Deep Learning

Advanced Machine Learning for NLP
Jordan Boyd-Graber

MATHEMATICAL DESCRIPTION
Learn the features and the function

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

\[ a^{(2)}_3 = f \left( W^{(1)}_{31} x_1 + W^{(1)}_{32} x_2 + W^{(1)}_{33} x_3 + b^{(1)}_3 \right) \]
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 $h_{W,b}(x)$.

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

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

$$\frac{\lambda}{2} \sum_l^{n_l-1} \sum_{i=1}^{s_l} \sum_{j=1}^{s_{l+1}} (W_{ji})^2$$ \hspace{1cm} (2)
Objective Function

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

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

- 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.

\[
\frac{\lambda}{2} \sum_{l}^{n_l-1} \sum_{s_i}^{s_i+1} \sum_{s_{i+1}}^{s_{i+1}} (W_{ji})^2 \tag{2}
\]
Objective Function

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

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

(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

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

(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 \( h_{W,b}(x) \).

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

(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

\[
\frac{\lambda}{2} \sum_{l}^{n_{l} - 1} \sum_{i=1}^{s_{l}} \sum_{j=1}^{s_{l+1}} (W'_{ji})^2
\]  

(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 $h_{W,b}(x)$.

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

$$\frac{\lambda}{2} \sum_l^{n_l - 1} \sum_i^{s_l} \sum_{j=1}^{s_{l+1}} (W_{ji})^2$$  \hspace{1cm} (2)

Sum over all destinations
Objective Function

Putting it all together:

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

(3)
Putting it all together:

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

(3)

- Our goal is to minimize $J(W, b)$ as a function of $W$ and $b$
Putting it all together:

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

(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:

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

- 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 \( J \)
Outline
Gradient Descent

Goal

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

- For convenience, write the input to sigmoid

\[ z_i^{(l)} = \sum_{j=1}^{n} W_{ij}^{(l-1)} x_j + b_i^{(l-1)} \]  (4)
Backpropagation

• For convenience, write the input to sigmoid

\[ z_i^{(l)} = \sum_{j=1}^{n} W_{ij}^{(l-1)} x_j + b_i^{(l-1)} \]  

(4)

• The gradient is a function of a node’s error \( \delta_i^{(l)} \)
Backpropagation

- For convenience, write the input to sigmoid

\[ z^{(l)}_i = \sum_{j=1}^{n} W^{(l-1)}_{ij} x_j + b^{(l-1)}_i \]  

(4)

- The gradient is a function of a node’s error \( \delta^{(l)}_i \)

- For output nodes, the error is obvious:

\[ \delta^{(n_l)}_i = \frac{\partial}{\partial z^{(n_l)}_i} \| y - h_{w,b}(x) \|^2 = -\left(y_i - a^{(n_l)}_i\right) \cdot f'(z^{(n_l)}_i) \frac{1}{2} \]  

(5)
Backpropigation

- For convenience, write the input to sigmoid
  \[ z_i^{(l)} = \sum_{j=1}^{n} W_{ij}^{(l-1)} x_j + b_i^{(l-1)} \]  
  \[ (4) \]

- The gradient is a function of a node’s error \( \delta_i^{(l)} \)
- For output nodes, the error is obvious:
  \[ \delta_i^{(n_l)} = \frac{\partial}{\partial z_i^{(n_l)}} \| y - h_{w,b}(x) \|^2 = -(y_i - a_i^{(n_l)}) \cdot f'(z_i^{(n_l)}) \cdot \frac{1}{2} \]  
  \[ (5) \]

- Other nodes must “backpropagate” downstream error based on connection strength
  \[ \delta_i^{(l)} = \left( \sum_{j=1}^{s_{l+1}} W_{ji}^{(l+1)} \delta_j^{(l+1)} \right) f'(z_i^{(l)}) \]  
  \[ (6) \]
Deep Learning from Data

Backpropogation

- For convenience, write the input to sigmoid

\[
z_i^{(l)} = \sum_{j=1}^{n} W_{ij}^{(l-1)} x_j + b_i^{(l-1)}
\]  

(4)

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

\[
\delta_i^{(n)} = \frac{\partial}{\partial z_i^{(n)}} \|y - h_{w,b}(x)\|^2 = -\left(y_i - a_i^{(n)}\right) \cdot f'(z_i^{(n)}) \frac{1}{2}
\]  

(5)

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

\[
\delta_i^{(l)} = \left(\sum_{j=1}^{s_{l+1}} W_{ji}^{(l+1)} \delta_j^{(l+1)}\right) f'(z_i^{(l)})
\]  

(6)
Backpropigation

- For convenience, write the input to sigmoid

\[ z_i^{(l)} = \sum_{j=1}^{n} W_{ij}^{(l-1)} x_j + b_i^{(l-1)} \]  

(4)

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

- For output nodes, the error is obvious:

\[ \delta_i^{(n_l)} = \frac{\partial}{\partial z_i^{(n_l)}} \| y - h_w,b(x) \|^2 = -\left( y_i - a_i^{(n_l)} \right) \cdot f'(z_i^{(n_l)}) \cdot \frac{1}{2} \]  

(5)

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

\[ \delta_i^{(l)} = \left( \sum_{j=1}^{s_{l+1}} W_{ji}^{(l+1)} \delta_j^{(l+1)} \right) f'(z_i^{(l)}) \]  

(6)

(chain rule)
Partial Derivatives

- For weights, the partial derivatives are

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

- For the bias terms, the partial derivatives are

$$\frac{\partial}{\partial b_i^{(l)}} J(W, b; x, y) = \delta_i^{(l+1)}$$  \hspace{1cm} (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 $i = 1 \ldots m$
   1. Use backpropagation to compute $\nabla_W J$ and $\nabla_b J$
   2. Update weight shifts $U^{(l)} = U^{(l)} + \nabla_W J(W, b; x, y)$
   3. Update bias shifts $V^{(l)} = V^{(l)} + \nabla_b J(W, b; x, y)$
3. Update the parameters

   \[
   W^{(l)} = W^{(l)} - \alpha \left[ \frac{1}{m} U^{(l)} \right] \quad (9)
   
   b^{(l)} = b^{(l)} - \alpha \left[ \frac{1}{m} V^{(l)} \right] \quad (10)
   \]
4. Repeat until weights stop changing