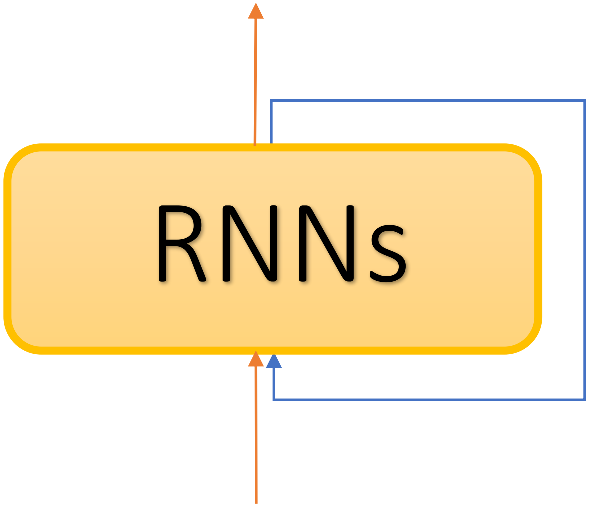


~~Under the hood~~
On the Surface

Elvis Saravia

Icebreaker

Can we predict the future based on our current decisions?

"Which research direction should I take?"

Elvis Saravia

Outline

- Part 1: Review Neural Network Essentials
- Part 2: Sequential Modeling
- Part 3: Introduction to Recurrent Neural Networks
- Part 4: RNNs with Tensorflow

Elvis Saravia

Part 1

The Neural Network

Perceptron Forward Pass

Computing output:

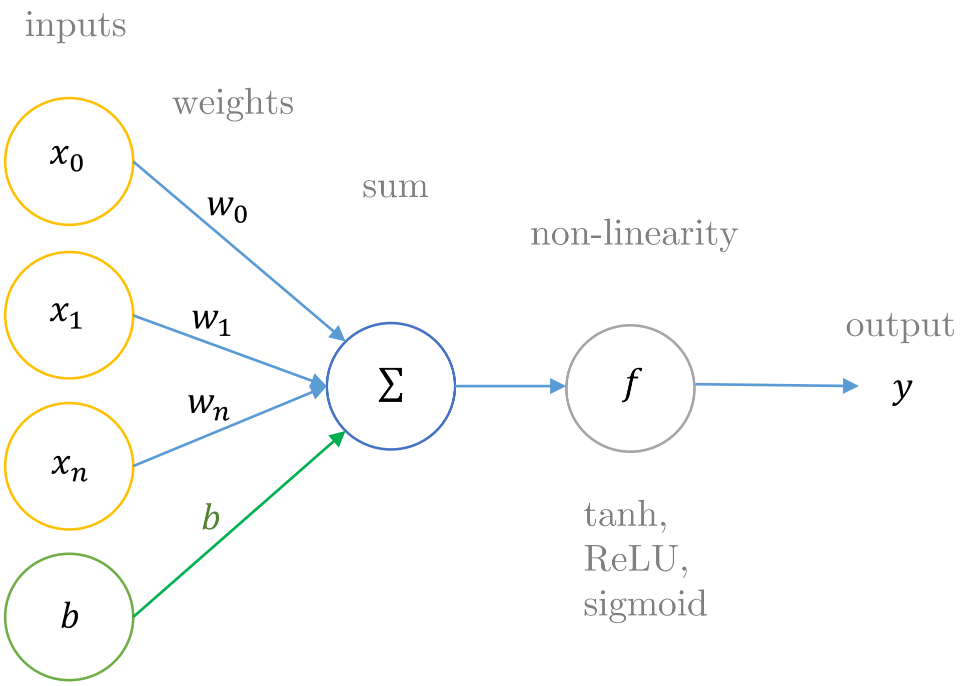
$$y = f\left(\left(\sum_{i=0}^{N} x_i * w_i\right) + b\right)$$

Vector form:

$$y = f(XW + b)$$

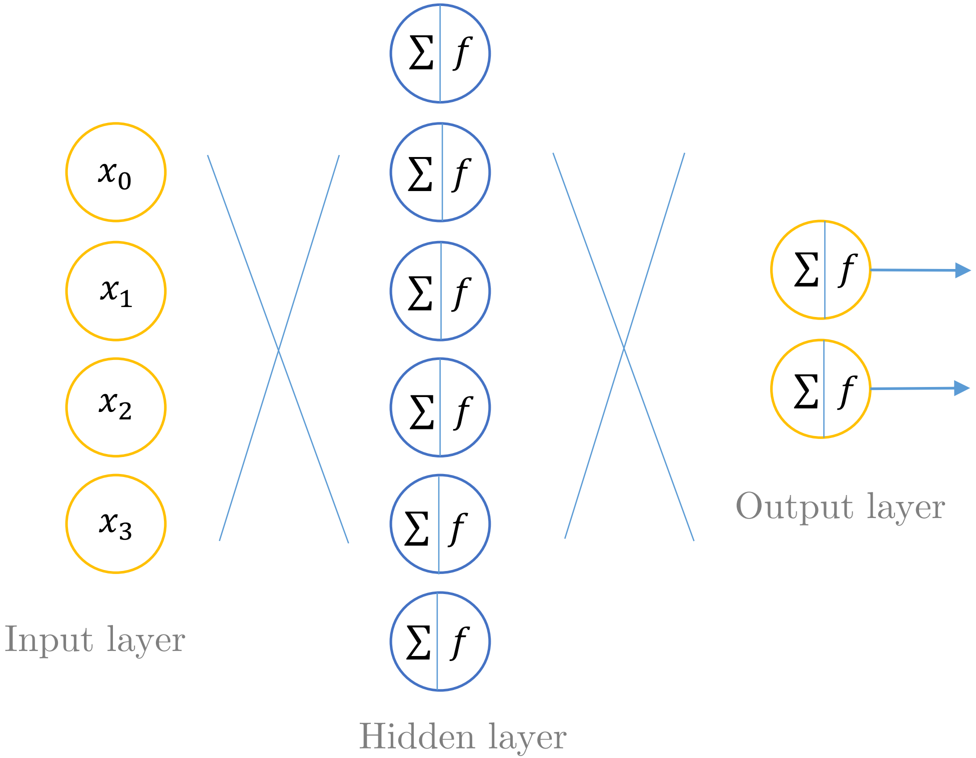
$$X = x_o, x_1, \ldots, x_n$$

$$W = w_o, w_1, \ldots, w_n$$

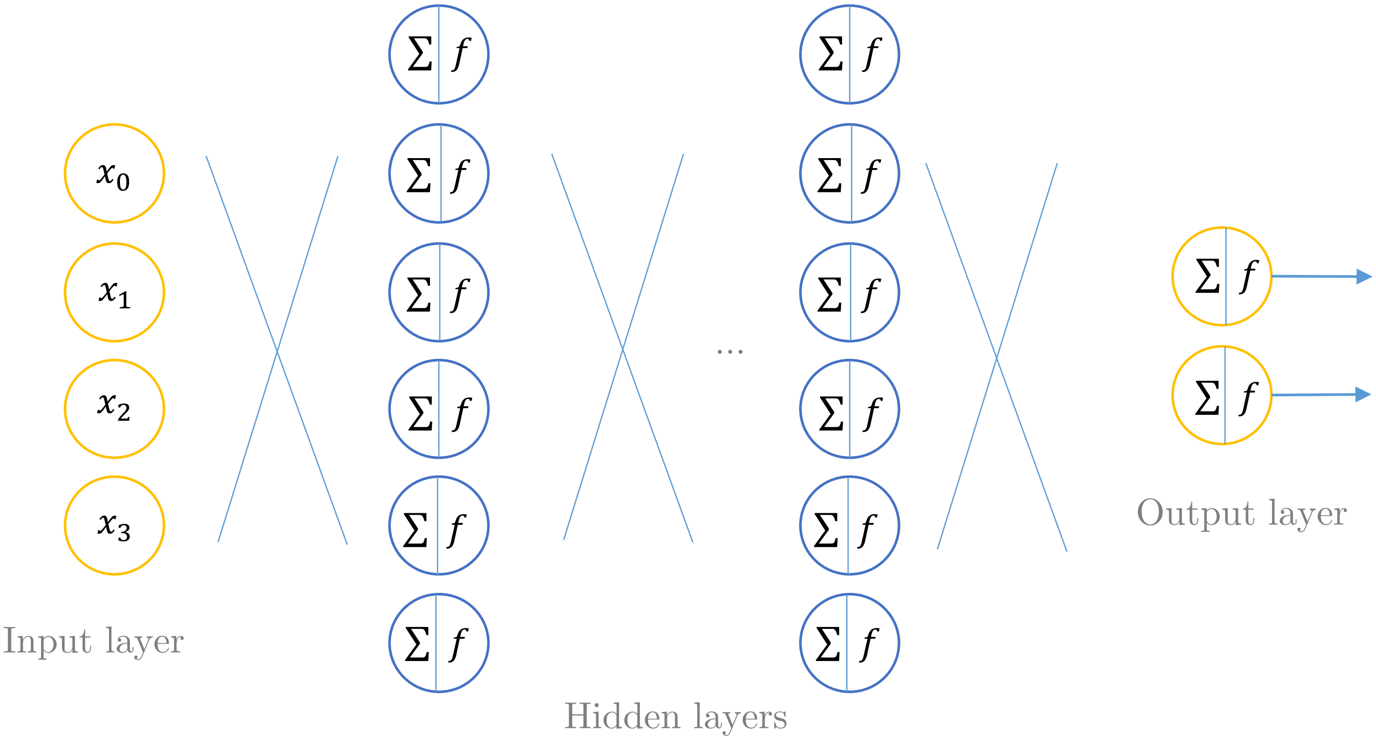


Elvis Saravia

Multi-Layer Perceptron (MLP)

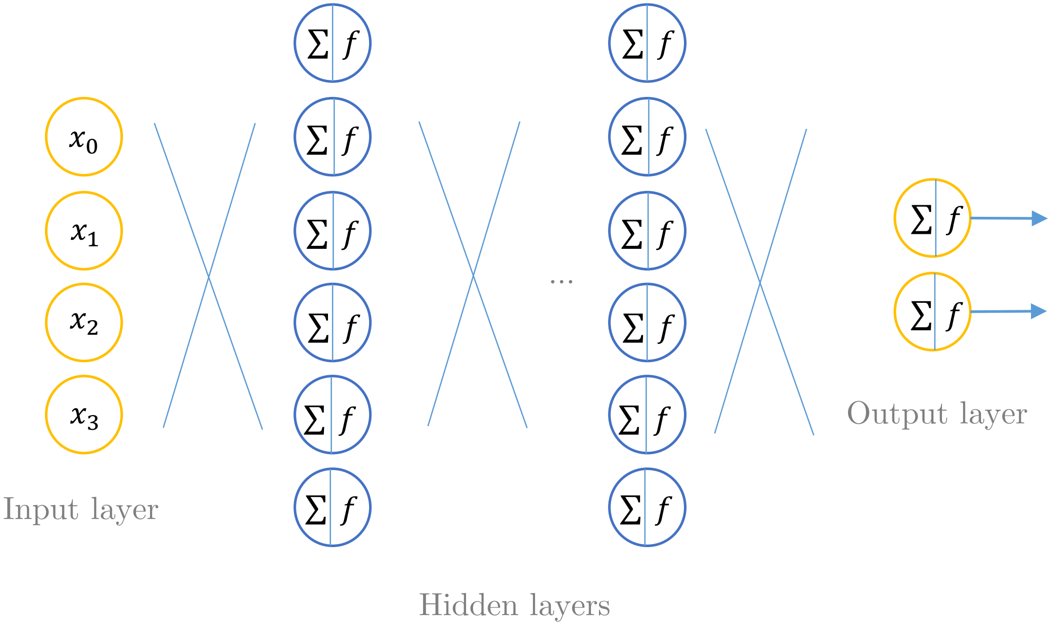


Deep Neural Networks (DNN)



Neural Network (Summary)

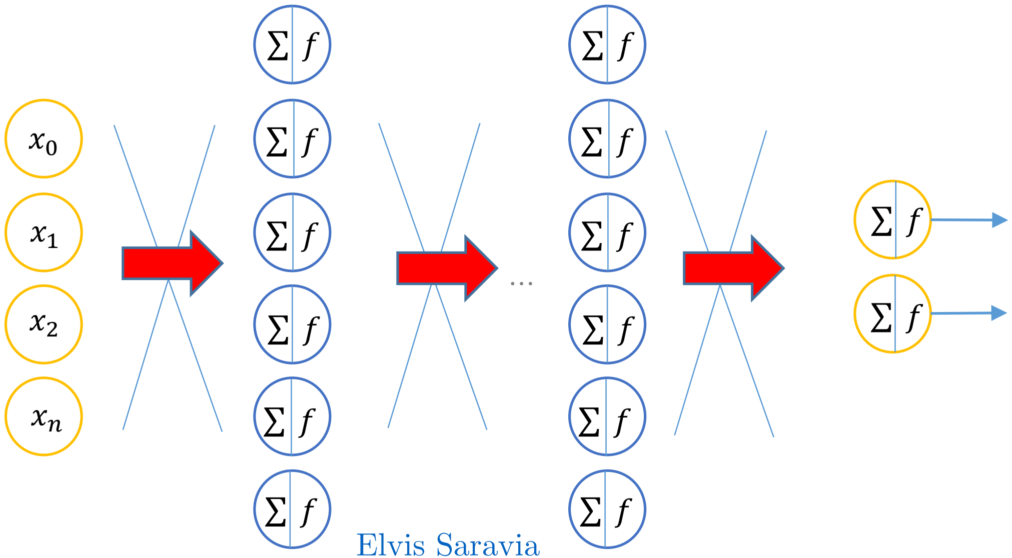
- Neural Networks learn features after input is fed into the hidden layers while updating weights through backpropagation (SGD)*.
- Data and activation flows in one direction through the hidden layers, but neurons never interact with each other.



* Stochastic Gradient Descent (SGD)

Drawbacks of NNs

- Lack sequence modeling capabilities: don't keep track of past information (i.e., no memory), which is very important to model data with a sequential nature.
- Input is of fixed size (more of this later)



Elvis Saravia

Part 2

Sequential Modeling

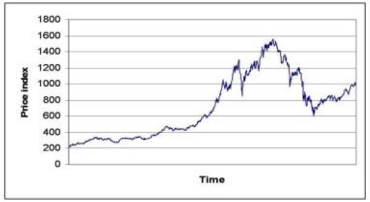
Elvis Saravia

Sequences

- Current values depend on previous values (e.g., melody notes, language rules)
- Order needs to be maintained to preserve meaning
- Sequences usually vary in length

"I took the bus this morning because it was very cold." Sentence

Speech waveform



Stock price

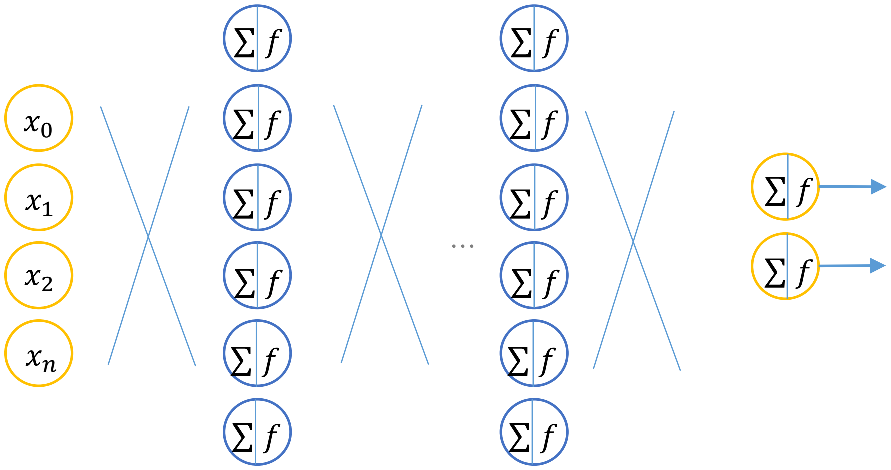
Elvis Saravia

Modeling Sequences

How to represent a sequence?

I love the coldness → $[\,1\ 1\ 0\ 1\ 0\ 0\ 1\,]^T$ →

Bag of Words (BOW)



What's the problem with the BOW representation?

Problem with BOW

Bag of words <u>does not preserve order</u>, therefore no semantics can be captured

"The food was good, not bad at all"

vs

"The food was bad, not good at all"

[1 1 0 1 1 0 1 0 1 0 0 1 1]

How to differentiate meaning of both sentences?

One-Hot Encoding

- Preserve order by maintaining <u>order within feature vector</u>

On Monday it was raining

[0 0 0 1 0 0 0 1 0 0 1 0 0 0 0 0 1 0 0 0 0 0 0 0 1]

We preserved order but what is the problem here?

Problem with One-Hot Encoding

- One-hot encoding cannot deal with variations of the same sequence.

On Monday it was raining ➡ [0 0 0 1 0 0 0 1 0 0 1 0 0 0 0 0 1 0 0 0 0 0 0 0 1]

$\neq$

It was raining on Monday ➡ [1 0 0 0 0 0 1 0 0 0 0 0 0 0 1 0 0 0 1 0 0 0 1 0 0]

Solution

Solution: We need to relearn the rules of language at each point in the sentence to preserve meaning.

On Monday it was raining

It was raining on Monday

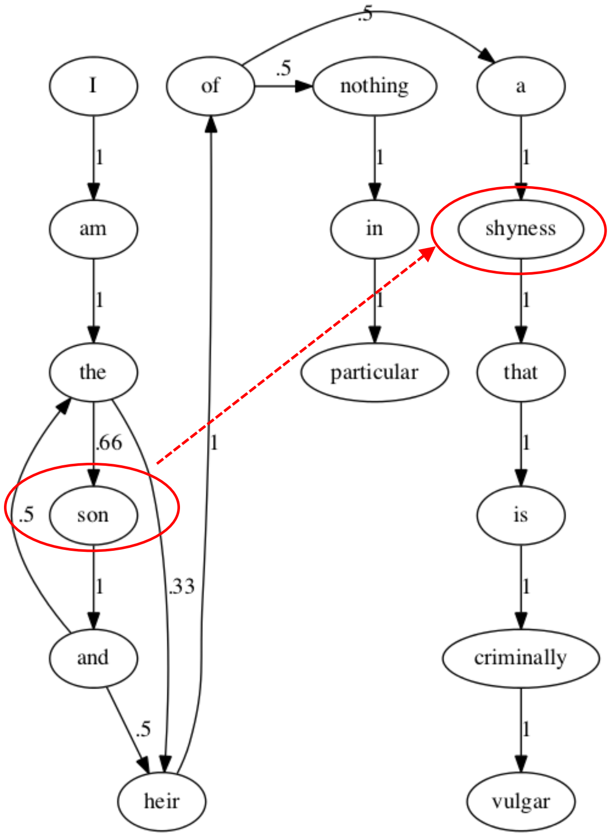
What was learned at the beginning of the sentence will have to be relearned at the end of the sentence.

No idea of state and what comes next!!!

Markov Models

State and transitions can be modeled, therefore it doesn't matter where in the sentence we are, we have an idea of what comes next based on the probabilities.

Problem: Each state depends only on the last state. *We can't model long-term dependencies!*



Elvis Saravia

Long-term dependencies

We need information from the far past and future to accurately model sequences.

In Italy, I had a great time and I learnt some of the _____ language

Elvis Saravia

It's time for Recurrent Neural Networks (RNNs)!!!

Part 3

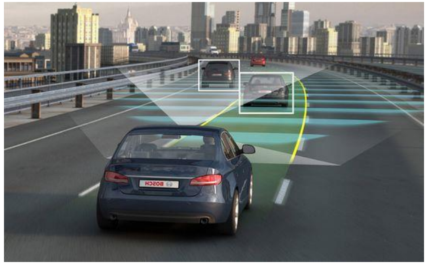
Recurrent Neural Networks
(RNNs)

Recurrent Neural Networks (RNNs)

- RNNs model sequential information by assuming long-term dependencies between elements of a sequence.
- RNNs maintain word order and share parameters across the sequence (i.e., no need to relearn rules).
- RNNs are recurrent because they perform the same task for every element of a sequence, with the output being depended on the previous computations.
- RNNs memorize information that has been computed so far, so they deal well with long-term dependencies.

Applications of RNN

- Analyze time series data to predict stock market
- Speech recognition (e.g., Emotion Recognition from Acoustic features)
- Autonomous driving
- Natural Language Processing (e.g., Machine Translation, Question and Answer)?




Elvis Saravia

Some examples

Google Magenta Project (Melody composer) – (https://magenta.tensorflow.org)

Sentence Generator - (http://goo.gl/onkPNd)

Image Captioning – (http://goo.gl/Nwx7Kh)

RNNs Main Components

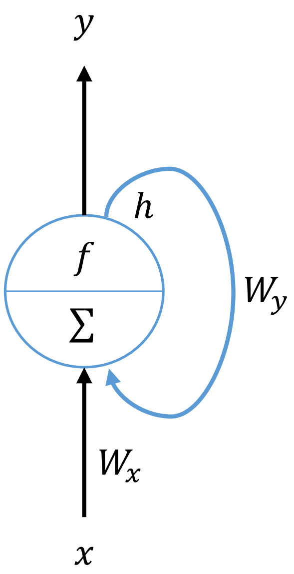
- Recurrent neurons
- Unrolling recurrent neurons
- Layer of recurrent neurons
- Memory cell containing hidden state

Elvis Saravia

Recurrent Neurons

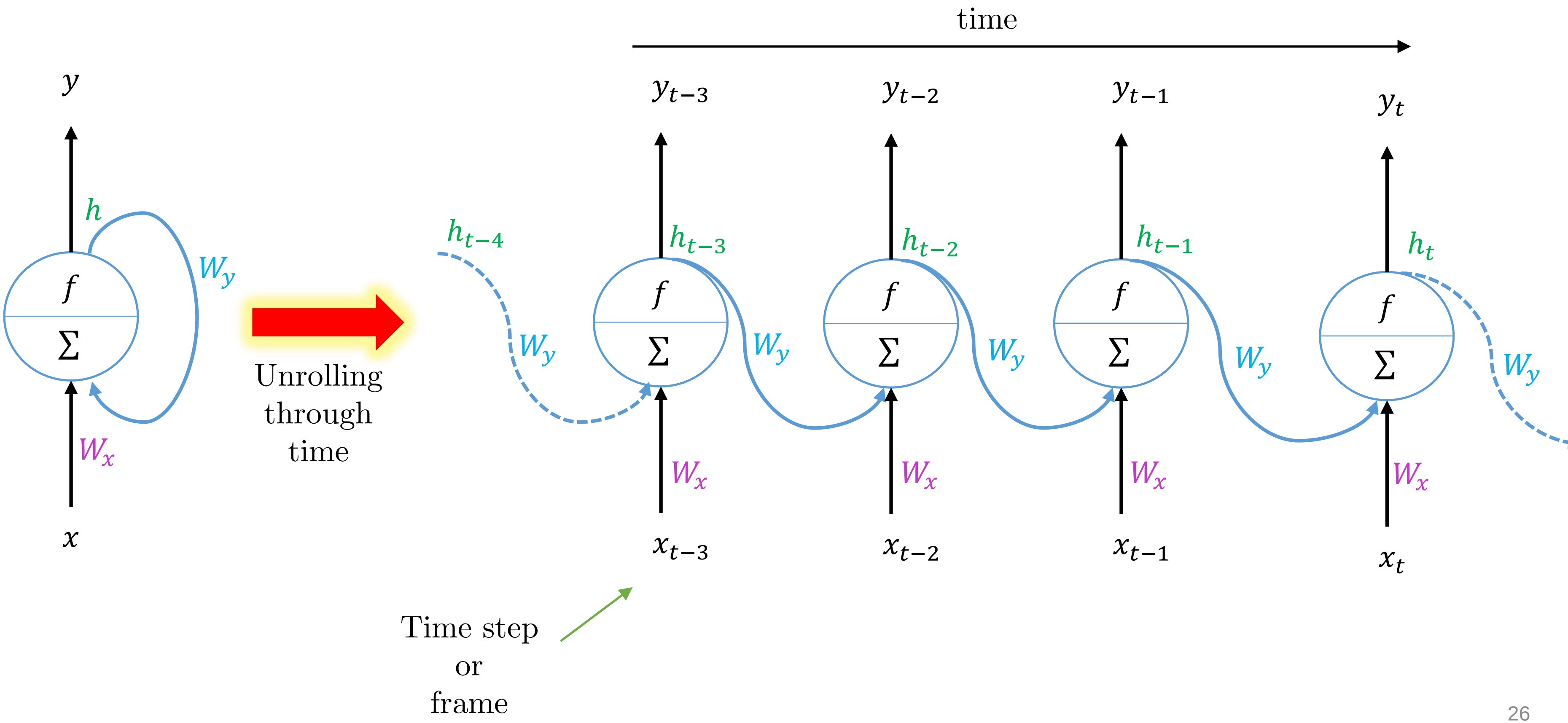
A simple recurrent neuron:

- receives input
- produces output
- sends output back to itself



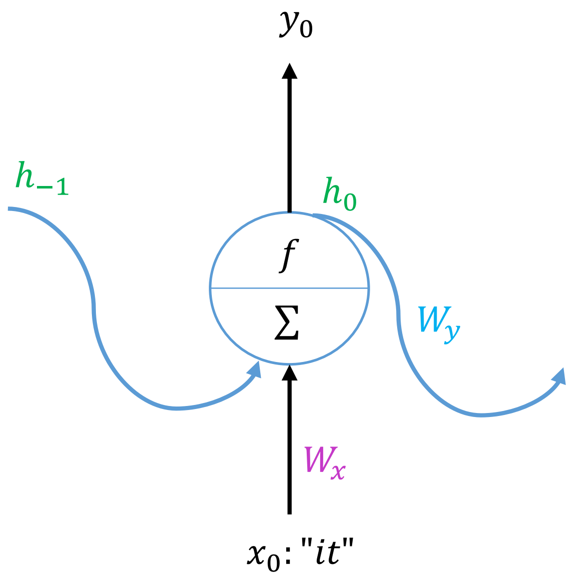
x - Input
y - Output (usually a vector of probabilities)
Σ - sum(W. x) + bias
f - Activation function (e.g., ReLU, tanh, sigmoid)
W_x - Weights for inputs
W_y - Weights for outputs of previous time step
h - Function of current inputs and previous time step

Unrolling/Unfolding recurrent neuron



RNNs remember previous state

t=0



x_0 : vector representing first word

y_0 : output at t=0

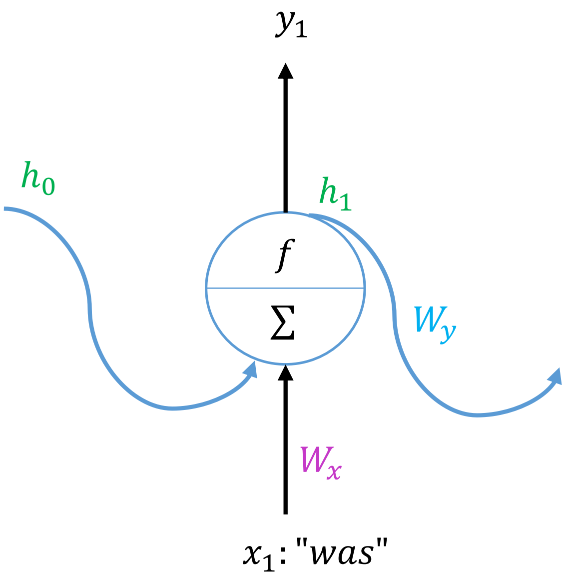
$$h_0 = \tanh(W_x x_0 + \cancel{W_y h_{-1}})$$

Can remember things from the past

W_x, W_y: weight matrices

RNNs remember previous state

t=1



x_1 : vector representing second word

y_1 : output at t=1

$$h_1 = \tanh(W_x x_1 + W_y h_0)$$

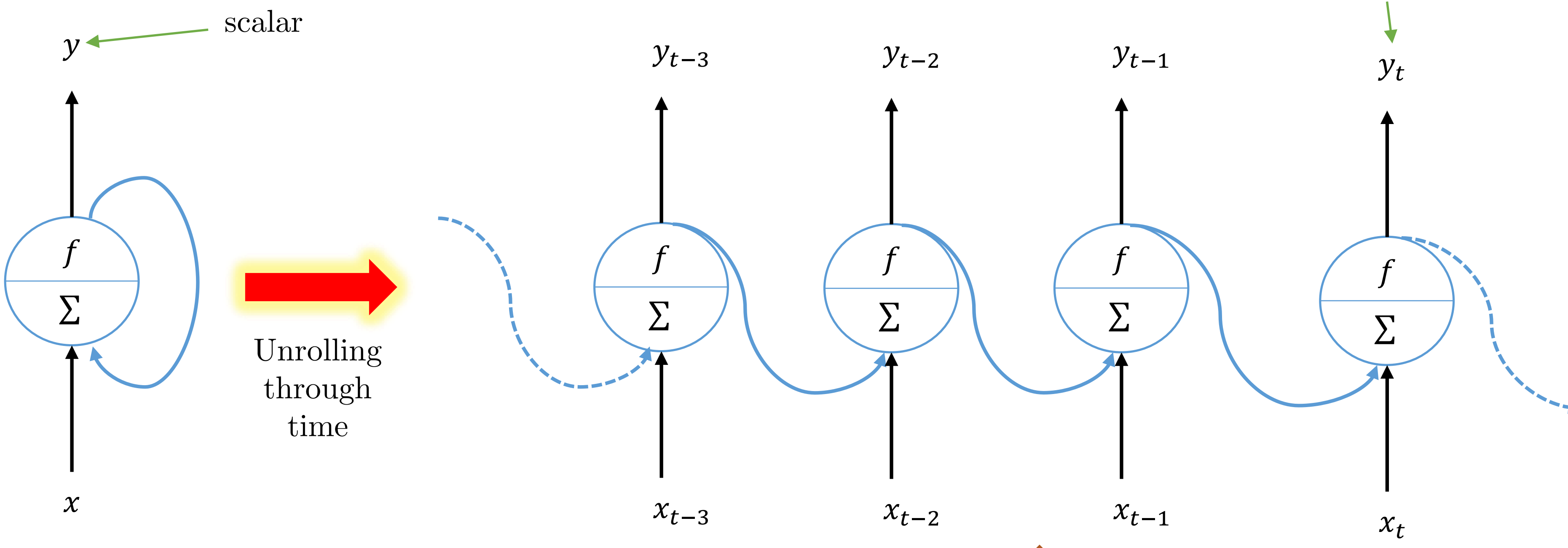
$$h_0 = \tanh(W_x x_0 + \cancel{W_y h_{-1}})$$

Can remember things from t=0

W_x, W_y: weight matrices stay the same so they are shared across sequence

Overview

$$y_t = f\left(W_x x_t + W_y y_{t-1} + b\right)$$



batch

[0,1,2]	[3,4,5]	[6,7,8]	[9,0,1]
[9,8,7]	[0,0,0]	[6,5,4]	[3,2,1]

Code Example

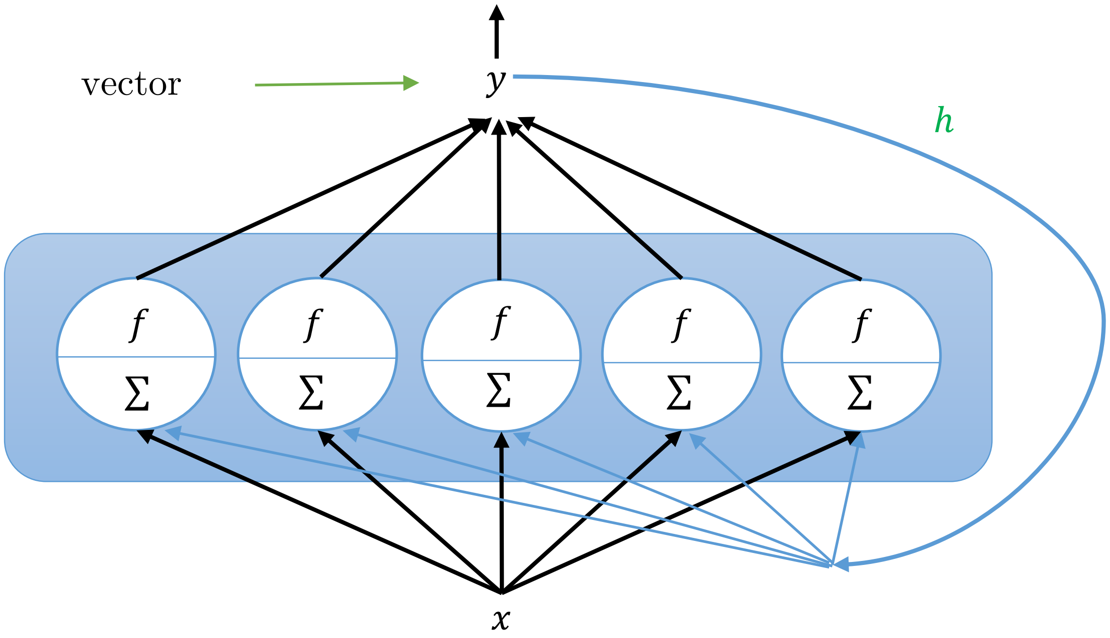
$$y_t = f\left(W_x x_t + W_y y_{t-1} + b\right)$$

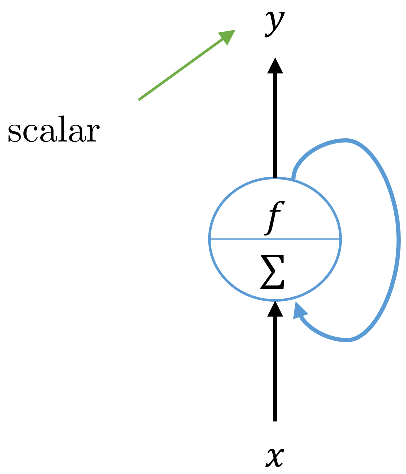
Where all the magic happens!

```python
# RNN unrolled through two time steps
N_INPUTS = 3 # number of features in input
N_NEURONS = 5

class BasicRNN(object):
    def __init__(self, n_inputs, n_neurons):
        self.X0 = tf.placeholder(tf.float32, [None, n_inputs])
        self.X1 = tf.placeholder(tf.float32, [None, n_inputs])

        Wx = tf.Variable(tf.random_normal(shape=[n_inputs, n_neurons], dtype=tf.float32))
        Wy = tf.Variable(tf.random_normal(shape=[n_neurons, n_neurons], dtype=tf.float32))
        b = tf.Variable(tf.zeros([1, n_neurons], dtype=tf.float32))

        self.Y0 = tf.tanh(tf.matmul(self.X0, Wx) + b)
        self.Y1 = tf.tanh(tf.matmul(self.Y0, Wy) + tf.matmul(self.X1, Wx) + b)
```

```python
# Now we feed input at both time steps

# Generate mini-batch with 4 instances (i.e., each instance has an input sequence of exactly two inputs)
X0_batch = np.array([[0,1,2], [3,4,5], [6,7,8], [9,0,1]]) # t = 0
X1_batch = np.array([[9,8,7], [0,0,0], [6,5,4], [3,2,1]]) # t = 1

model = BasicRNN(N_INPUTS, N_NEURONS)
with tf.Session() as sess:
    # initialize and run all variables so that we can use their values directly
    init = tf.global_variables_initializer()
    sess.run(init)
    Y0_val, Y1_val = sess.run([model.Y0, model.Y1], feed_dict={model.X0: X0_batch, model.X1: X1_batch})
```

Layer of Recurrent Neurons





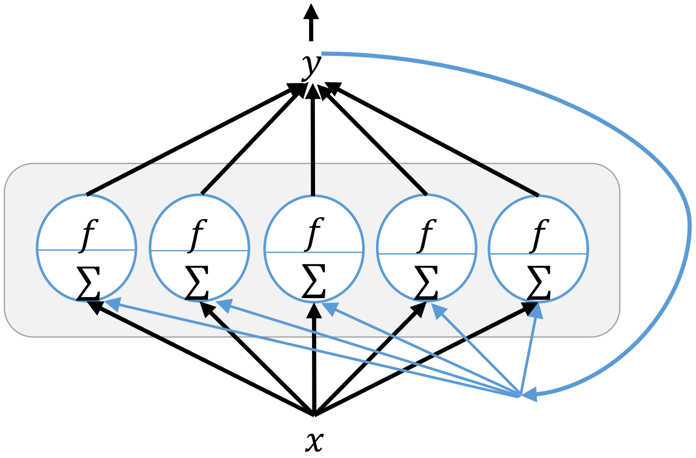
Elvis Saravia

Unrolling Layer

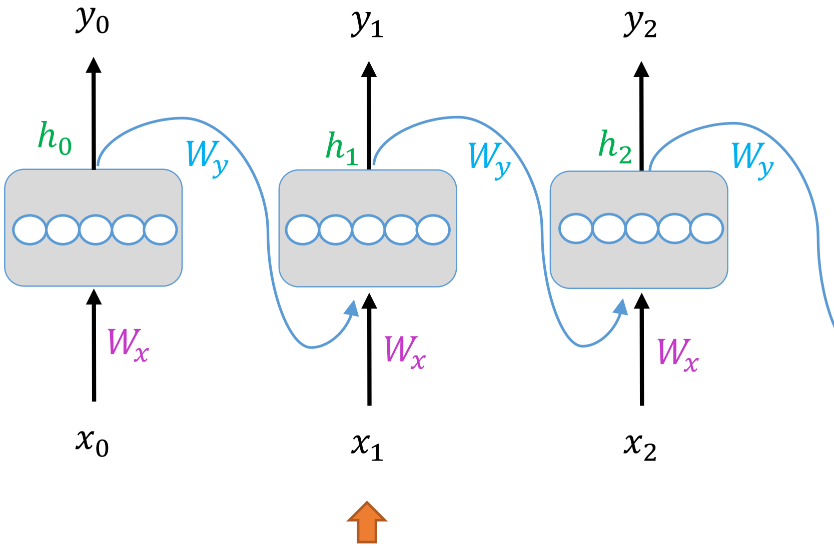
$$Y_t = f\left(X_t.W_x + Y_{t-1}.W_y + b\right)$$

vector form

$$Y_t = f([X_t \quad Y_{t-1}].W + b), \qquad W = \begin{bmatrix} W_x \\ W_y \end{bmatrix}$$

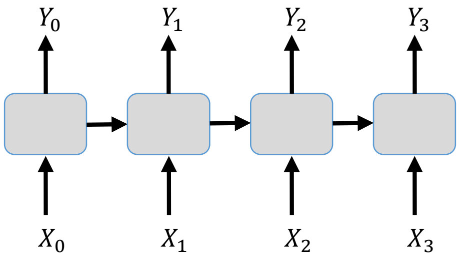


Unrolling through time



[1,2,3]	[1,2,3]	[1,2,3]
[4,5,6]	[7,8,9]	[0,0,0]

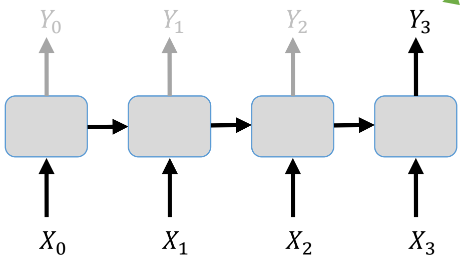
Variations of RNNs: Input / Output



sequence to sequence

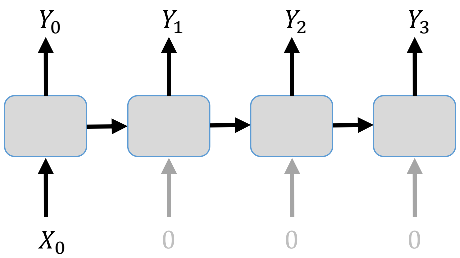
- Stock price
- Other time series

vector of
probabilities
over classes
a.k.a softmax



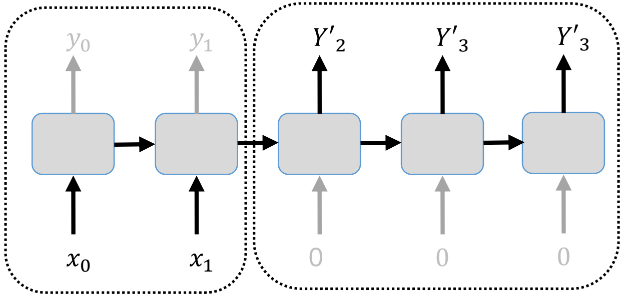
sequence to vector

- Sentiment analysis ([-1,+1])
- Other classification tasks



vector to sequence

- Image captioning



Encoder **Decoder**

- Translation:
- E: Sequence to vector
- D: Vector to sequence

Training

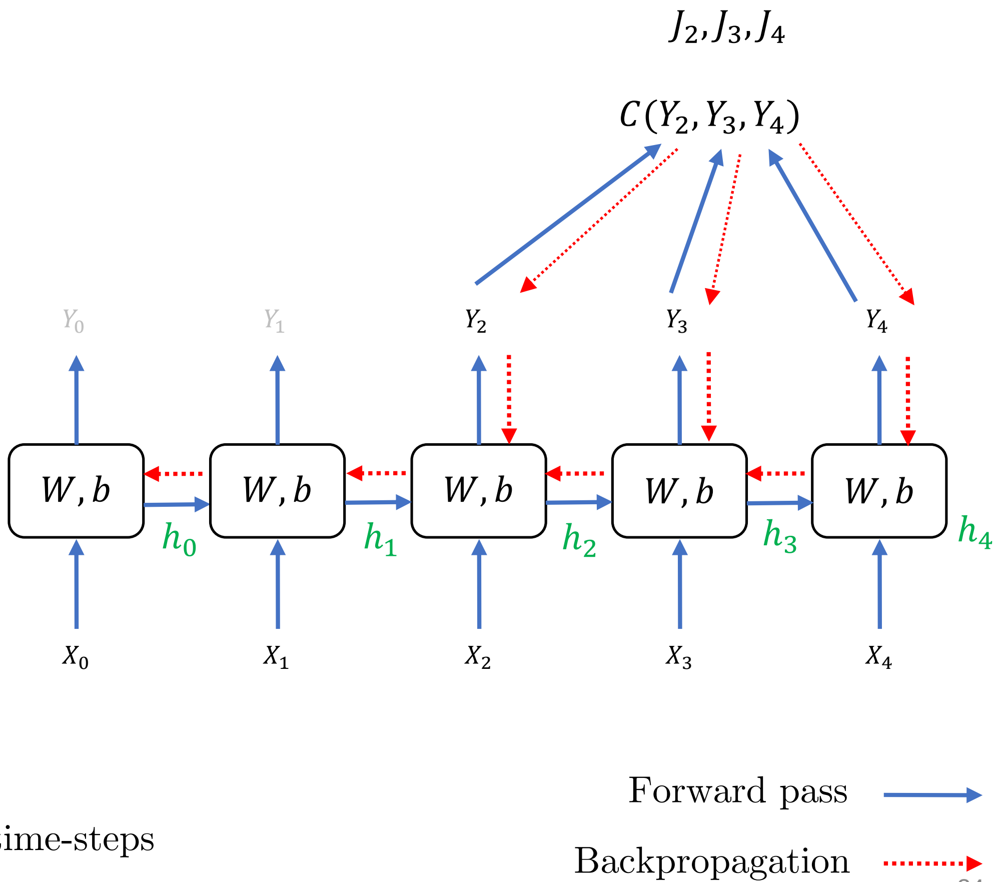
- Forward pass
- Compute Loss via cost function C
- Minimize Loss by backpropagation through time (BPTT)

$$\frac{\partial J}{\partial W} = \sum_t \frac{\partial J_t}{\partial W}$$

For one single time step t:

$$\frac{\partial J_4}{\partial W} = \sum_{k=0}^{4} \frac{\partial J_4}{\partial y_4} \frac{\partial y_4}{\partial h_4} \frac{\partial h_4}{\partial h_k} \frac{\partial h_k}{\partial W}$$

Counting the contributions of W in previous time-steps to the error at time-step t (using Chain rule)



Drawbacks

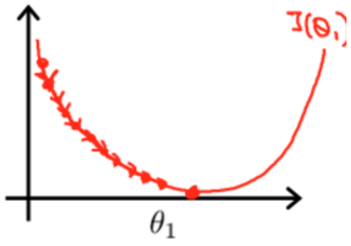
Main problem: Vanishing gradients (gradients gets too small)

Intuition: As sequences get longer, gradients tend to get too small during backpropagation process.

$$\frac{\partial J_n}{\partial W} = \sum_{k=0}^{n} \frac{\partial J_n}{\partial y_n} \frac{\partial y_n}{\partial h_n} \frac{\partial h_n}{\partial h_k} \frac{\partial h_k}{\partial W}$$

$$\frac{\partial h_n}{\partial h_{n-1}} \frac{\partial h_{n-1}}{\partial h_{n-2}} \cdots \frac{\partial h_3}{\partial h_2} \frac{\partial h_2}{\partial h_1} \frac{\partial h_1}{\partial h_0}$$

We are just multiplying a lot of small numbers together



Elvis Saravia

Solutions

Long Short-Term Memory Networks –

Deal with vanishing gradient problem, therefore more reliable to model long-term dependencies, especially for very long sequences.

RNN Extensions

Extended Readings:

- Bidirectional RNNs – passing states in both directions
- Deep (Bidirectional) RNNs – stacking RNNs
- LSTM networks – Adaptation of RNNs

Elvis Saravia

In general

- RNNs are great for analyzing sequences of any arbitrary length.
- RNNs are considered “anticipatory” models
- RNNs are also considered creative learning models as they can, for example, predict set of musical notes to play next in melody, and selects an appropriate one.

Part 3

RNNs in Tensorflow

Elvis Saravia

Demo

- Building RNNS in Tensorflow
- Trainining RNNS in Tensorflow
- Image Classification
- Text Classification

References

- Introduction to RNNs - http://www.wildml.com/2015/09/recurrent-neural-networks-tutorial-part-1-introduction-to-rnns/
- NTHU Machine Learning Course - https://goo.gl/B4EqMi
- Hands-On Machine Learning with Scikit-Learn and Tensorflow (Book)