Deep Learning

Recurrent Neural Networks

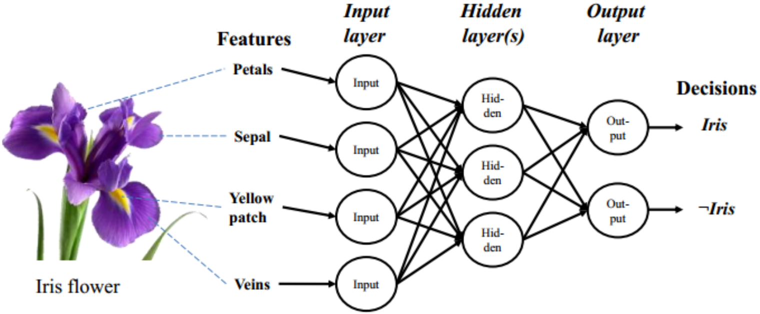
Vinayakumar R
https://sites.google.com/site/vinayakumarr77/

January 24, 2018

Outline

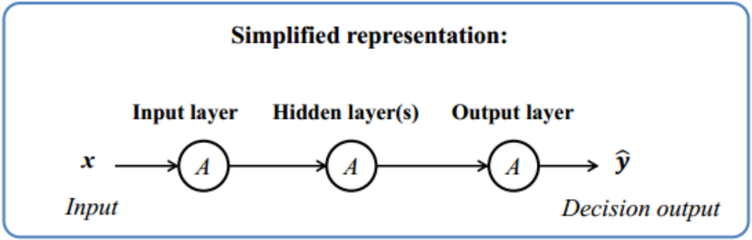
Recap: feed-forward artificial neural network

- W. McCulloch and W. Pitts , 1940s Abstract mathematical model of a brain cell
- F. Rosenblatt, 1958 Perceptron for classification
- P. Werbos, 1975 Multi-layer artificial neural network



Recap: feed-forward artificial neural network

- Decisions are based on current inputs: No memory about the past , No future scope



- Vector of input features x
- Vector of predicted values $ypred$
- Neural activation $y_{pred} = A(\sum_{i=1}^{n} x_i w_{iy} + b_y)$, where A is some activation function and w and b are weights and bias respectively

Temporal Dependencies

Analyzing temporal dependencies → Improved decisions

Frame 0	Frame 1	Frame 2	Frame 3	Frame 4
Stem: seen Petals: hidden	Stem: seen Petals: hidden	Stem: seen Petals: partial	Stem: partial Petals: partial	Stem: hidden Petals: seen

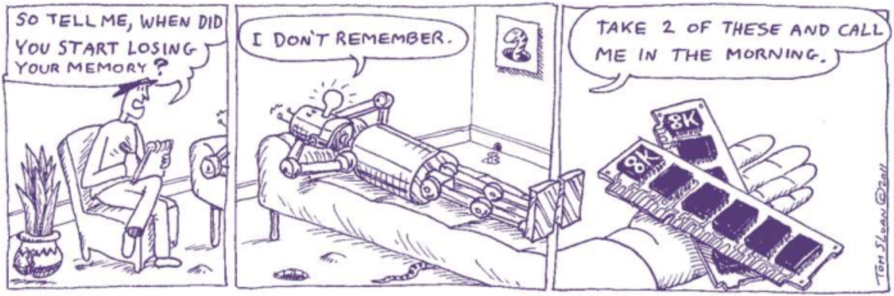
Frame 0	Frame 1	Frame 2	Frame 3	Frame 4
$P(Iris)$: 0.1 $P(\neg Iris)$: 0.9	$P(Iris)$: 0.11 $P(\neg Iris)$: 0.89	$P(Iris)$: 0.2 $P(\neg Iris)$: 0.8	$P(Iris)$: 0.45 $P(\neg Iris)$: 0.55	**$P(Iris)$: 0.9** $P(\neg Iris)$: 0.1

Decision on sequence of observations

Backpropogation

- The goal of the back propagation training algorithm is to modify the weights of a neural network in order to minimize the error of the network outputs compared to some expected output in response to corresponding inputs.
  The general algorithm is as follows:
- Present a training input pattern and propagate it through the network to get an output.
- Compare the predicted outputs to the expected outputs and calculate the error.
- Calculate the derivatives of the error with respect to the network weights.
- Adjust the weights to minimize the error.
- Repeat.

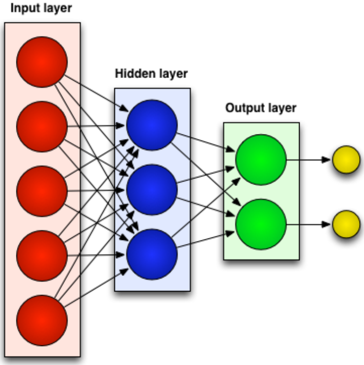
Temporal Dependencies



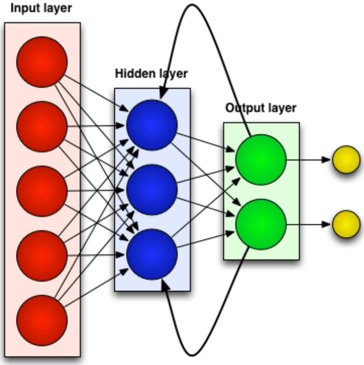
Memory is important → Reasoning relies on experience

RECURRENT NEURAL NETWORK ARCHITECTURES

Feed forward Neural Networks:
Connections between the units do not form a cycle

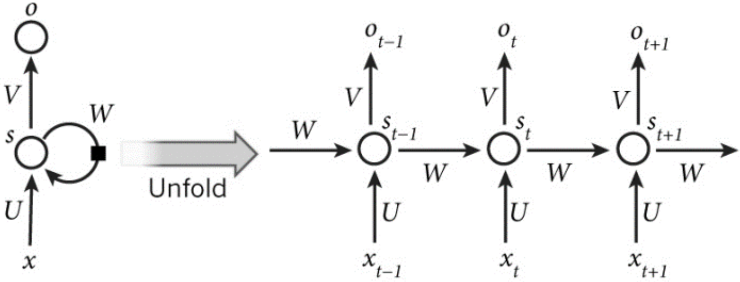


Recurrent Neural Network:
Connections between units form cyclic paths



An unrolled RNN (in time) can be considered as a deep neural network (DNN)

$$S_t = f\left(Ux_t + Ws_{t-1}\right)$$
$$y = g\left(Vs_t\right)$$



- x_t: input at time t
- s_t: hidden state at time t (memory of the network)
- f: an activation function (e.g: $tanh()$ and ReLUs
- U, V, W: network parameters (unlike a feedforward neural network, an RNN shares the same parameters across all time steps)
- g: activation function for the output layer (typically a softmax function)
- y: output of the network at time t

Backpropogation through time

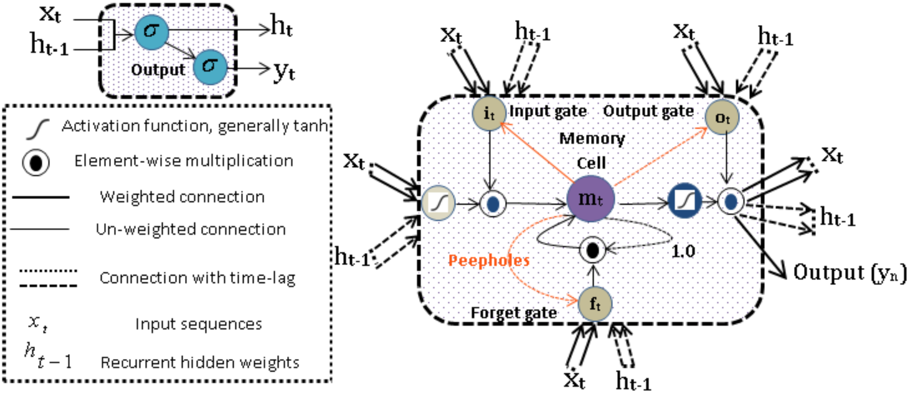
Backpropagation Through Time, or BPTT, is the application of the Backpropagation training algorithm to recurrent neural network applied to sequence data like a time series.

- Present a sequence of time steps of input and output pairs to the network.
- Unroll the network then calculate and accumulate errors across each time step.
- Roll-up the network and update weights.
- Repeat

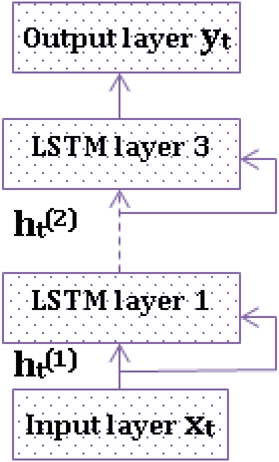
Long short term memory

- When Backpropagation is used in very deep neural networks and in unrolled recurrent neural networks, the gradients that are calculated in order to update the weights can become unstable.
- They can become very large numbers called exploding gradients or very small numbers called the vanishing gradient problem.
- This problem is alleviated in deep multilayer Perceptron networks through the use of the Rectifier transfer function
- In recurrent neural network architectures, this problem has been alleviated using a new type of architecture called the Long Short-Term Memory Networks

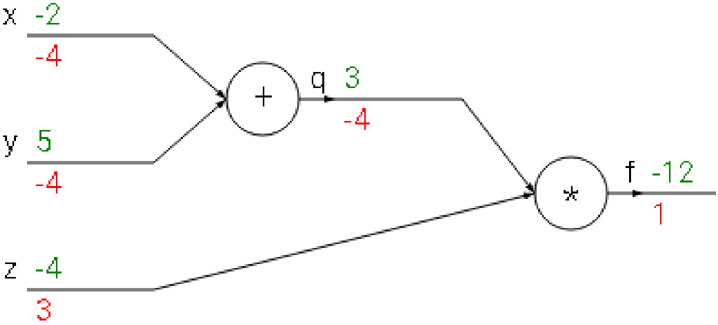
Architecture of RNN unit and LSTM memory block



Stacked LSTM



Backpropogation example



- # set some inputs
- $x = -2; y = 5; z = -4$
- # perform the forward pass
- $q = x + y$ # q becomes 3
- $f = q * z$ # f becomes -12
- # perform the backward pass (backpropagation) in reverse order:
- # first backprop through $f = q * z$
- $dfdz = q$ # $df/dz = q$, so gradient on z becomes 3
- $dfdq = z$ # $df/dq = z$, so gradient on q becomes -4
- # now backprop through $q = x + y$
- $dfdx = 1.0 * dfdq$ # $dq/dx = 1$. And the multiplication here is the chain rule!
- $dfdy = 1.0 * dfdq$ # $dq/dy = 1$

Thank You!!!