Neural Networks

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

\( \text{pred} = w^T y \)

Neural Network Basics

Neural Networks

Linear classifier

Neural network

Linear classification in the transformed space!

Deep Neural Networks

\[
\begin{align*}
  y &= g(Wx + b) \\
  z &= g(Vy + c) \\
  z &= g(Vg(Wx + b) + c)
\end{align*}
\]

“Feedforward” computation (not recurrent)

Output of first layer

Adopted from Chris Dyer
Deep Neural Networks

Feedforward Networks, Backpropagation

Vectorization and Softmax

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

- Three classes, “different weights”
  - \( w_1^\top x = -1.1 \to 0.036 \)
  - \( w_2^\top x = 2.1 \to 0.89 \) class probs
  - \( w_3^\top x = -0.4 \to 0.07 \)

- Softmax operation = “exponentiate and normalize”
- We write this as: \( \text{softmax}(Wx) \)

Logistic Regression with NNs

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

\[ P(y|x) = \text{softmax}(Wf(x)) \]  
\[ P(y|x) = \text{softmax}(W_g(Vf(x))) \]

- Weight vector per class; \( W \) is [num classes x num feats]
- Now one hidden layer
**Neural Networks for Classification**

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

- **Input:** \( x \) (\( n \) features)
- **Output:** \( y \) (\( num\_classes \) probs)
- **Hidden Units:** \( d \) hidden units
- **Nonlinearity:** \( g \) (tanh, relu, ...)
- **Matrix:** \( V \) (\( d \times n \) matrix), \( W \) (\( num\_classes \times d \) matrix)

**Softmax Function**

\[
z = g(Vf(x))
\]

\[
P(y|x) = \text{softmax}(Wz)
\]

**Training Neural Networks**

- Maximize log likelihood of training data
- \( P(y|x) = \text{softmax}(Wz) \) \( z = g(Vf(x)) \)
- \( i^* \): index of the gold label
- \( e_i \): 1 in the \( i \)th row, zero elsewhere. Dot by this = select \( i \)th index

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

**Computing Gradients**

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

- Gradient with respect to \( W \)

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

- Looks like logistic regression with \( z \) 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

Next Time

- More implementation details: practical training techniques
- Word representations / word vectors
- word2vec, GloVe