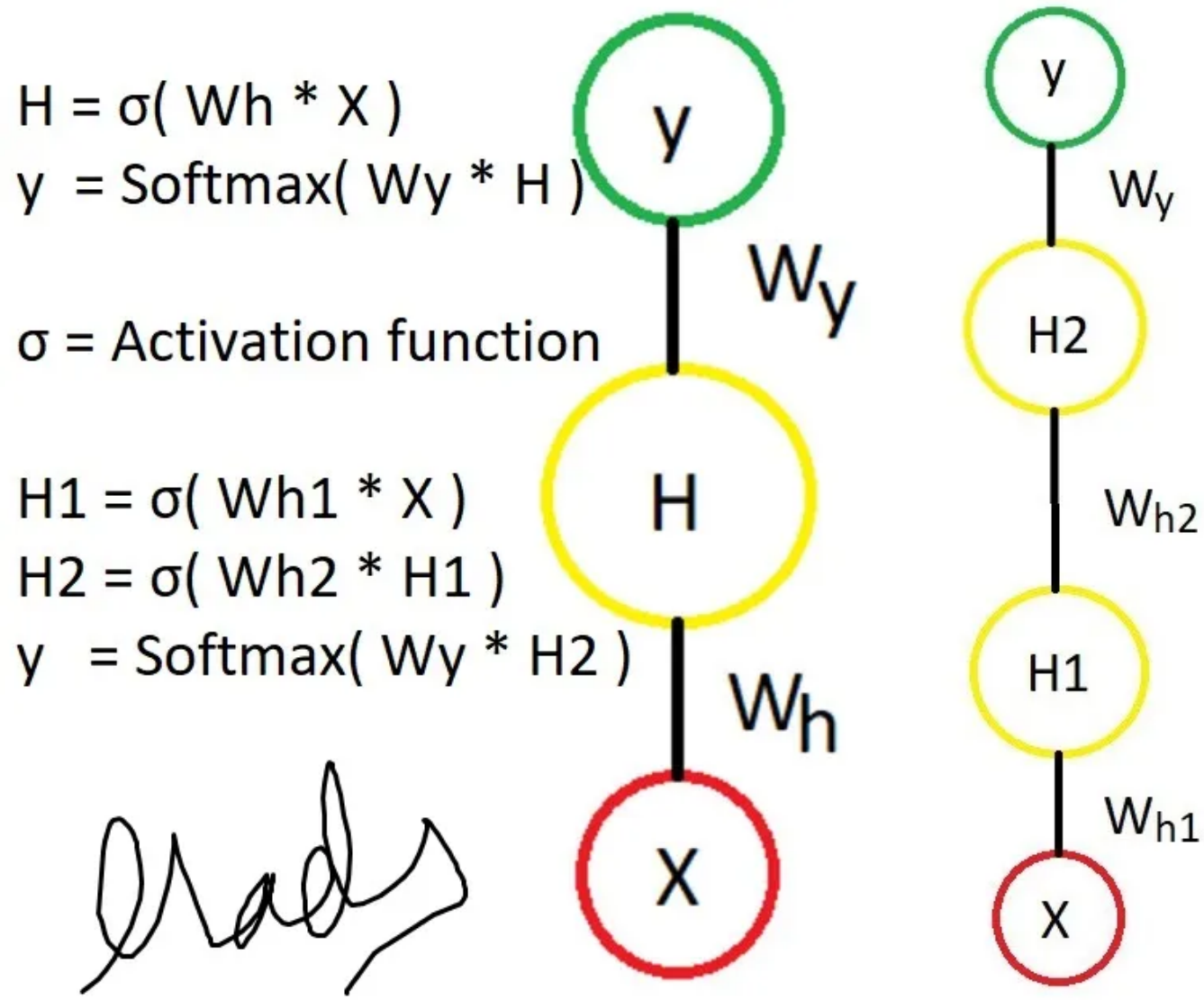
in RNN’s Input is considered as time steps.

ex : input(X) = [“this”, “movie”, “is”, “not”, “good”]

Time stamp for “this” is x(0), “movie” is x(1), “is” is x(2) ,“not” is x(3) and “good” is x(4).

First let’s understand what RNN cell contains!

I hope and assume you know Feed Forward NN or you can read my earlier story here NN. Summary of FFNN is



Feed Forward NN.

In Feed forward neural network we have X(input) and H(Hidden) and y(output)

you can have as many hidden layers as you want but weights (W)for every hidden layers are different.

Above *Wh1* and *Wh2* are different.

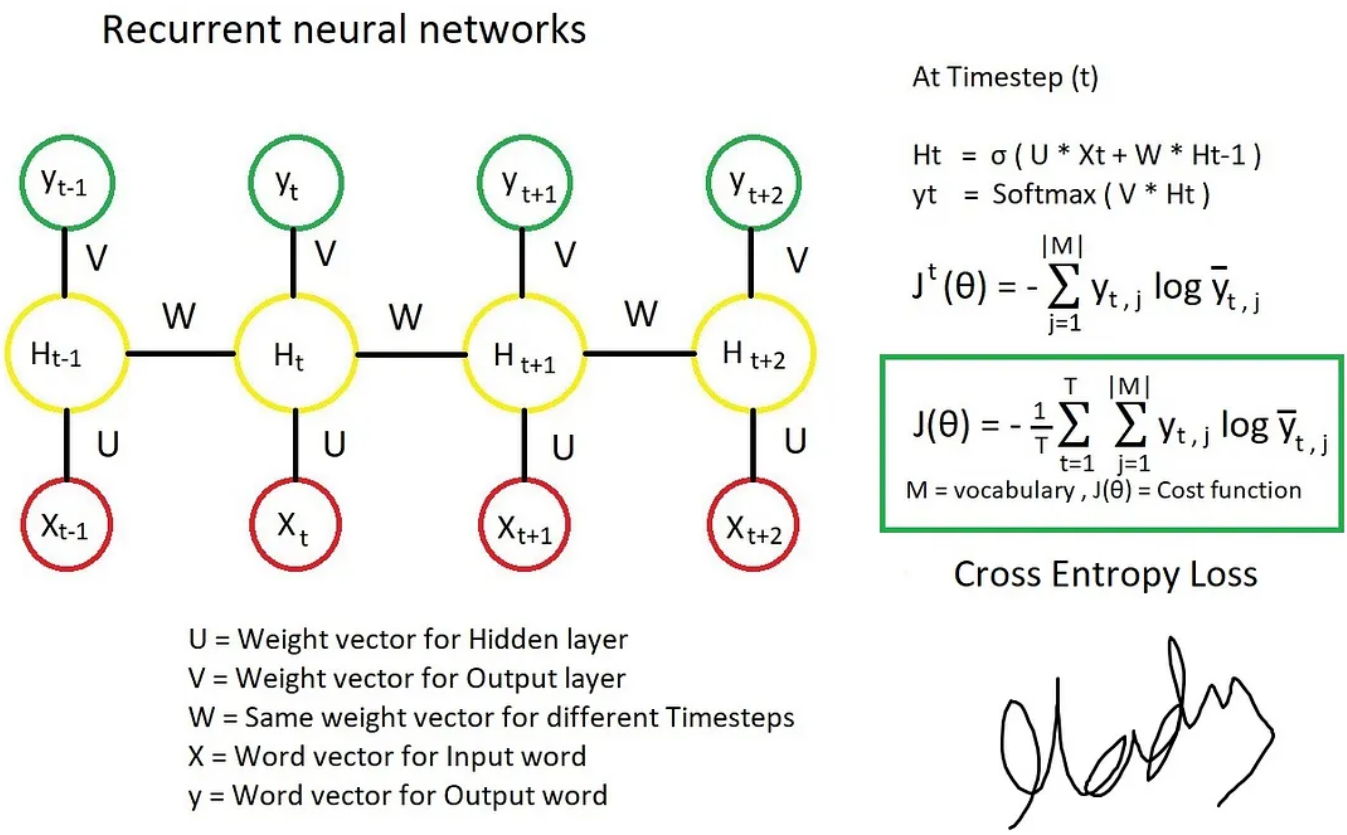
The RNN cell contains a set of feed forward neural networks cause we have time steps.

The RNN has: sequential input, sequential output, multiple timesteps, and multiple hidden layers.

Unlike FFNN , here we calculate hidden layer values not only from input values but also previous time step values and Weights (W) at hidden layers are same for time

steps.

Here is the complete picture for RNN and it's Math.



In the picture we are calculating the Hidden layer time step (t) values so

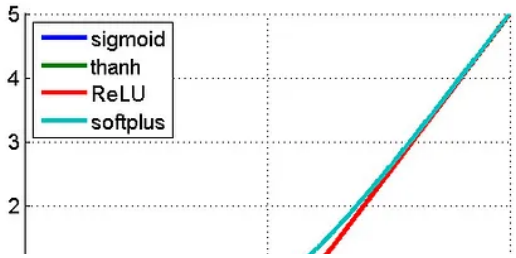
Ht = Activatefunction(input * Hweights + W * Ht-1)

yt = softmax(Hweight* Ht)

Ht-1 is the previous time step and as i said W's are same for all timesteps.

The activation function can be Tanh, Relu, Sigmoid, etc..

	Propagation
Sigmoid	$y_s = \frac{1}{1+e^{-x_s}}$



Above we calculated only for Ht similarly we can calculate for all other timesteps.

Steps:

1. Calculate **Ht-1** from **U** and **X**
2. Calculate **yt-1** from **Ht-1** and **V**
3. Calculate **Ht** from **U,X,W** and **Ht-1**
4. Calculate **yt** from **V** and **Ht** and so on…

Note :

1.**U** and **V** are weight vectors, different for every time step.

2.We can even calculate hidden layer(all time steps) first then calculate y values.

3. Weight vectors are random initially.

Once Feed forwarding is done then we need to calculate the error and backpropagate the error using back propagation.

we use Cross entropy as cost function (assume you know so not going into details)

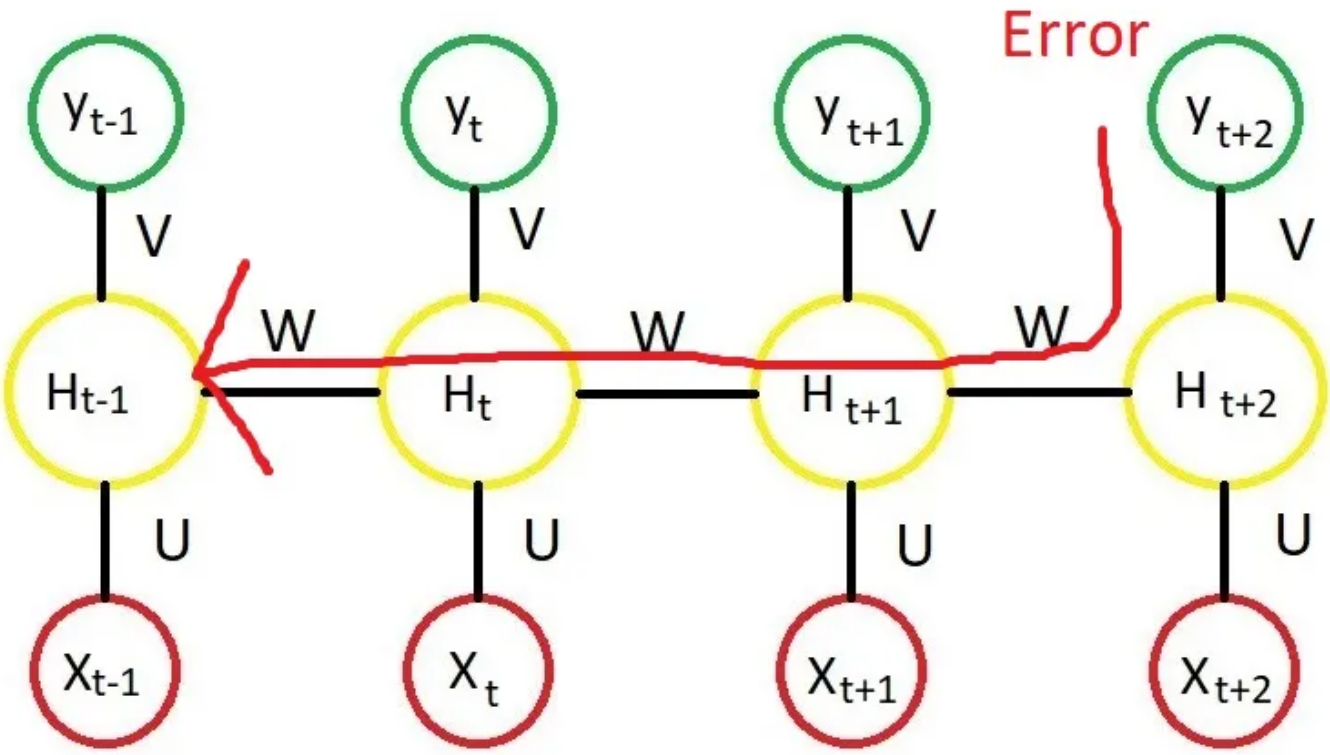
> *BPTT (Back propagation through time)*

if you know how Normal neural network works , the rest is pretty easy , if you don't know, here is my article that talks about **Artificial Neural Networks.**

We need to calculate the below terms

1. how much does the ***total error*** change with respect to the ***output (hidden and output units)*** ? (or how much is a change in ***output***)
2. how much does the ***output*** change with respect to ***weights (U,V,W)***? (or how much is a change in ***weights***)

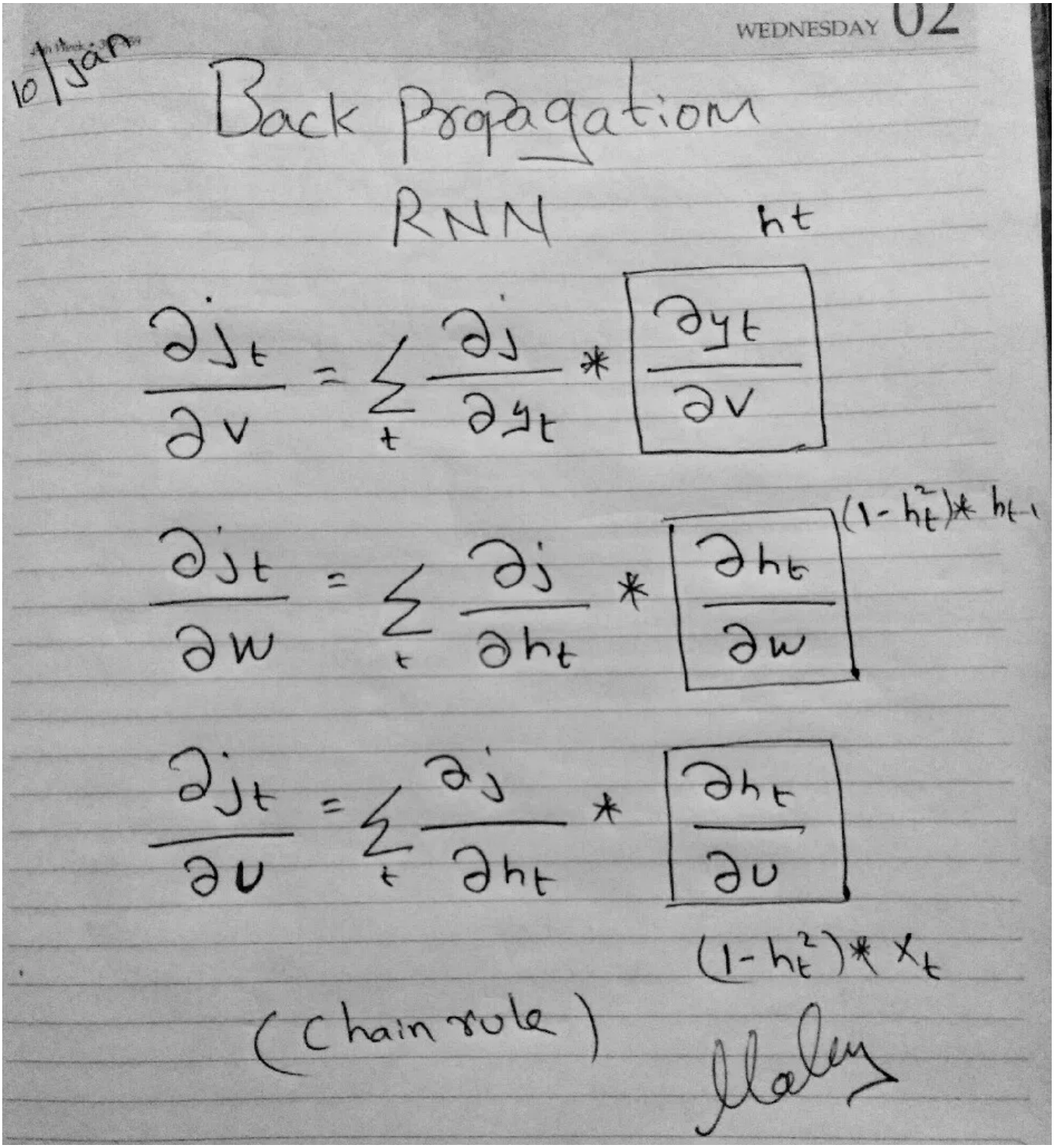
Since W's are same for all time steps we need to go all the way back to make an update.



Remember the BP for RNN is as same as neural networks BP

but here Current time step is calculated based on the previous time step so we have to traverse all the way back.

if we apply chain rule which looks like this



W's are same for all the time steps so the chain rule expands more and more

A similar but a different way of working out the equations can be seen in Richard Sochers's Recurrent Neural Network lecture slide.

- Similar but simpler RNN formulation:

$$h_t = W f(h_{t-1}) + W^{(hx)} x_{[t]}$$
$$\hat{y}_t = W^{(S)} f(h_t)$$

- Total error is the sum of each error at time steps t

$$\frac{\partial E}{\partial W} = \sum_{t=1}^{T} \frac{\partial E_t}{\partial W}$$

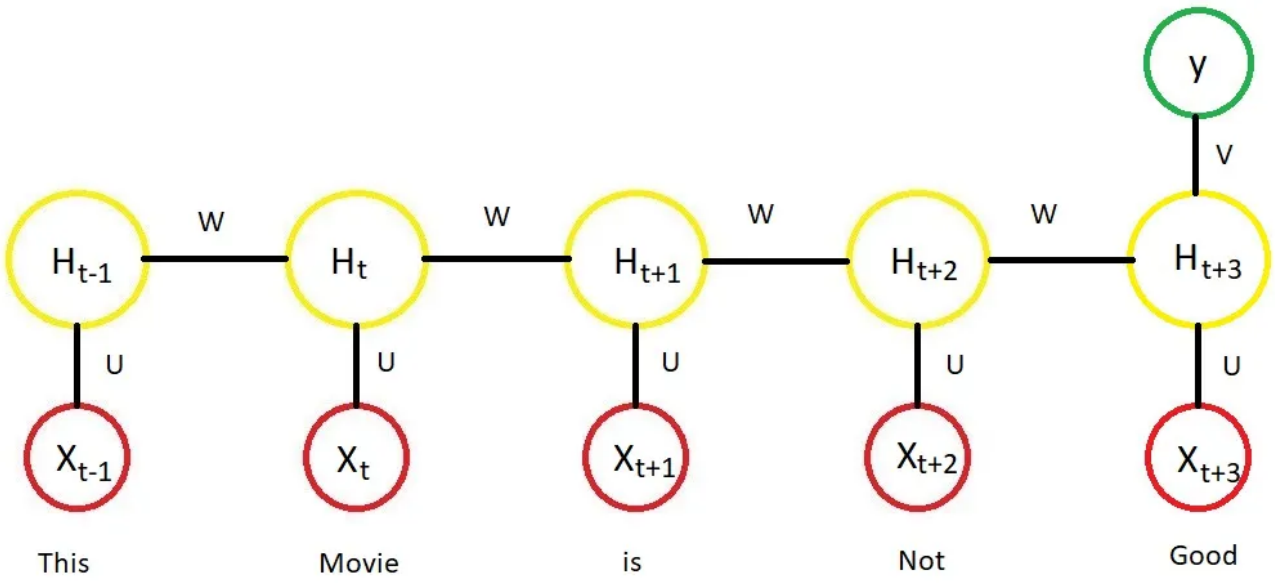
- Hardcore chain rule application:

$$\frac{\partial E_t}{\partial W} = \sum_{k=1}^{t} \frac{\partial E_t}{\partial y_t} \frac{\partial y_t}{\partial h_t} \frac{\partial h_t}{\partial h_k} \frac{\partial h_k}{\partial W}$$

So here **Et** is same as our **J(θ)**

U, V and W should get updated using any optimization algorithms like gradient descent (Take a look at my story here **GD**).

Now if we go back and talk about our sentiment problem here is the RNN for that



We give word vectors or one hot encoding vectors for every word as input and we do feed forward and BPTT ,Once the training is done, we can give new text for prediction.

It learns something like whereever “not” + positive word = negative.

I hope you get that.

Problems with RNN → Vanishing/exploding gradient problem

Since W’s are same for all timesteps, during back propagation as we go back adjusting the weights, The signal gets either too weak or too strong which cause either vanishing

or exploding problem.

To avoid this we use either GRU or LSTM which I will cover in the next Stories

So That's it for this story , In the next story I will build the Recurrent neural network from scratch and using Tensorflow using the above steps and same Math.

Suggestions /questions are welcome.

Photos are designed using Paint in windows.

See ya!

Machine Learning Deep Learning Recurrent Neural Network NLP

Artificial Intelligence

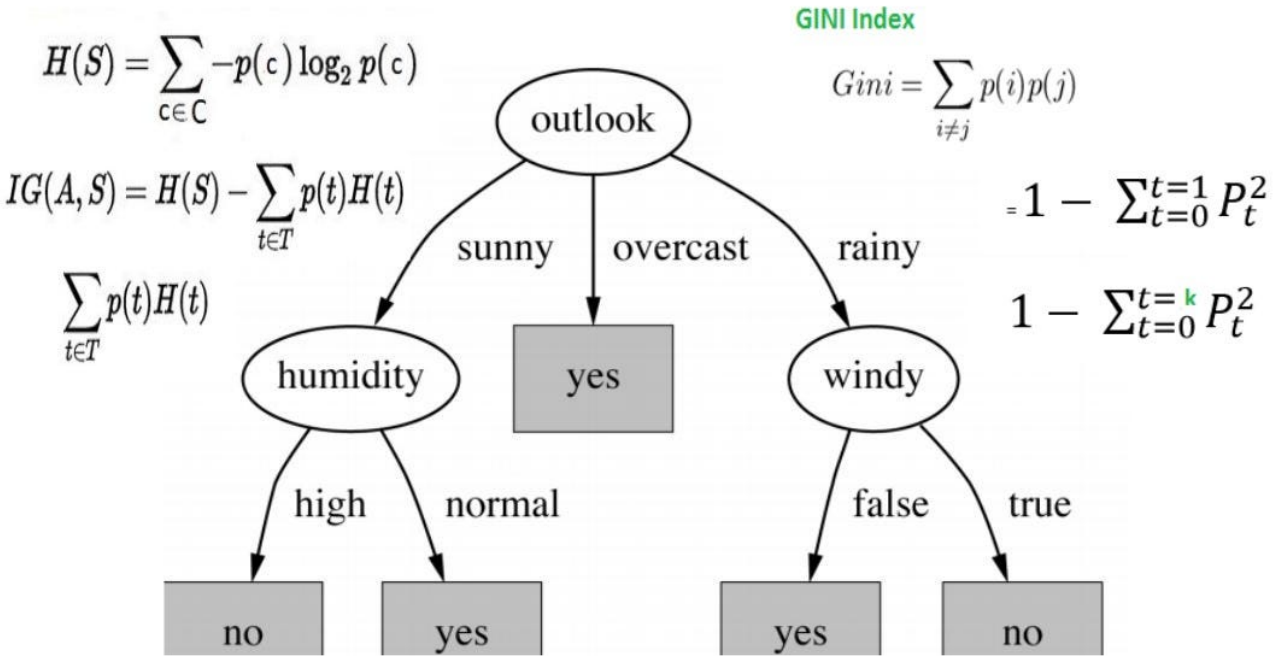
Follow

Written by Madhu Sanjeevi (Mady)

4.2K Followers · Editor for Deep Math Machine learning.ai

Writes about Technology (AI, Blockchain) | interested in Programming || Science || Math
https://www.linkedin.com/in/madhusanjeeviai

More from Madhu Sanjeevi (Mady) and Deep Math Machine learning.ai



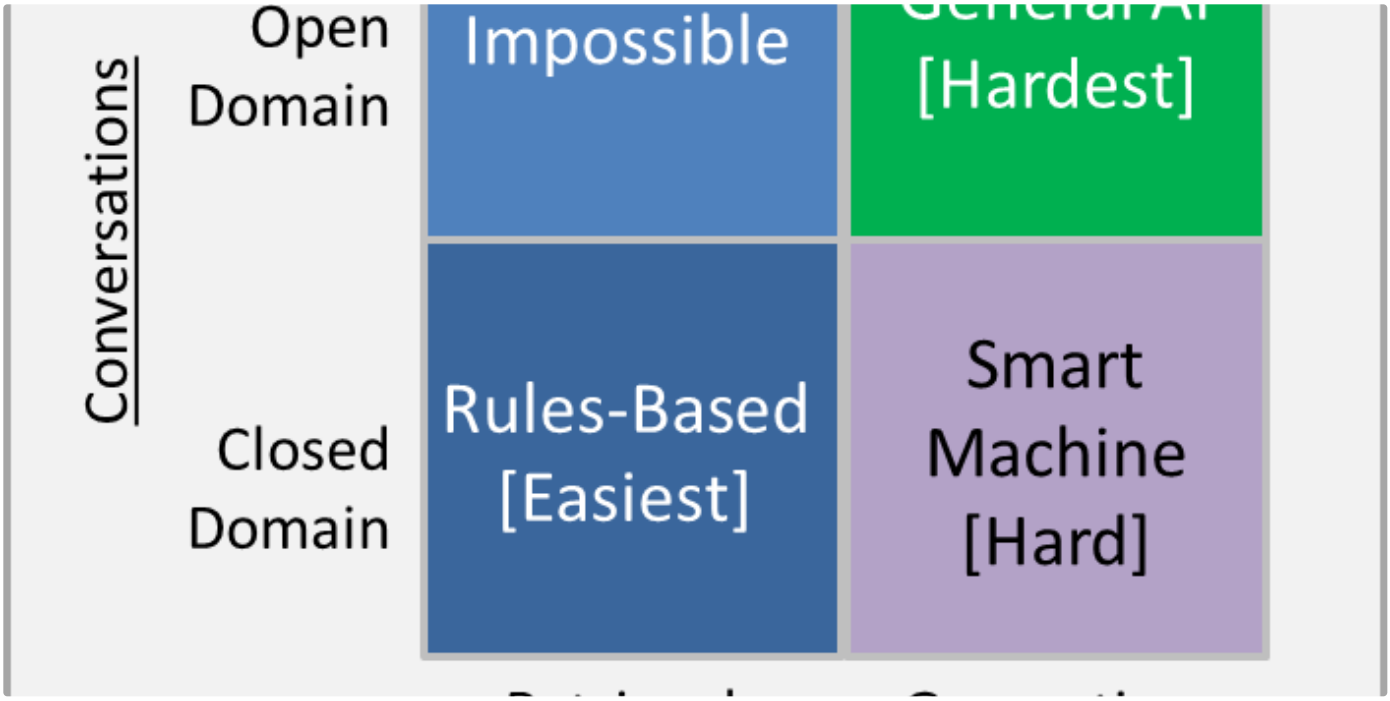
Madhu Sanjeevi (Mady) in Deep Math Machine learning.ai

Chapter 4: Decision Trees Algorithms

Decision tree is one of the most popular machine learning algorithms used all along, This story I wanna talk about it so let's get...

6 min read · Oct 6, 2017

5.3K 26



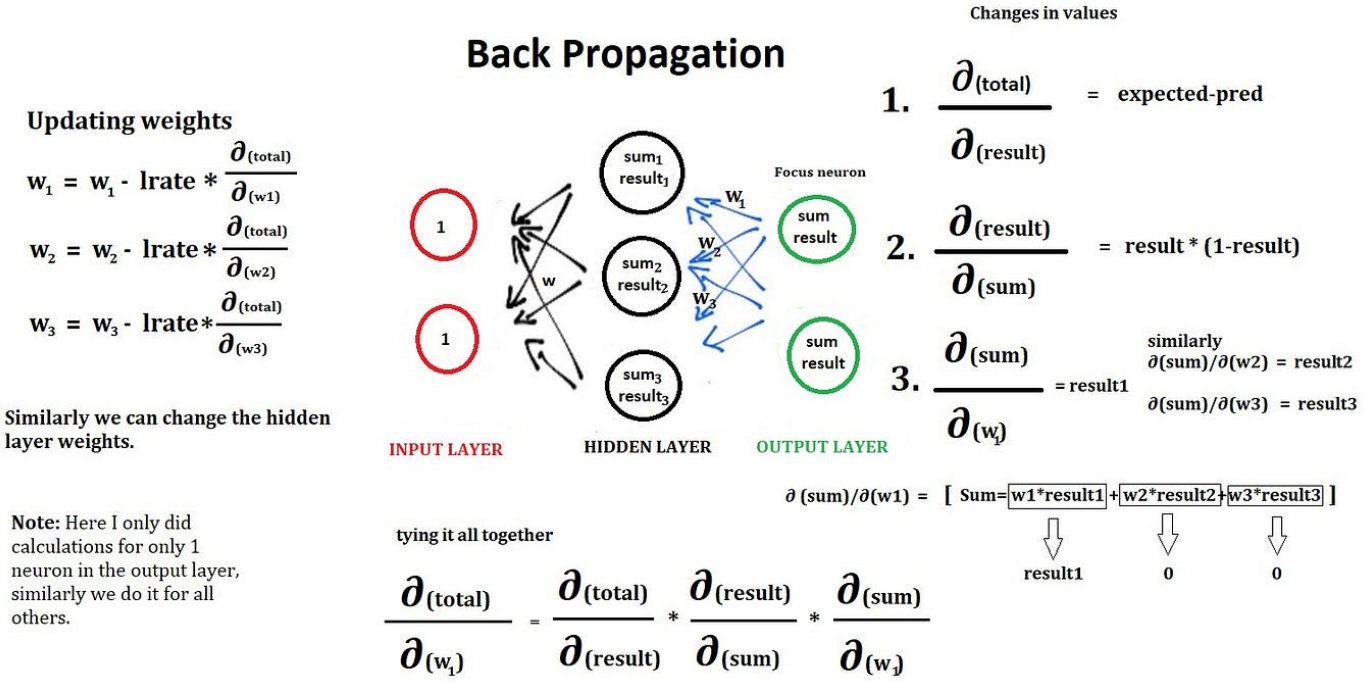
Madhu Sanjeevi (Mady) in Deep Math Machine learning.ai

Chapter 11: ChatBots to Question & Answer systems.

I am really excited to write this story , so far I have talked about Machine learning,deep learning,Math and programming and I am sick of...

13 min read · Apr 19, 2018

1.7K 18



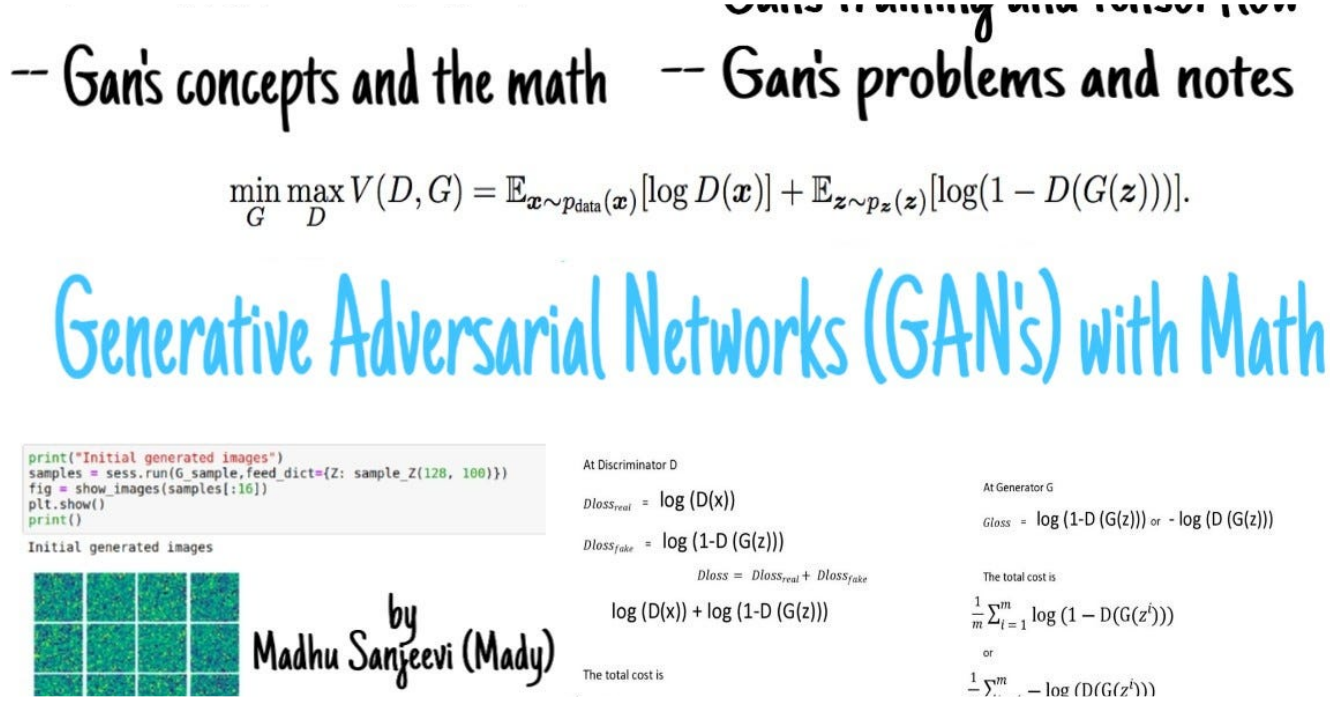
Madhu Sanjeevi (Mady) in Deep Math Machine learning.ai

Chapter 7 : Artificial neural networks with Math.

I have been talking about the machine learning for a while, I wanna talk about Deep learning as I got bored of ML.

6 min read · Oct 11, 2017

629 8



Madhu Sanjeevi (Mady) in Deep Math Machine learning.ai

Ch:14 General Adversarial Networks (GAN's) with Math.

Discriminative vs generative , Gan's training and tensorflow, gan's concepts and the math and gans problems.

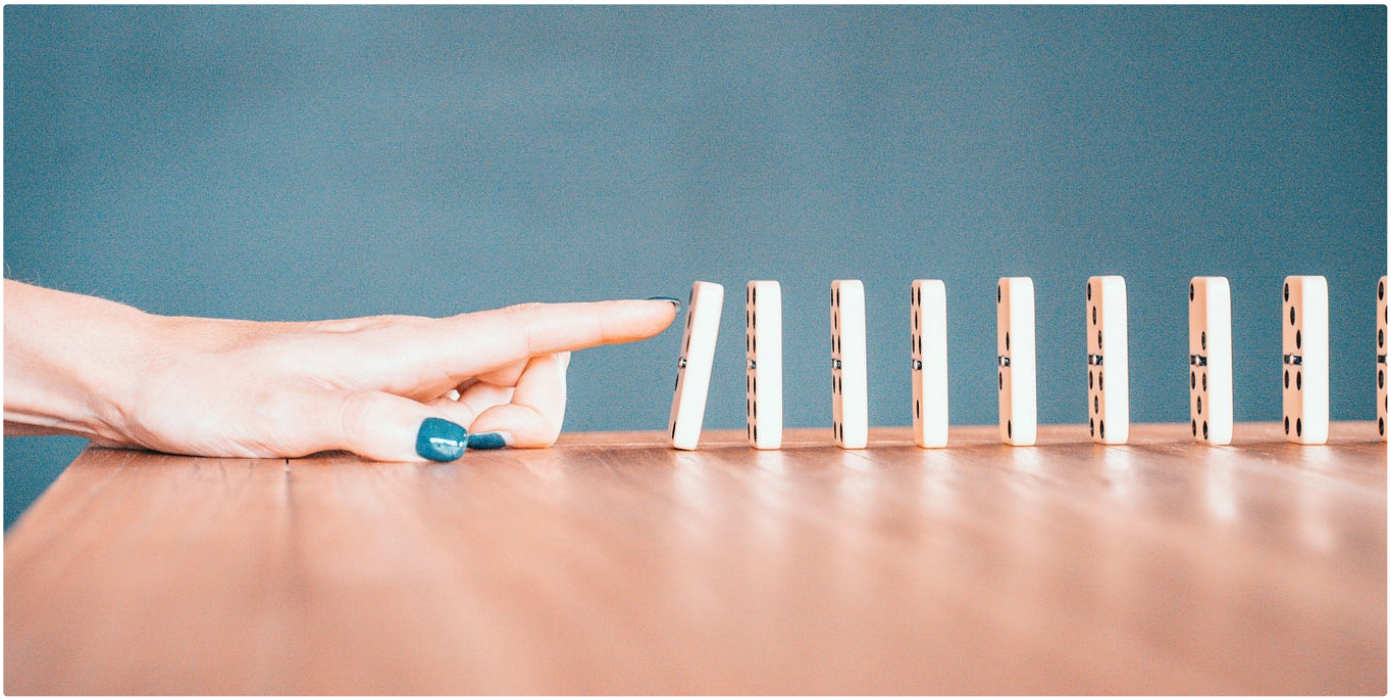
11 min read · Jan 14, 2019

Prateek Gaurav

NLP : Zero To Hero [Part 1: Introduction, BOW, TF-IDF & Word2Vec]

Natural Language Processing (NLP) has become an integral part of various industries, including healthcare, finance, and e-commerce, to...

· 10 min read · Mar 23



Youssef Hosni in Towards AI

Building An LSTM Model From Scratch In Python

How to build a basic LSTM using Basic Python libraries

· 17 min read · Jan 2

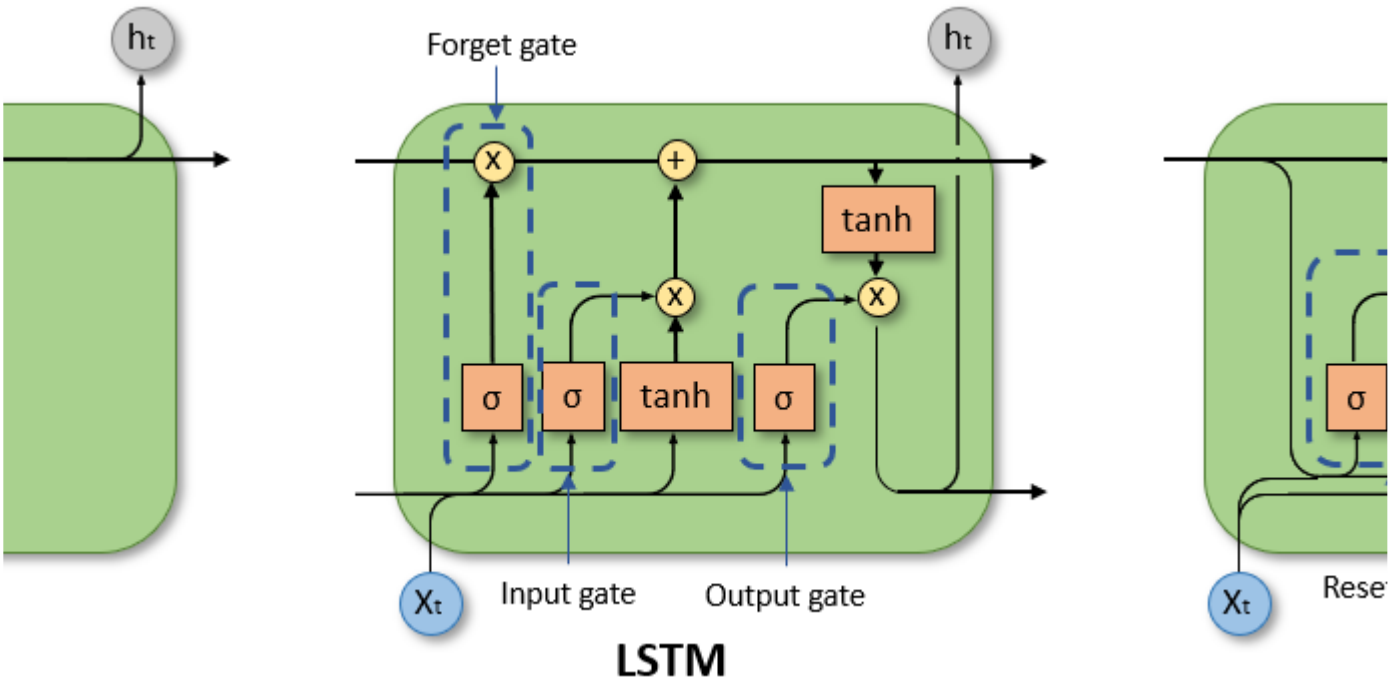
Recurrent Neural Networks (RNN)

Raj Pandey

Recurrent Neural Network RNN

Introduction

5 min read · 5 days ago



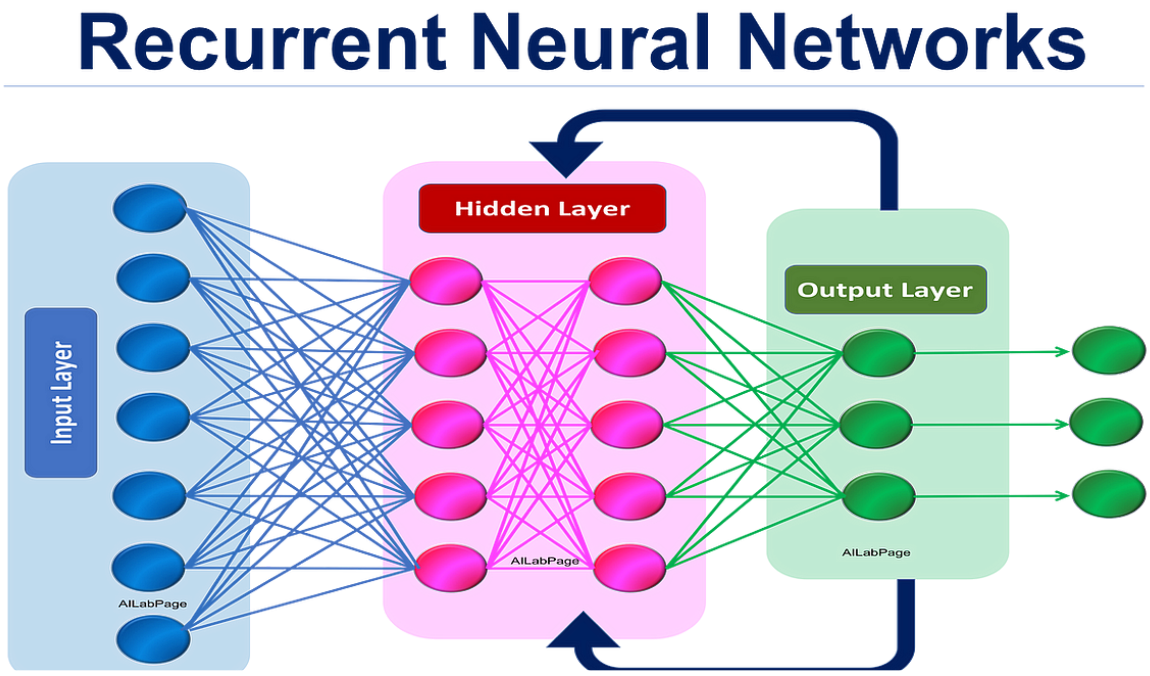
Jonte Dancker in Towards Data Science

A Brief Introduction to Recurrent Neural Networks

An introduction to RNN, LSTM, and GRU and their implementation

12 min read · Dec 26, 2022

303 4



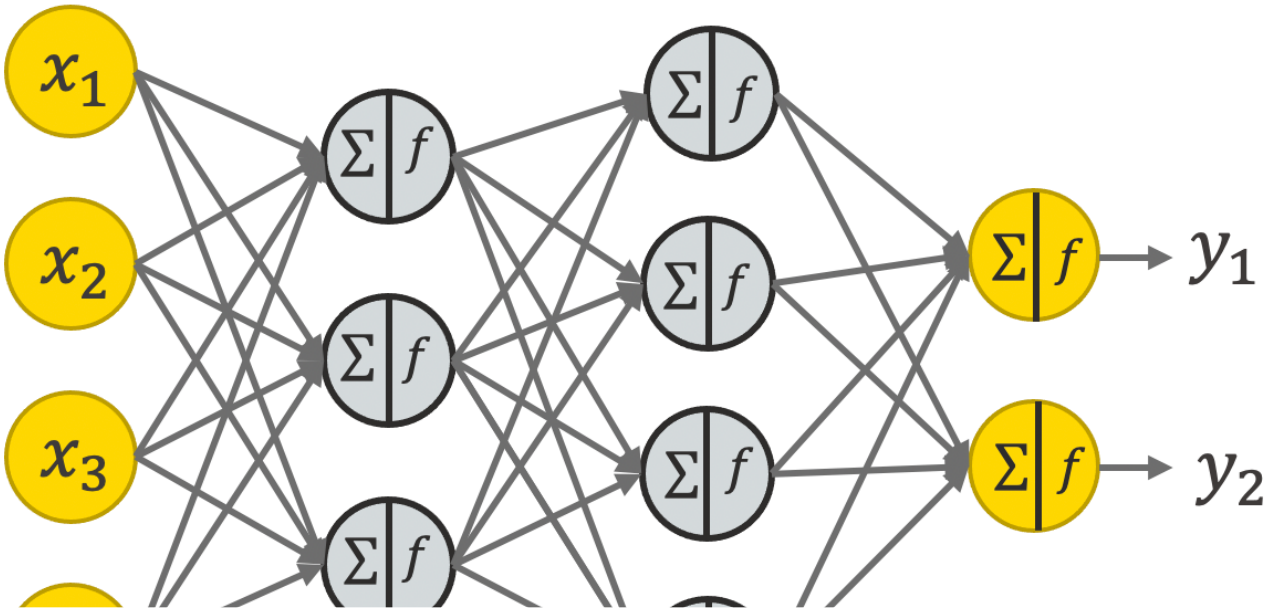
Farheenshaukat

Recurrent Neural Network (RNN)

A recurrent neural network (RNN) is a type of neural network that is commonly used for processing sequential data such as speech, natural...

8 min read · May 11

54



Saankhya Mondal

Neural Network and Back-propagation from Scratch

Back-propagation is the basic algorithm used in every neural network-based architectures. In an age when several Python libraries and...

5 min read · Dec 7, 2022

53

See more recommendations