Deep Learning for NLP
(without Magic)

Richard Socher and Christopher Manning
Stanford University
NAACL 2013, Atlanta
http://nlp.stanford.edu/courses/NAACL2013/

*with a big thank you to Yoshua Bengio, with whom we participated in the previous ACL 2012 version of this tutorial
Part 1.2: The Basics

From logistic regression to neural nets
Demystifying neural networks

Neural networks come with their own terminological baggage

... just like SVMs

But if you understand how logistic regression or maxent models work

Then you already understand the operation of a basic neural network neuron!

A single neuron
A computational unit with $n$ (3) inputs and 1 output and parameters $W, b$
This is exactly what a neuron computes

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

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

\( b \): 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 ...

But we don’t have to decide ahead of time what variables these logistic regressions are trying to predict!
A neural network = running several logistic regressions at the same time

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

It is the training criterion 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....
How do we train the weights $W$?

- For a single supervised layer, we train just like a maxent model – we calculate and use error derivatives (gradients) to improve
  - Online learning: Stochastic gradient descent (SGD)
    - Or improved versions like AdaGrad (Duchi, Hazan, & Singer 2010)
  - Batch learning: Conjugate gradient or L-BFGS

- A multilayer net could be more complex because the internal ("hidden") logistic units make the function non-convex ... just as for hidden CRFs [Quattoni et al. 2005, Gunawardana et al. 2005]
  - But we can use the same ideas and techniques
    - Just without guarantees ...
  - We “backpropagate” error derivatives through the model
Non-linearities: Why they’re needed

• For logistic regression: map to probabilities
• Here: function approximation, e.g., regression or classification
  • Without non-linearities, deep neural networks can’t do anything more than a linear transform
    • Extra layers could just be compiled down into a single linear transform
  • Probabilistic interpretation unnecessary except in the Boltzmann machine/graphical models
    • People often use other non-linearities, such as \texttt{tanh}, as we’ll discuss in part 3
Summary
Knowing the meaning of words!

You now understand the basics and the relation to other models

- Neuron = logistic regression or similar function
- Input layer = input training/test vector
- Bias unit = intercept term/always on feature
- Activation = response
- Activation function is a logistic (or similar “sigmoid” nonlinearity)
- Backpropagation = running stochastic gradient descent backward layer-by-layer in a multilayer network
- Weight decay = regularization / Bayesian prior