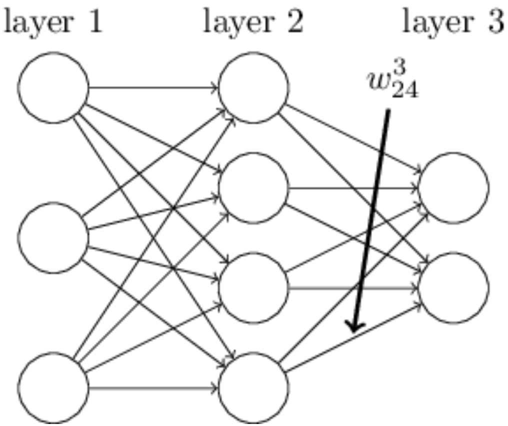
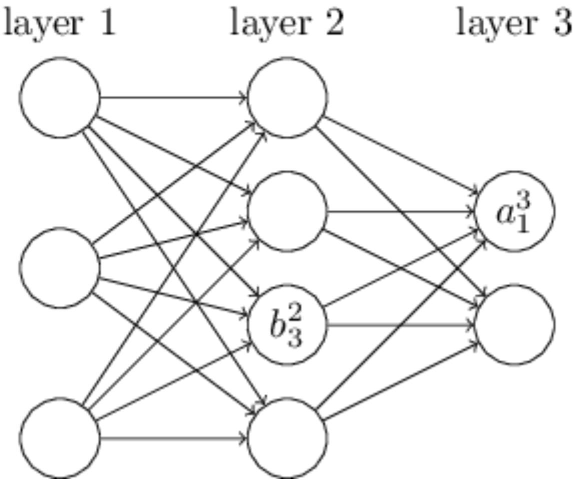
Backprop Chapter Summary

10/20/2019

Notation



w^l_{jk} is the weight from the k^{th} neuron in the $(l-1)^{\text{th}}$ layer to the j^{th} neuron in the l^{th} layer



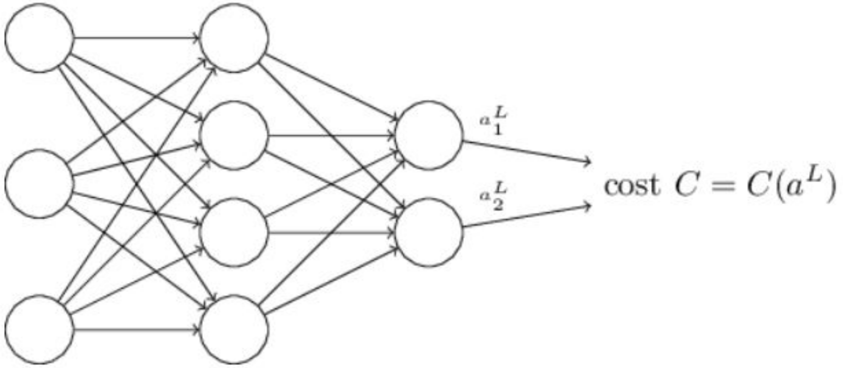
$$a^l_j = \sigma\left(\sum_k w^l_{jk} a^{l-1}_k + b^l_j\right)$$

$$a^l = \sigma(w^l a^{l-1} + b^l).$$

Goal of backprop

- Compute partial derivative of cost function with respect to any weight or bias in network
- Assumptions about cost function
  - Can be written as an average over individual training examples
  - Can be written as a function of outputs from NN

$$C = \frac{1}{n}\sum_x C_x$$



Example cost function

$$C = \frac{1}{2}\|y - a^L\|^2 = \frac{1}{2}\sum_j (y_j - a_j^L)^2$$

Error of neuron

$$\delta_j^l \equiv \frac{\partial C}{\partial z_j^l}.$$

- z_j^l represents the input to the activation of the jth neuron in the lth layer
- Larger error means changes in z_j^l will have larger impact
- To reduce cost, move z_j^l in the opposite direction of the error

Four fundamental equations behind backprop

- Equation for error of output layer
- Equation for error in terms of error of next layer
- Equation for rate of change of cost with respect to any bias in the network
- Equation for rate of change of cost with respect to any weight in the network

Equation for error of output layer

$$\delta_j^L = \frac{\partial C}{\partial a_j^L} \sigma'(z_j^L)$$

$$\delta^L = \nabla_a C \odot \sigma'(z^L)$$

Equation for error in terms of error of next layer

$$\delta^l = ((w^{l+1})^T \delta^{l+1}) \odot \sigma'(z^l)$$

Equation for rate of change of cost with respect to any bias in the network

$$\frac{\partial C}{\partial b_j^l} = \delta_j^l$$

Equation for rate of change of cost with respect to any weight in the network

$$\frac{\partial C}{\partial w^l_{jk}} = a^{l-1}_k \delta^l_j$$

Backprop Algorithm

1. **Input x:** Set the corresponding activation a^1 for the input layer.

2. **Feedforward:** For each $l = 2, 3, \ldots, L$ compute $z^l = w^l a^{l-1} + b^l$ and $a^l = \sigma(z^l)$.

3. **Output error δ^L:** Compute the vector $\delta^L = \nabla_a C \odot \sigma'(z^L)$.

4. **Backpropagate the error:** For each $l = L-1, L-2, \ldots, 2$ compute $\delta^l = ((w^{l+1})^T \delta^{l+1}) \odot \sigma'(z^l)$.

5. **Output:** The gradient of the cost function is given by $\frac{\partial C}{\partial w^l_{jk}} = a^{l-1}_k \delta^l_j$ and $\frac{\partial C}{\partial b^l_j} = \delta^l_j$.

Stochastic Gradient Descent

1. **Input a set of training examples**

2. **For each training example x:** Set the corresponding input activation $a^{x,1}$, and perform the following steps:

   - **Feedforward:** For each $l = 2, 3, \ldots, L$ compute $z^{x,l} = w^l a^{x,l-1} + b^l$ and $a^{x,l} = \sigma(z^{x,l})$.

   - **Output error $\delta^{x,L}$:** Compute the vector $\delta^{x,L} = \nabla_a C_x \odot \sigma'(z^{x,L})$.

   - **Backpropagate the error:** For each $l = L-1, L-2, \ldots, 2$ compute $\delta^{x,l} = ((w^{l+1})^T \delta^{x,l+1}) \odot \sigma'(z^{x,l})$.

3. **Gradient descent:** For each $l = L, L-1, \ldots, 2$ update the weights according to the rule $w^l \rightarrow w^l - \frac{\eta}{m} \sum_x \delta^{x,l} (a^{x,l-1})^T$, and the biases according to the rule $b^l \rightarrow b^l - \frac{\eta}{m} \sum_x \delta^{x,l}$.