



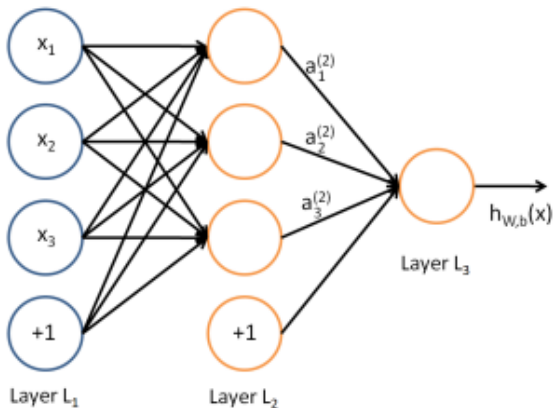
# Multilayer Networks

Machine Learning: Jordan Boyd-Graber

University of Maryland

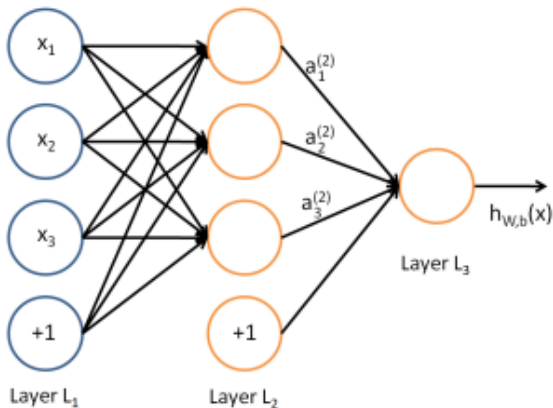
MATHEMATICAL DESCRIPTION

## Learn the features and the function



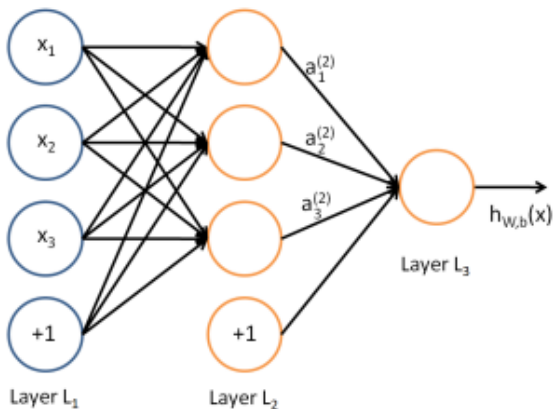
$$a_1^{(2)} = f\left(W_{11}^{(1)} x_1 + W_{12}^{(1)} x_2 + W_{13}^{(1)} x_3 + b_1^{(1)}\right)$$

## Learn the features and the function



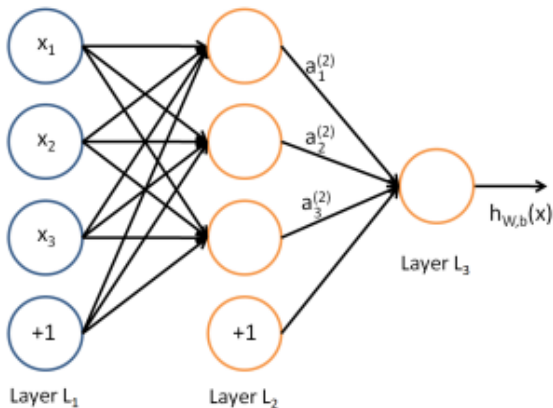
$$a_2^{(2)} = f\left(W_{21}^{(1)}x_1 + W_{22}^{(1)}x_2 + W_{23}^{(1)}x_3 + b_2^{(1)}\right)$$

## Learn the features and the function



$$a_3^{(2)} = f\left(w_{31}^{(1)}x_1 + w_{32}^{(1)}x_2 + w_{33}^{(1)}x_3 + b_3^{(1)}\right)$$

## Learn the features and the function



$$h_{W,b}(x) = a_1^{(3)} = f\left(W_{11}^{(2)} a_1^{(2)} + W_{12}^{(2)} a_2^{(2)} + W_{13}^{(2)} a_3^{(2)} + b_1^{(2)}\right)$$

## Objective Function

- For every example  $x, y$  of our supervised training set, we want the label  $y$  to match the prediction  $h_{W,b}(x)$ .

$$J(W, b; x, y) \equiv \frac{1}{2} \|h_{W,b}(x) - y\|^2 \quad (1)$$

## Objective Function

- For every example  $x, y$  of our supervised training set, we want the label  $y$  to match the prediction  $h_{W,b}(x)$ .

$$J(W, b; x, y) \equiv \frac{1}{2} \|h_{W,b}(x) - y\|^2 \quad (1)$$

- We want this value, summed over all of the examples to be as small as possible

## Objective Function

- For every example  $x, y$  of our supervised training set, we want the label  $y$  to match the prediction  $h_{W,b}(x)$ .

$$J(W, b; x, y) \equiv \frac{1}{2} \|h_{W,b}(x) - y\|^2 \quad (1)$$

- We want this value, summed over all of the examples to be as small as possible
- We also want the weights not to be too large

$$\frac{\lambda}{2} \sum_l^{n_l-1} \sum_{i=1}^{s_l} \sum_{j=1}^{s_{l+1}} (w_{ji}^l)^2 \quad (2)$$



## Objective Function

- For every example  $x, y$  of our supervised training set, we want the label  $y$  to match the prediction  $h_{W,b}(x)$ .

$$J(W, b; x, y) \equiv \frac{1}{2} \|h_{W,b}(x) - y\|^2 \quad (1)$$

- We want this value, summed over all of the examples to be as small as possible
- We also want the weights not to be too large

$$\frac{\lambda}{2} \sum_l^{n_l-1} \sum_{i=1}^{s_l} \sum_{j=1}^{s_{l+1}} (w_{ji}^l)^2 \quad (2)$$

## Objective Function

- For every example  $x, y$  of our supervised training set, we want the label  $y$  to match the prediction  $h_{W,b}(x)$ .

$$J(W, b; x, y) \equiv \frac{1}{2} \|h_{W,b}(x) - y\|^2 \quad (1)$$

- We want this value, summed over all of the examples to be as small as possible
- We also want the weights not to be too large

$$\frac{\lambda}{2} \sum_l^{n_l-1} \sum_{i=1}^{s_l} \sum_{j=1}^{s_{l+1}} (w_{ji}^l)^2 \quad (2)$$

Sum over all layers

## Objective Function

- For every example  $x, y$  of our supervised training set, we want the label  $y$  to match the prediction  $h_{W,b}(x)$ .

$$J(W, b; x, y) \equiv \frac{1}{2} \|h_{W,b}(x) - y\|^2 \quad (1)$$

- We want this value, summed over all of the examples to be as small as possible
- We also want the weights not to be too large

$$\frac{\lambda}{2} \sum_l^{n_l-1} \sum_{i=1}^{s_l} \sum_{j=1}^{s_{l+1}} (w_{ji}^l)^2 \quad (2)$$

Sum over all sources

## Objective Function

- For every example  $x, y$  of our supervised training set, we want the label  $y$  to match the prediction  $h_{W,b}(x)$ .

$$J(W, b; x, y) \equiv \frac{1}{2} \|h_{W,b}(x) - y\|^2 \quad (1)$$

- We want this value, summed over all of the examples to be as small as possible
- We also want the weights not to be too large

$$\frac{\lambda}{2} \sum_l^{n_l-1} \sum_{i=1}^{s_l} \sum_{j=1}^{s_{l+1}} (w_{ji}^l)^2 \quad (2)$$

Sum over all destinations

## Objective Function

Putting it all together:

$$J(W, b) = \left[ \frac{1}{m} \sum_{i=1}^m \frac{1}{2} \|h_{W,b}(x^{(i)}) - y^{(i)}\|^2 \right] + \frac{\lambda}{2} \sum_l^{n_l-1} \sum_{i=1}^{s_l} \sum_{j=1}^{s_{l+1}} (w_{ji}^l)^2 \quad (3)$$

## Objective Function

Putting it all together:

$$J(W, b) = \left[ \frac{1}{m} \sum_{i=1}^m \frac{1}{2} \|h_{W,b}(x^{(i)}) - y^{(i)}\|^2 \right] + \frac{\lambda}{2} \sum_l^{n_l-1} \sum_{i=1}^{s_l} \sum_{j=1}^{s_{l+1}} (W_{ji}^l)^2 \quad (3)$$

- Our goal is to minimize  $J(W, b)$  as a function of  $W$  and  $b$

## Objective Function

Putting it all together:

$$J(W, b) = \left[ \frac{1}{m} \sum_{i=1}^m \frac{1}{2} \|h_{W,b}(x^{(i)}) - y^{(i)}\|^2 \right] + \frac{\lambda}{2} \sum_l^{n_l-1} \sum_{i=1}^{s_l} \sum_{j=1}^{s_{l+1}} (w_{ji}^l)^2 \quad (3)$$

- Our goal is to minimize  $J(W, b)$  as a function of  $W$  and  $b$
- Initialize  $W$  and  $b$  to small random value near zero

## Objective Function

Putting it all together:

$$J(W, b) = \left[ \frac{1}{m} \sum_{i=1}^m \frac{1}{2} \|h_{W,b}(x^{(i)}) - y^{(i)}\|^2 \right] + \frac{\lambda}{2} \sum_l^{n_l-1} \sum_{i=1}^{s_l} \sum_{j=1}^{s_{l+1}} (w_{ji}^l)^2 \quad (3)$$

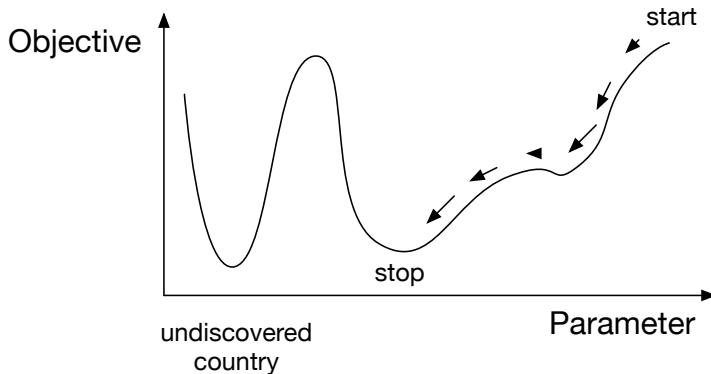
- Our goal is to minimize  $J(W, b)$  as a function of  $W$  and  $b$
- Initialize  $W$  and  $b$  to small random value near zero
- Adjust parameters to optimize  $J$



## Gradient Descent

### Goal

Optimize  $J$  with respect to variables  $W$  and  $b$



## Backpropigation

- For convenience, write the input to sigmoid

$$z_i^{(l)} = \sum_{j=1}^n w_{ij}^{(l-1)} x_j + b_i^{(l-1)} \quad (4)$$

## Backpropigation

- For convenience, write the input to sigmoid

$$z_i^{(l)} = \sum_{j=1}^n w_{ij}^{(l-1)} x_j + b_i^{(l-1)} \quad (4)$$

- The gradient is a function of a node's error  $\delta_i^{(l)}$

## Backpropagation

- For convenience, write the input to sigmoid

$$z_i^{(l)} = \sum_{j=1}^n w_{ij}^{(l-1)} x_j + b_i^{(l-1)} \quad (4)$$

- The gradient is a function of a node's error  $\delta_i^{(l)}$
- For output nodes, the error is obvious:

$$\delta_i^{(n_l)} = \frac{\partial}{\partial z_i^{(n_l)}} \|y - h_{w,b}(x)\|^2 = -(y_i - a_i^{(n_l)}) \cdot f'(z_i^{(n_l)}) \frac{1}{2} \quad (5)$$

## Backpropagation

- For convenience, write the input to sigmoid

$$z_i^{(l)} = \sum_{j=1}^n w_{ij}^{(l-1)} x_j + b_i^{(l-1)} \quad (4)$$

- The gradient is a function of a node's error  $\delta_i^{(l)}$
- For output nodes, the error is obvious:

$$\delta_i^{(n_l)} = \frac{\partial}{\partial z_i^{(n_l)}} \|y - h_{w,b}(x)\|^2 = -(y_i - a_i^{(n_l)}) \cdot f'(z_i^{(n_l)}) \frac{1}{2} \quad (5)$$

- Other nodes must “backpropagate” **downstream error** based on connection strength

$$\delta_i^{(l)} = \left( \sum_{j=1}^{s_{l+1}} w_{ji}^{(l+1)} \delta_j^{(l+1)} \right) f'(z_i^{(l)}) \quad (6)$$

## Backpropagation

- For convenience, write the input to sigmoid

$$z_i^{(l)} = \sum_{j=1}^n w_{ij}^{(l-1)} x_j + b_i^{(l-1)} \quad (4)$$

- The gradient is a function of a node's error  $\delta_i^{(l)}$
- For output nodes, the error is obvious:

$$\delta_i^{(n_l)} = \frac{\partial}{\partial z_i^{(n_l)}} \|y - h_{w,b}(x)\|^2 = -(y_i - a_i^{(n_l)}) \cdot f'(z_i^{(n_l)}) \frac{1}{2} \quad (5)$$

- Other nodes must “backpropagate” downstream error based on **connection strength**

$$\delta_i^{(l)} = \left( \sum_{j=1}^{s_{l+1}} w_{ji}^{(l+1)} \delta_j^{(l+1)} \right) f'(z_i^{(l)}) \quad (6)$$

## Backpropagation

- For convenience, write the input to sigmoid

$$z_i^{(l)} = \sum_{j=1}^n w_{ij}^{(l-1)} x_j + b_i^{(l-1)} \quad (4)$$

- The gradient is a function of a node's error  $\delta_i^{(l)}$
- For output nodes, the error is obvious:

$$\delta_i^{(n_l)} = \frac{\partial}{\partial z_i^{(n_l)}} \|y - h_{w,b}(x)\|^2 = -(y_i - a_i^{(n_l)}) \cdot f'(z_i^{(n_l)}) \frac{1}{2} \quad (5)$$

- Other nodes must “backpropagate” downstream error based on connection strength

$$\delta_i^{(l)} = \left( \sum_{j=1}^{s_{l+1}} w_{ji}^{(l+1)} \delta_j^{(l+1)} \right) f'(z_i^{(l)}) \quad (6)$$

(chain rule)

## Partial Derivatives

- For weights, the partial derivatives are

$$\frac{\partial}{\partial W_{ij}^{(l)}} J(W, b; x, y) = a_j^{(l)} \delta_i^{(l+1)} \quad (7)$$

- For the bias terms, the partial derivatives are

$$\frac{\partial}{\partial b_i^{(l)}} J(W, b; x, y) = \delta_i^{(l+1)} \quad (8)$$

- But this is just for a single example ...



## Full Gradient Descent Algorithm

1. Initialize  $U^{(l)}$  and  $V^{(l)}$  as zero
2. For each example  $i = 1 \dots m$ 
  - 2.1 Use backpropagation to compute  $\nabla_W J$  and  $\nabla_b J$
  - 2.2 Update weight shifts  $U^{(l)} = U^{(l)} + \nabla_{W^{(l)}} J(W, b; x, y)$
  - 2.3 Update bias shifts  $V^{(l)} = V^{(l)} + \nabla_{b^{(l)}} J(W, b; x, y)$
3. Update the parameters

$$W^{(l)} = W^{(l)} - \alpha \left[ \left( \frac{1}{m} U^{(l)} \right) \right] \quad (9)$$

$$b^{(l)} = b^{(l)} - \alpha \left[ \frac{1}{m} V^{(l)} \right] \quad (10)$$

4. Repeat until weights stop changing