A Neuron

• If you know Logistic Regression, then you already understand a basic neural network neuron!
A Neuron
is essentially a binary logistic regression unit

\[ h_{w,b}(x) = f(w^T x + b) \]

\[ f(z) = \frac{1}{1 + e^{-z}} \]

\( b \): We can have an “always on” feature, which gives a class prior, or separate it out, as a bias term.

\( w, b \) are the parameters of this neuron i.e., this logistic regression model.
A Neural Network

= running several logistic regressions at the same time

If we feed a vector of inputs through a bunch of logistic regression functions, then we get a vector of outputs ...
A Neural Network
= running several logistic regressions at the same time

... which we can feed into another logistic regression function

It is the loss function that will direct what the intermediate hidden variables should be, so as to do a good job at predicting the targets for the next layer, etc.
A Neural Network

= running several logistic regressions at the same time

Before we know it, we have a multilayer neural network....
\[ f: \text{Activation Function} \]

We have

\[
\begin{align*}
a_1 &= f(W_{11}x_1 + W_{12}x_2 + W_{13}x_3 + b_1) \\
a_2 &= f(W_{21}x_1 + W_{22}x_2 + W_{23}x_3 + b_2) \\
\text{etc.}
\end{align*}
\]

In matrix notation

\[
z = Wx + b
\]

\[
a = f(z)
\]

where \( f \) is applied element-wise:

\[
f([z_1, z_2, z_3]) = [f(z_1), f(z_2), f(z_3)]
\]
Activation Function

logistic ("sigmoid")

\[ f(z) = \frac{1}{1 + \exp(-z)}. \]

\[ f'(z) = f(z)(1 - f(z)) \]

\[ \text{tanh} \]

\[ f(z) = \tanh(z) = \frac{e^z - e^{-z}}{e^z + e^{-z}}, \]

\[ f'(z) = 1 - f(z)^2 \]

\[ \text{tanh is just a rescaled and shifted sigmoid} \]

\[ \tanh(z) = 2 \text{logistic}(2z) - 1 \]
### Activation Function

<table>
<thead>
<tr>
<th>Function</th>
<th>Formula</th>
</tr>
</thead>
<tbody>
<tr>
<td>hard tanh</td>
<td>$\text{HardTanh}(x) = \begin{cases} -1 &amp; \text{if } x &lt; -1 \ x &amp; \text{if } -1 \leq x \leq 1 \ 1 &amp; \text{if } x &gt; 1 \end{cases}$</td>
</tr>
<tr>
<td>soft sign</td>
<td>$\text{softsign}(z) = \frac{a}{1 +</td>
</tr>
<tr>
<td>rectified linear</td>
<td>$\text{rect}(z) = \max(z, 0)$</td>
</tr>
</tbody>
</table>

- hard tanh similar but computationally cheaper than tanh and saturates hard.
- Glorot and Bengio, *AISTATS 2011* discuss softsign and rectifier
Non-linearity

- Logistic (Softmax) Regression only gives linear decision boundaries
Non-linearity

- Neural networks can learn much more complex functions and nonlinear decision boundaries!
Non-linearity

\[ y = g(Wx + b) \]
\[ z = g(Vg(Wx + b) + c) \]

output of first layer

With no nonlinearity:
\[ z = VWx + Vb + c \]
Equivalent to \( z = Ux + d \)
Non-linearity

Nodes in the hidden layer can learn interactions or conjunctions of features

\[ y = -2x_1 - x_2 + 2 \tanh(x_1 + x_2) \]
What about Word2vec (Skip-gram and CBOW)?
So, what about Word2vec (Skip-gram and CBOW)?

It is not deep learning — but “shallow” neural networks. It is — in fact — a log-linear model (softmax regression). So, it is faster over larger dataset yielding better embeddings.
Learning Neural Networks

Computing these looks like running this network in reverse (backpropagation)
Strategy for Successful NNs

• Select network structure appropriate for problem
  - Structure: Single words, fixed windows, sentence based, document level; bag of words, recursive vs. recurrent, CNN, …
  - Nonlinearity

• Check for implementation bugs with gradient checks
• Parameter initialization
• Optimization tricks

• Check if the model is powerful enough to overfit
  - If not, change model structure or make model “larger”
  - If you can overfit: regularize
Neural Machine Translation
Recurrent Neural Network (RNN)
Recurrent Neural Network (RNN)
Long Short-Term Memory Networks (LSTM)  
(Hochreiter & Schmidhuber, 1997)
Long Short-Term Memory Networks (LSTM)

\[
\begin{align*}
    z_t &= \sigma \left( W_z \cdot [h_{t-1}, x_t] \right) \\
    r_t &= \sigma \left( W_r \cdot [h_{t-1}, x_t] \right) \\
    \tilde{h}_t &= \tanh \left( W \cdot [r_t \ast h_{t-1}, x_t] \right) \\
    h_t &= (1 - z_t) \ast h_{t-1} + z_t \ast \tilde{h}_t
\end{align*}
\]

Source: Colah's Blog
Gated Recurrent Unit (GRU) (Cho et al., 2014)

\[
\begin{align*}
    z_t &= \sigma (W_z \cdot [h_{t-1}, x_t]) \\
    r_t &= \sigma (W_r \cdot [h_{t-1}, x_t]) \\
    \tilde{h}_t &= \tanh (W \cdot [r_t \ast h_{t-1}, x_t]) \\
    h_t &= (1 - z_t) \ast h_{t-1} + z_t \ast \tilde{h}_t
\end{align*}
\]
Twitter Conversation Data

Tesco Mobile @tescomobile
@RiccardoEspaa7 She's clearly going through a rough time in life.
Details

Riccardo Esposito @RiccardoEspaa7
@tescomobile I know either that or shit is going insane!
There is nothing wrong with tesco mobile
Details

Tesco Mobile @tescomobile
@RiccardoEspaa7 Together Riccardo, we'll make this world a better place.
Details

Riccardo Esposito @RiccardoEspaa7
@tescomobile me and you against the world
Details

Tesco Mobile @tescomobile
@RiccardoEspaa7 Yeah baby.
Details

Riccardo Esposito @RiccardoEspaa7
@tescomobile no one can stop us
Details
Neural Conversation

Source: Google’s blog
Neural Conversation

modeling speakers

Source: A Persona-Based Neural Conversation Model, Li et al. (ACL 2016)
Neural Network Toolkits

- **PyTorch**: [http://pytorch.org/](http://pytorch.org/)
  - Facebook AI Research and many others

- **Tensorflow**: [https://www.tensorflow.org/](https://www.tensorflow.org/)
  - By Google, actively maintained, bindings for many languages

- **DyNet**: [https://github.com/clab/dynet](https://github.com/clab/dynet)
  - CMU and other individual researchers, dynamic structures that change for every training instance

- **Caffe**: [http://caffe.berkeleyvision.org/](http://caffe.berkeleyvision.org/)
  - UC Berkeley, for vision

- **Theano**: [http://deeplearning.net/software/theano](http://deeplearning.net/software/theano)
  - University of Montreal, less and less maintained
Happy Turkey Day!

Source: http://www.butterball.com/how-tos/stuffing-vs-dressing