THE STRUCTURE OF AN LSTM NETWORK

Evan Conley
MTH 490

Defining Neural Networks

Neural Network:

- Mimics biological neural networks
- Made up of ‘neurons’ and ‘synapses’
- Neuron Form: $\sigma(Wx + b)$
- Functional Approximators

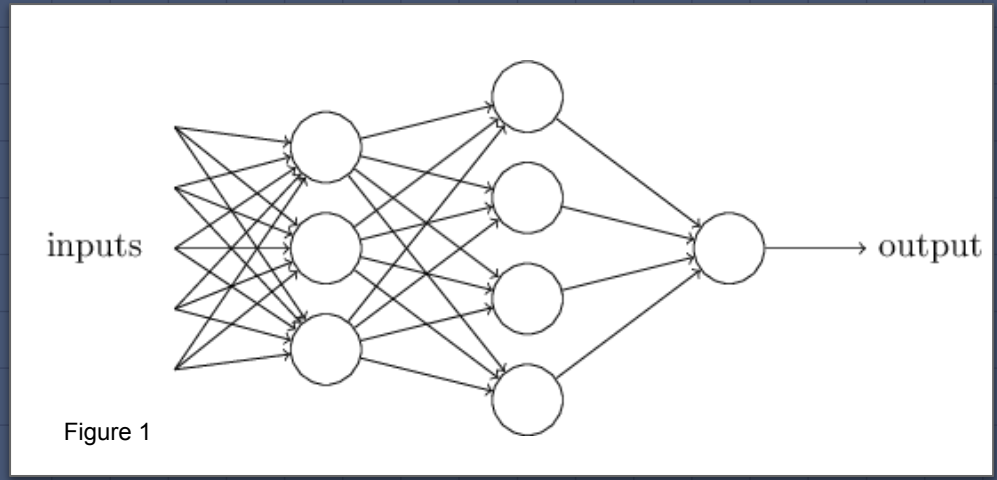


Figure 1

Figure 1. Retreived from: http://neuralnetworksanddeeplearning.com/chap1.html

Defining Classification

Classification:

- Supervised learning methodology
- We seek a function f: X → Y, where X is an input space and Y is the finite set of known labels.

Our Problem: Taking IMDB movie review data and outputting a classification label of either ‘positive’ or ‘negative’.

"[...]There are sequences in this movie that simply do not need to exist and provide literally nothing to the story.

[...] All that said, the special effects look and sound great, and it was fun to see Mark Hamill acting again. Other than that this movie is a disgrace not just to the franchise but to anyone with a brain. Stupid, stupid, stupid."

Input Layer

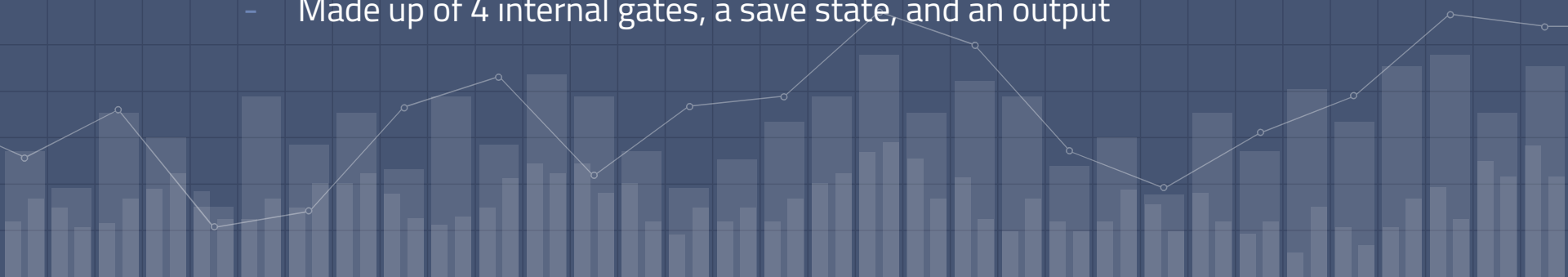
Word Embeddings:

- Utilizing a pre-trained word embeddings model
- Gensim Stanford word vectorization model
- Input Simplified: If word exists in embedding matrix, use it, else, use 0 as a placeholder.

Design: Architecture

LSTM:

- Long-Short Term Memory Network
- Recurrent Network Architecture adept at handling time series data
- Made up of 4 internal gates, a save state, and an output



Activation Functions

Nonlinearities:

Wx + b is of the form of a linear equation… not able to approximate nonlinear equations.

Activation functions break linearity, so the network can approximate nonlinear systems.

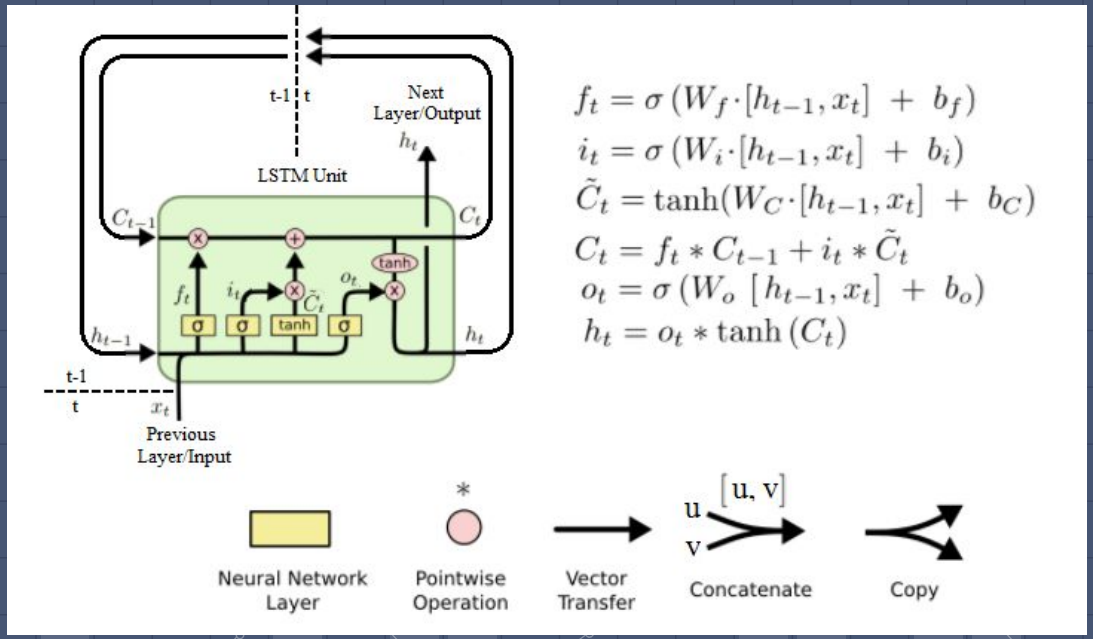
$\tanh(x) = (1-e^{-2x})/(1+e^{-2x})$

- Final activation at output step of cell.
- Partial derivative resists 0 convergence
- Range -1 to 1

$\sigma(x) = 1/(1+e^{-x})$

- Represents firing rate of each gate in
- Range: 0 to 1
- Neuron (0 is no fire, 1 is full fire)

The LSTM Units



$$f_t = \sigma\left(W_f \cdot [h_{t-1}, x_t] \; + \; b_f\right)$$
$$i_t = \sigma\left(W_i \cdot [h_{t-1}, x_t] \; + \; b_i\right)$$
$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] \; + \; b_C)$$
$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$$
$$o_t = \sigma\left(W_o \; [h_{t-1}, x_t] \; + \; b_o\right)$$
$$h_t = o_t * \tanh\left(C_t\right)$$

Figure 2

Figure 2. Retreived from: http://colah.github.io/posts/2015-08-Understanding-LSTMs/

Output Layer

Softmax Classification Layer

- A form of the logistic equation, this function takes in an output vector (each row of the output matrix) and outputs a vector of the same size where all indices add to 1

$$\sigma(\mathbf{z})_j = \frac{e^{z_j}}{\sum_{k=1}^{K} e^{z_k}} \quad \text{for } j = 1, \ldots, K.$$

Figure 3

- Chosen as it creates probability distribution, allotting some probability for each class label

Figure 3. Retrieved from: https://en.wikipedia.org/wiki/Softmax_function

Training:

STEP 1 | STEP 2 | STEP 3

Forward propagation:

Process, by which, the network uses current weights to complete a forward pass, resulting in an output.

Loss:

The error (or loss) determined by a loss function.

$$H(p,q) = -\sum_{i=1}^{m} y_i log_2(q_i)$$

Backpropagation:

The process of ‘learning’ where is error is propagated back in the network and weights are updated.

Performance Of Our Network

LSTM Performance					
Model	Accuracy	Sensitivity (TPR)	Specificity (TNR)	Miss Rate (FNR)	Fallout (FPR)
50/25/25 Epochs:2	0.8487	0.8174	0.8024	0.1826	0.1976
50/25/25 Epochs:4	0.8243	0.8159	0.813	0.1841	0.187
50/0/50 Epochs:2	0.8146	0.8294	0.8302	0.1706	0.1698
50/0/50 Epochs:4	0.831	0.8451	0.8453	0.1549	0.1547

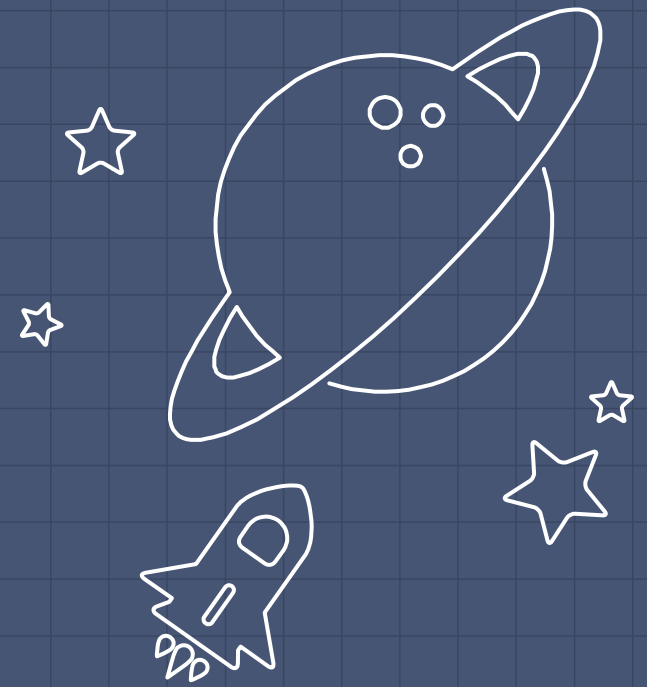
Output on Example:

```
Classification of your sentiment:

Unformatted network output:
        [[0.9620251  0.03797485]]

Actual classification label:
        Negative Sentiment!
```

Thank You!



For those wondering...

- I utilized Python as the language for the net
- Used the Keras on top of Tensorflow to build network
- Used sklearn to do train/test/validation split and creation of confusion matrices.

Similar Model

- GRU (Gated Recurrent Unit)
- Much simpler / More efficient
- Similar results

Created other models as well...

- These included convolutional, bi-directional LSTM, and GRU networks.
- Surprisingly, convolutional neural network, was the overall winner in all rounds.
- Found it was overfit, though.