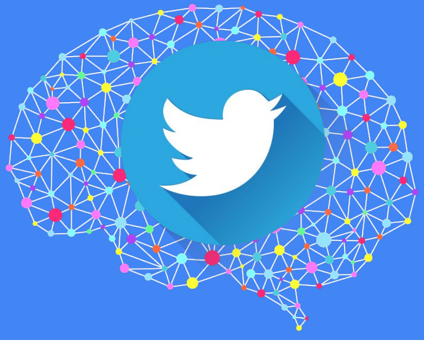
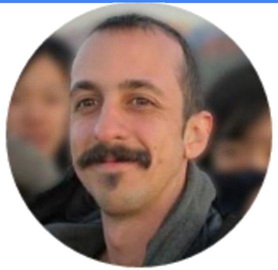
Twitter Sentiment Analysis with Neural Networks



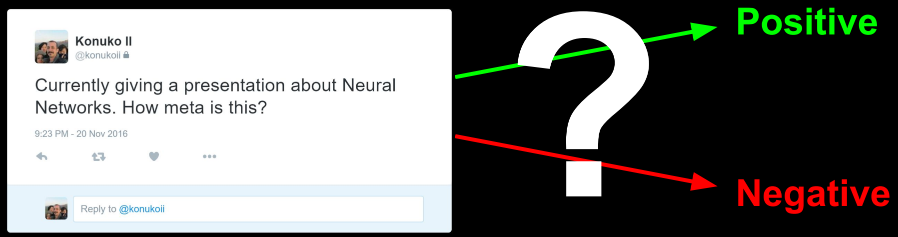


Pedro M. Sosa - Shayan Sadigh

Motivation:

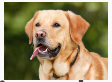
- Understand and **build** our own NN.
- Solve a real-world problem.

Twitter Sentiment Analysis



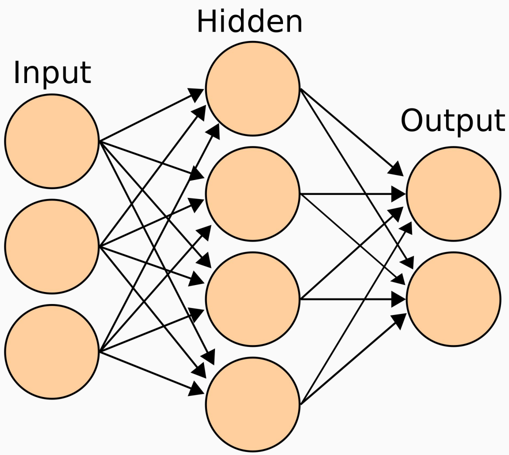
Understanding Neural Networks

Basics

- Just another supervised learning algorithm like SVMs or random forests.
- Takes an input and produces an output.
    - E.G. Input:  Output: **Dog**
- Must first be trained on a dataset of input / output pairs.

Understanding Neural Networks

Inside the Network



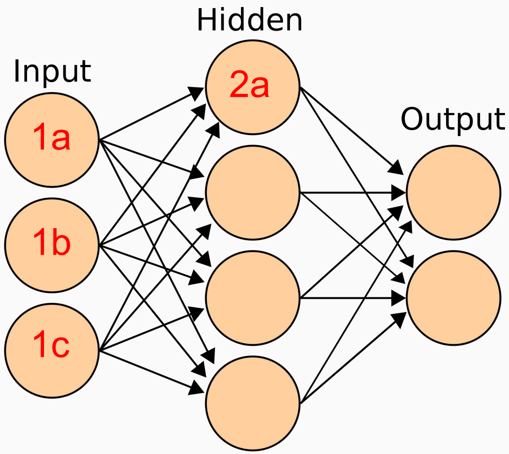
Neural networks are a bunch of nodes separated into layers. Each node in a layer has directed edges pointing to nodes in the next layer.

The first layer is called the input layer, the last is the output layer, and anything in-between are hidden layers.

Understanding Neural Networks

Forward Propagation

Let's focus on the 4 labeled nodes.



Input to 2a = input to 1a + input to 1b + input to 1c

Each directed line actually has its own unique "weight" associated with it. So the input to 2a is more like:

w1(input to 1a) + w2(input to 1b) + w3(input to 1c)

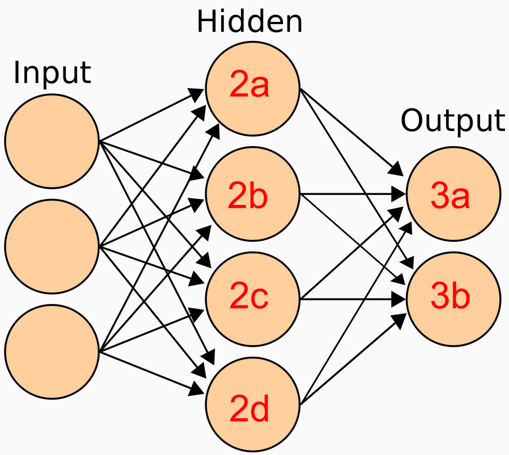
Actually, a fancy function called the "activation function" is applied after we find the above sum, so it's really more like:

activation(w1(input to 1a) + w2(input to 1b) + w3(input to 1c))

The activation function simply normalizes the sum (it maps huge numbers to a number between 0 and 1, or alternatively -1 and 1).

Understanding Neural Networks

Forward Propagation



The result from the previous slide is passed forward to the nodes in the next layer, 3a and 3b.

3a and 3b also receive inputs from 2b, 2c, and 2d.

Just like in the previous slide, the inputs are summed up and an activation function is applied.

3a = activation(w11(input to 2a) + w12(input to 2b) + w13(input to 2c))
3b = activation(w21(input to 2a) + w22(input to 2b) + w23(input to 2c))

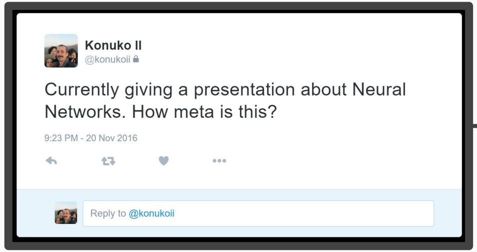
This whole process is called forward propagation. The output of the neural network is whatever the value is at the nodes in the output layer, 3a and 3b. (In this case a 2D vector).

Data Representation

Defining the Input to our Neural Network

Data Representation

How to represent tweets?



Numerical Representation
(without Losing Information)

P

Data Representation

1. Naive Bayesian Probabilities

I Like Pie!

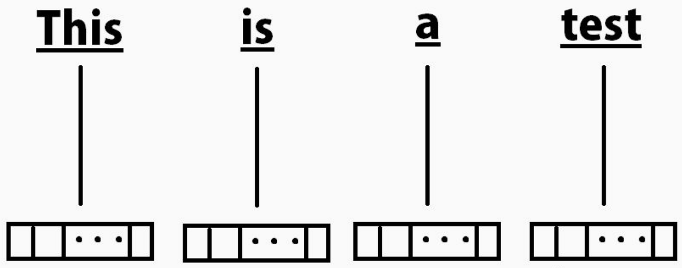
P('I' = Good) , P('I' = Bad) , P('Like' = Good) , P('Like' = Bad) , P('Pie' = Good) , P('Pie' = Bad)

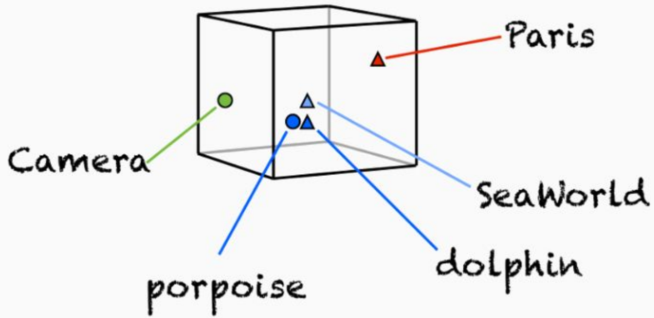
0.5 0.5 0.9 0.4 0.4 0.3

Data Representation

2. Word Embeddings

- New methodology and gaining a lot of momentum.
- Each word Represented as n-dimensional vectors.
- Built our own Word Embeddings using Keras library and our own dataset.
- Decided to use 32 dimensional vectors.





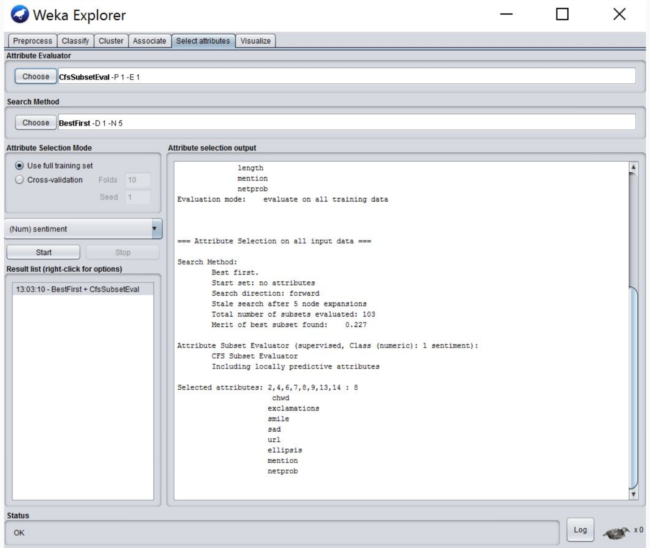
Data Representation

3. Feature Vector

- Focus on features “unique” to a Tweet

- Selected multiple features, and tested their usefulness with WEKA.

- Correlation-based Feature Subset Selection*
    - Individual Predictive Ability + Redundancy

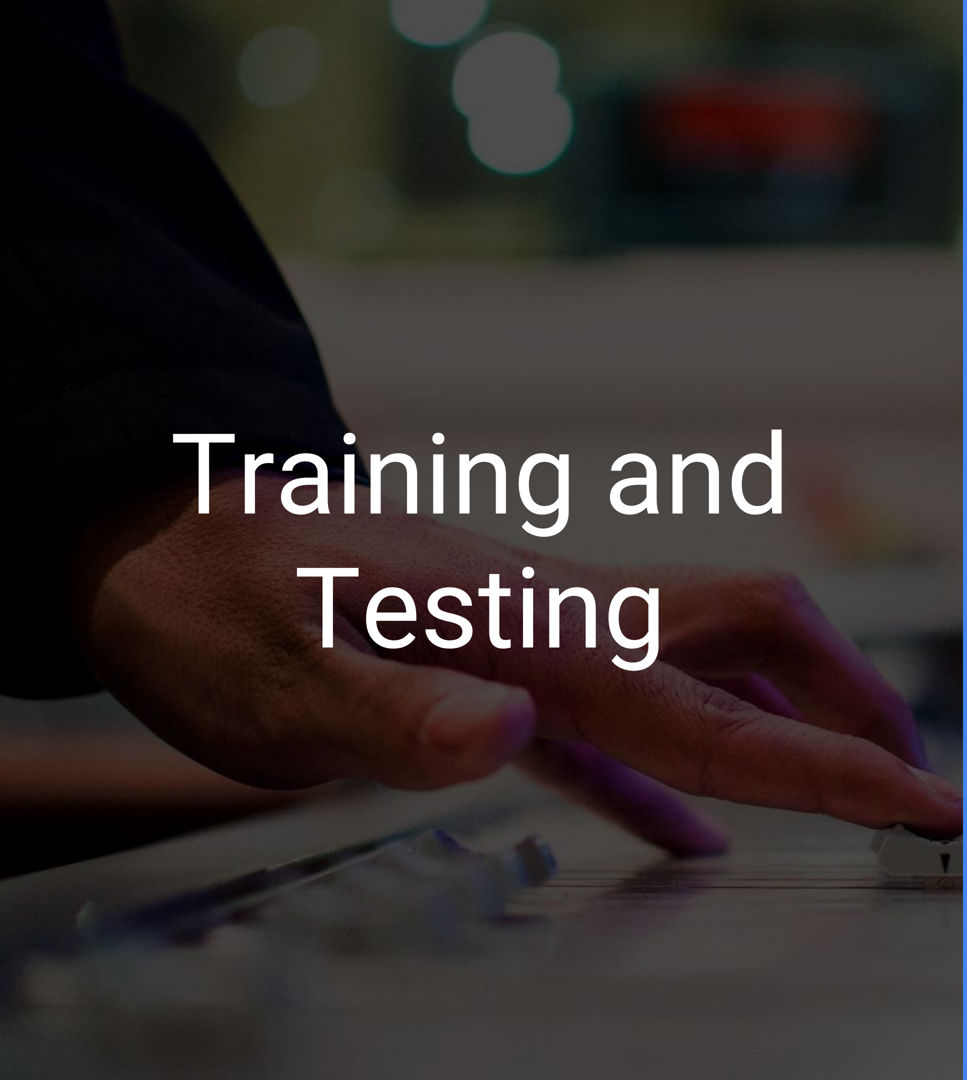


*M. A. Hall (1998). Correlation-based Feature Subset Selection for Machine Learning. Hamilton, New Zealand.

Data Representation

3. Feature Vector (contd.)

Feature Name	Description	Selected
chwd	# of Characters/ # of Words	✓
exclamation	# of exclamation marks (!)	✓
smile	# of positive emoticons	✓
sad	# of negative emoticons	✓
url	# of URLs shared	✓
ellipsis	# of ellipsis (...)	✓
mention	# of mentions (@someone)	✓
netprob	$\sum_{W \in Tweet}(P_{pos}(W) - P_{neg}(W))$	✓
question	# of question marks (?)	-
pronoun	# of pronouns (I, me, mine...)	-
hashtags	# of hashtags (#topic)	-
capitals	# of uppercase letters	-
length	Length of the Tweet	-



Training and Testing

Running Our Neural Network

Training and Testing

Setup

Our Own Neural Network

FeedFoward w/ BackProp
Bias, Activation Sigmoid Function

VS

Dataset: Kaggle
Train/Test Sets: 10k
Balanced Good/Bad

Keras (w/ Tensorflow Backend)

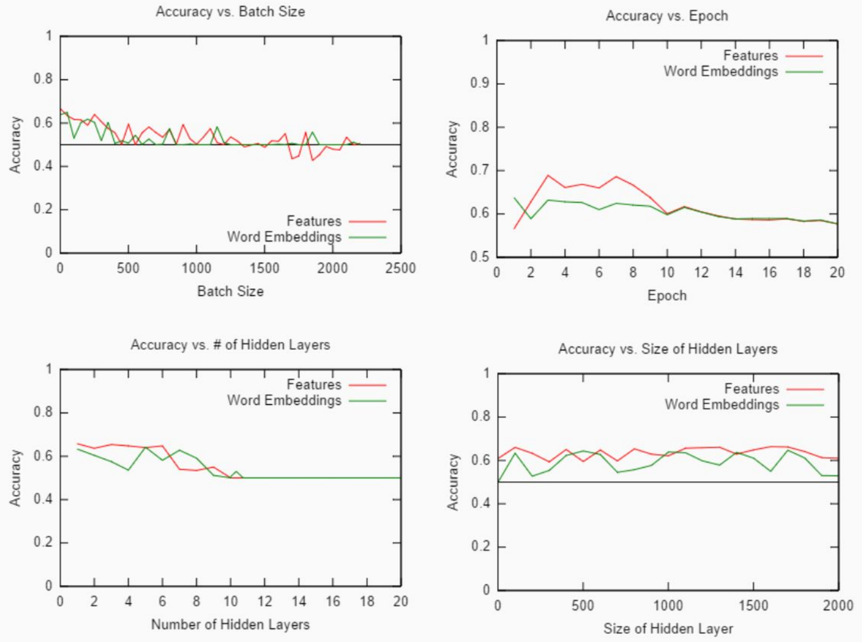
Same setup as our own NN.

S

Training and Testing

Fine Tuning Parameters

- **Batch Size**
  - Small: Good but costly (performance).
  - Big: Mean gradient loses precision.
- **Epoch**
  - Small: Subpar learning.
  - Big: Overfitting is possible.
- **# of Hidden Layers**
  - Small: Good, but could lose learning capacity.
  - Big: Vanishing Gradient Problem. Poor learning.
- **Size of Hidden Layer**
  - Small: Good, but could lose learning capacity.
  - Big: Still good, but unnecessarily costly (performance)



Results for Keras NN

P

Training and Testing

Experiments: Keras

100 Trial Runs with **Keras**
1 output node, 1 hidden layer, 250 hidden nodes, 3 epoch, default rate, size 1 batches

Input Type	Average Acc.	Max Acc.	Min Acc.	Std. Dev.
Naive Probabilities (size 128 batches)	63.04%	63.82%	62.32%	0.0053
Word Embeddings	61.40%	65.05%	58.30%	0.0150
Feature Vectors	**67.20%**	71.15%	63.30%	0.0177

Training and Testing

Experiments: Our Network

100 Trial Runs with **our Neural Network**
2 output nodes, 1 hidden layer, 20 hidden nodes, 100 epochs, 0.1 rate, size 50 batches

Input Type	Average Acc.	Max Acc.	Min Acc.	Std. Dev.
Naive Probabilities	62.90%	63.92%	62.14%	0.0070
Word Embeddings	58.43%	61.77%	55.02%	0.0162
Feature Vectors	**66.80%**	68.75%	60.35%	0.0158

Future Work

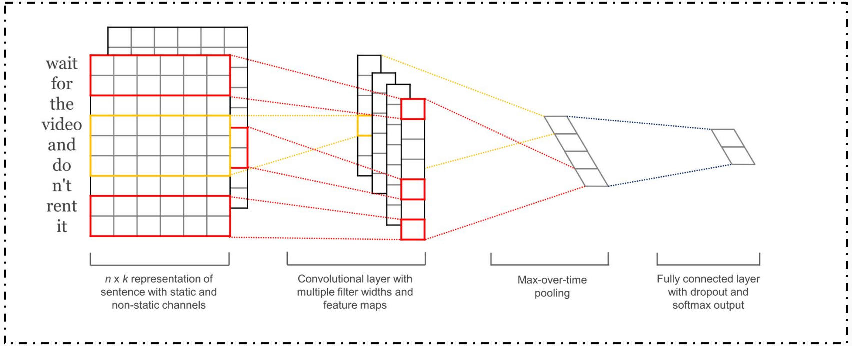
Convolutional Neural Networks

Convolutional Neural Networks

- Convolutional Networks learn and look for certain patterns that might show up on any given segment of a tweet.

- For example it can learn that phrases such as *“down to earth”* mean a positive sentiment, albeit “down” being generally associated as negative.

- Implemented with Keras NN using the Word Embeddings as input.
    - **Obtained a 2.81% Increase in Accuracy**



NN Type	Average Acc.	Max Acc.	Min Acc.	Std. Dev.
Original NN	61.40%	65.05%	58.30%	0.0150
CNN	64.21%	67.05%	62.10%	0.0134

Conclusions

Understood, Experimented and Implemented NN concepts.
Successfully built our own NN.

Paper, Code & Reference shared in Github (github.com/pmsosa/CS273-Project).
Giving others a good starting point!

NNs are a viable solution the Twitter Sentiment Analysis challenge.
Key seems to be how to represent the datasets.