Neural Net Basics
Neural Networks

Linear model:  \( y = \mathbf{w} \cdot \mathbf{x} + b \)

\[
\begin{align*}
  y &= g(\mathbf{w} \cdot \mathbf{x} + b) \\
  y &= g(\mathbf{Wx} + \mathbf{b})
\end{align*}
\]

Nonlinear transformation \quad Warp \quad Shift

\[
\text{pred} = \mathbf{w'}^\top y
\]

Neural Networks

Linear classifier

Neural network

Linear classification in the transformed space!

Taken from http://colah.github.io/posts/2014-03-NN-Manifolds-Topology/
Deep Neural Networks

y = \text{g}(Wx + b)

z = \text{g}(Vy + c)

z = \text{g}(V\text{g}(Wx + b) + c)

"Feedforward" computation (not recurrent)
Deep Neural Networks

Taken from http://colah.github.io/posts/2014-03-NN-Manifolds-Topology/
Feedforward Networks, Backpropagation
Vectorization and Softmax

\[ P(y|x) = \frac{\exp(w_y^\top x)}{\sum_{y'} \exp(w_{y'}^\top x)} \]

- Single scalar probability

- Three classes, “different weights”
  \[ w_1^\top x = -1.1 \quad 0.036 \]
  \[ w_2^\top x = 2.1 \quad 0.89 \quad \text{class probs} \]
  \[ w_3^\top x = -0.4 \quad 0.07 \]

- Softmax operation = “exponentiate and normalize”

- We write this as: softmax(\(Wx\))
Logistic Regression with NNs

\[ P(y|x) = \frac{\exp(w_y^T x)}{\sum_{y'} \exp(w_{y'}^T x)} \]  \( \triangleright \) Single scalar probability

\[ P(y|x) = \text{softmax}(W f(x)) \]  \( \triangleright \) Weight vector per class; 
\[ W \] is [num classes x num feats]

\[ P(y|x) = \text{softmax}(W g(V f(x))) \]  \( \triangleright \) Now one hidden layer
$P(y|x) = \text{softmax}(Wg(Vf(x)))$
Training Neural Networks

\[ P(y|x) = \text{softmax}(Wz) \quad z = g(Vf(x)) \]

- Maximize log likelihood of training data

\[ \mathcal{L}(x, i^*) = \log P(y = i^*|x) = \log (\text{softmax}(Wz) \cdot e_{i^*}) \]

- \( i^* \): index of the gold label

- \( e_i \): 1 in the \( i \)th row, zero elsewhere. Dot by this = select \( i \)th index

\[ \mathcal{L}(x, i^*) = Wz \cdot e_{i^*} - \log \sum_j \exp(Wz) \cdot e_j \]
Computing Gradients

\[ \mathcal{L}(\mathbf{x}, i^*) = W \mathbf{z} \cdot e_{i^*} - \log \sum_j \exp(W \mathbf{z}) \cdot e_j \]

- Gradient with respect to \( W \)

\[
\frac{\partial}{\partial W_{ij}} \mathcal{L}(\mathbf{x}, i^*) = \begin{cases} 
  z_j - P(y = i | \mathbf{x})z_j & \text{if } i = i^* \\
  -P(y = i | \mathbf{x})z_j & \text{otherwise}
\end{cases}
\]

- Looks like logistic regression with \( \mathbf{z} \) as the features!
Neural Networks for Classification

\[ P(y|x) = \text{softmax}(Wg(Vf(x))) \]
Backpropagation

- Gradients of output weights $W$ are easy to compute — looks like logistic regression with hidden layer $z$ as feature vector.

- Use the chain rule from calculus to compute an update for $V$. Looks like running the network in reverse.

- Need to remember the values from the forward computation.

- Autodiff tools mean you never need to implement this!
Neural Nets History
History: NN “dark ages”

- Convnets: applied to MNIST by LeCun in 1998

- LSTMs: Hochreiter and Schmidhuber (1997)

2008-2013: A glimmer of light...

- Collobert and Weston 2011: “NLP (almost) from scratch”
  - Feedforward neural nets induce features for sequential CRFs (“neural CRF”)
  - 2008 version was marred by bad experiments, claimed SOTA but wasn’t, 2011 version tied SOTA

- Krizhevsky et al. (2012): AlexNet for vision

- Socher 2011-2014: tree-structured RNNs working okay
2014: Stuff starts working


- Sutskever et al. + Bahdanau et al.: seq2seq for neural MT (LSTMs work for NLP?)

- Chen and Manning transition-based dependency parser (even feedforward networks work well for NLP?)

- 2015: explosion of neural nets for everything under the sun
Why didn’t they work before?

- **Datasets too small**: for MT, not really better until you have 1M+ parallel sentences (and really need a lot more)

- **Optimization not well understood**: good initialization, per-feature scaling + momentum (Adagrad / Adadelta / Adam) work best out-of-the-box
  - **Regularization**: dropout is pretty helpful
  - **Computers not big enough**: can’t run for enough iterations

- **Inputs**: need word representations to have the right continuous semantics
More implementation details: practical training techniques

Word representations / word vectors

word2vec, GloVe