Intro Neural Networks

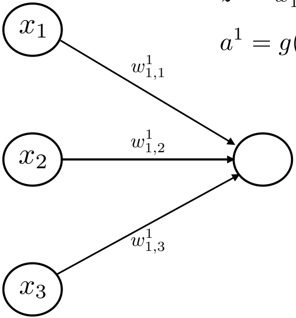
Matt Olson

September 8, 2017

Overview

- Q: What is a Deep Neural Network?
  - A bunch of logistic regressions jammed together
- Training has recently become feasible: Hinton (2006), GPUs
- We will cover: notation, back prop, examples
- Outline / discussion study group organization

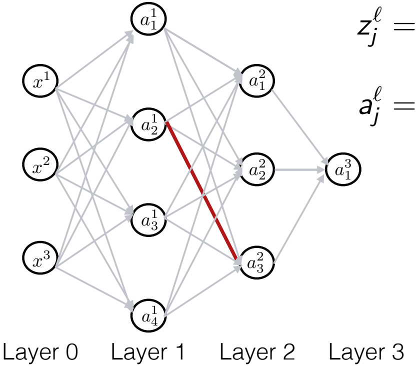
The Perceptron (Logistic Regression)



$$z^1 = w^1_{1,1}x_1 + w^1_{1,2}x_2 + w^1_{1,3}x_3$$

$$a^1 = g(z^1)$$

Larger Networks

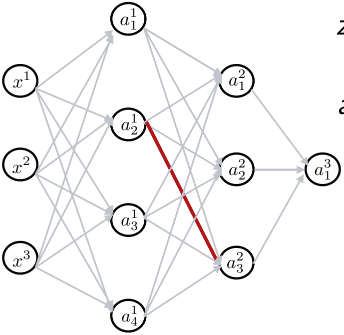


$$z_j^\ell = \sum_k w_{j,k}^l a_k^{l-1} \quad \text{for} \quad l = 1, \ldots, 3$$

$$a_j^\ell = g\left(z_j^\ell\right)$$

- $w_{j,k}^\ell$ connects node k in layer $\ell - 1$ to node j in layer ℓ.
- The red edge is $w_{3,2}^2$.

Larger Networks Ctd.



$$z_j^\ell = \sum_k w_{j,k}^l a_k^{l-1} \quad \text{for } l = 1, \ldots, 3$$

$$a_j^\ell = g\left(z_j^\ell\right)$$

- (vectorized) $z^\ell = W_\ell a^\ell$ and $a^\ell = g\left(z^\ell\right)$
- Prediction at a point x: $\widehat{y}(x) = g\left(W_3 g\left(W_2 g\left(W_1 x\right)\right)\right)$

The Learning Problem

Observe $\{(x_1, y_1), \ldots, (x_n, y_n)\}$ with $y_i \in \{0, 1\}$. Fix a network architecture with L hidden layers. Find weights $W_1, \ldots, W_L$ that minimize

$$\underset{W_1,\ldots,W_L}{\text{minimize}} \quad 1/n \sum_{i=1}^{n} \mathcal{L}\left(y_i, \widehat{y}\left(x_i\right)\right)$$

where $\mathcal{L}$ is a loss function (say, cross-entropy) and

$$\widehat{y}\left(x_i\right) = g\left(W_L g\left(W_{L-1} g\left(\cdots g\left(W_1 x\right)\right)\right)\right)$$

- This minimization problem is attacked through gradient descent
- Need to compute $\nabla_W \mathcal{L}\left(y_i, \widehat{y}\left(x_i\right)\right)$

The Chain Rule

Recall that for functions $h_1, \ldots, h_L$ (of conformable dimensions)

$$
\begin{aligned}
f &= h_L \circ h_{L-1} \circ \cdots \circ h_1 \\
Df &= Dh_L \circ Dh_{L-1} \circ \cdots \circ Dh_1 \\
\nabla f &= (Dh_1)^T \circ (Dh_2)^T \circ \cdots \circ (Dh_L)^T
\end{aligned}
$$

Gradient Calculations

For simplicity, let's return to the network with two hidden layers from a previous slide and take $\mathcal{L}(y, \theta) = 1/2\,(y - \theta)^2$.

- Feed x_i through the network to get activations a^1, a^2, a^3 and prediction $\widehat{y}(x_i)$
- Form $G(\ell) = \text{diag}\left([g'(z_1^\ell), \ldots, g'(z_{L_\ell}^\ell)]\right)$ for $\ell = 1, 2, 3$
- "Trick": $\frac{\partial \mathcal{L}}{\partial w_{j,k}^\ell} = \frac{\partial \mathcal{L}}{\partial z_j^\ell} \frac{\partial z_j^\ell}{\partial w_{j,k}^l}$
  - $\frac{\partial \mathcal{L}}{\partial z^3} = G(3)\,(y_i - \widehat{y}(x_i))$
  - $\frac{\partial \mathcal{L}}{\partial z^2} = G(2) W_3^T G(3)\,(y_i - \widehat{y}(x_i))$
  - $\frac{\partial \mathcal{L}}{\partial z^1} = G(1) W_2^T G(2) W_3^T G(3)\,(y_i - \widehat{y}(x_i))$

Backpropagation

An efficient algorithm for computing $\frac{\partial \mathcal{L}}{\partial w^l_{j,k}}$ at a point x. For a neural network with $L+1$ layers:

(1) Feed x through the network and compute $a^1, \ldots, a^L$ and prediction $\widehat{y}(x)$.

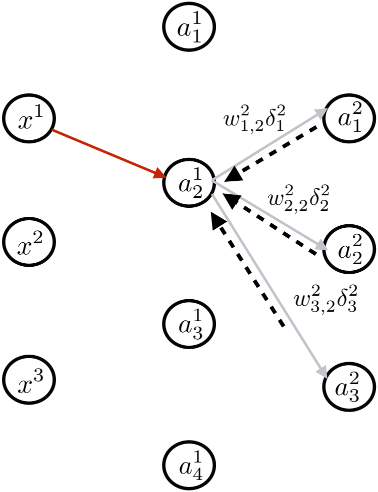
(2) $G(\ell) = \text{diag}\left([g'(z^\ell_1), \ldots, g'(z^\ell_{L_\ell})]\right)$ for $\ell = 1, \ldots, L$.

(3) $\delta^L = G(L)\,(y - \widehat{y}(x))$

(4) $\delta^\ell = G(\ell) W^T_{\ell+1} \delta^{\ell+1}$ for $\ell = 1, \ldots, L-1$.

(5) $\frac{\partial \mathcal{L}}{\partial w^\ell_{j,k}} = a^{\ell-1}_k \delta^\ell_j$

Zooming In



$$\delta_2^1 = (w_{1,2}^2\delta_1^2 + w_{2,2}^2\delta_2^2 + w_{3,2}^2\delta_3^2)g'(z_2^1)$$

$$\frac{\partial \mathcal{L}}{\partial w_{2,1}^1} = \delta_2^1 x^1$$