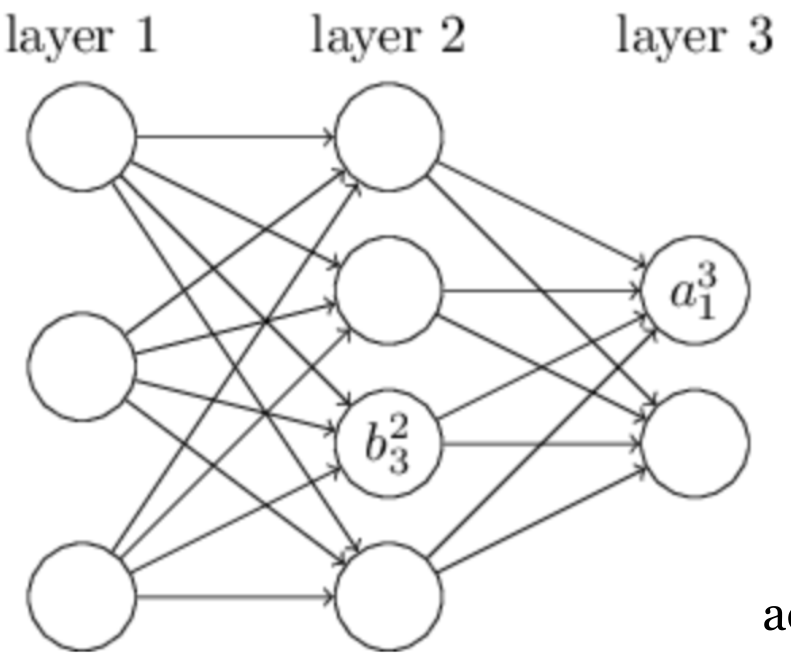
全连接



$$a_j^l = \sigma\left(\sum_k w_{jk}^l a_k^{l-1} + b_j^l\right)$$

activation vector

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

weight matrix

bias vector

how the activations in one layer relate to activations in the previous layer: we just apply the weight matrix to the activations, then add the bias vector, and finally apply the σ function.

反向传播是关于如何改变网络中的权重和偏置从而改变代价函数。 这意味着计算偏导数$\partial C / \partial w_{jk}^{l}$ 和 $\partial C / \partial b_{j}^{l}$ 。

为了计算这些，引入一个中间量 δ_{j}^{l} ，将其称为第l层第j个神经元中的误差(error)。或称为敏感度图（sensitivity map）。

反向传播将给我们一个计算误差 δ_{j}^{l} 的过程，然后将 δ_{j}^{l} 与 $\partial C / \partial w_{jk}^{l}$ 和 $\partial C / \partial b_{j}^{l}$ 联系起来。

weighted input

$$z_{j}^{l} = \sum_{k} w_{jk}^{l} a_{k}^{l-1} + b_{j}^{l}$$

error

$$\delta_{j}^{l} \equiv \frac{\partial C}{\partial z_{j}^{l}}$$

z_{j}^{l} the weighted input to the activation function for neuron j in layer l

δ^{l} vector of errors associated with layer l

An equation for the error in the output layer

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

desired output

$$C = \frac{1}{2}\sum_j (y_j - a_j^L)^2 \quad \Rightarrow \quad \partial C / \partial a_j^L = (a_j^L - y_j)$$

darknet: y - a

$$\delta_j^L = \frac{\partial C}{\partial a_j^L}\sigma'(z_j^L) \quad \Rightarrow \quad \delta^L = \nabla_a C \odot \sigma'(z^L)$$

Hadamard product

elementwise product of the two vectors $(s \odot t)_j = s_j t_j$

$$\begin{bmatrix} 1 \\ 2 \end{bmatrix} \odot \begin{bmatrix} 3 \\ 4 \end{bmatrix} = \begin{bmatrix} 1 * 3 \\ 2 * 4 \end{bmatrix} = \begin{bmatrix} 3 \\ 8 \end{bmatrix} \qquad \delta^L = (a^L - y) \odot \sigma'(z^L)$$

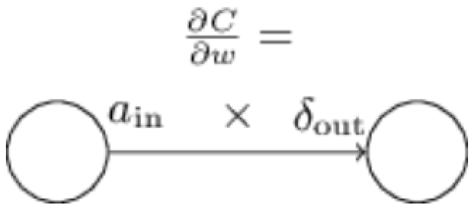
An equation for the rate of change of the cost with respect to any bias in the network:

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

An equation for the rate of change of the cost with respect to any weight in the network:

$$\frac{\partial C}{\partial w_{jk}^l} = a_k^{l-1}\delta_j^l$$

$$\frac{\partial C}{\partial w} = a_{\text{in}}\delta_{\text{out}}$$



推导过程可参考：http://neuralnetworksanddeeplearning.com/chap2.html

对于全连接层：

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

$$\delta^{l-1} = \left(\left(w^l\right)^T \delta^l\right) \odot \sigma'\left(z^{l-1}\right) \tag{FC-1}$$

$$\frac{\partial C}{\partial b_j^l} = \delta_j^l \tag{FC-2}$$

$$\frac{\partial C}{\partial w_{jk}^l} = a_k^{l-1} \delta_j^l \tag{FC-3}$$

$$b^l = b^l - \eta \frac{\partial C}{\partial b^l} \tag{FC-4}$$

$$w^l = w^l - \eta \frac{\partial C}{\partial w^l} \tag{FC-5}$$

In particular, given a mini-batch of m training examples, the following algorithm applies a gradient descent learning step based on that mini-batch:

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 $b^l \rightarrow b^l - \frac{\eta}{m} \sum_x \delta^{x,l}$.

YOLOv3当前层的参数的具体梯度计算过程

设当前层为第l-1层,那么计算其敏感度分两步:

1. 在l层的backward()函数的最后部分,会计算l-1层的

$$delta^{l-1} = \delta^{l} \frac{\partial z^{l}}{\partial a^{l-1}} \qquad \text{(P1)}$$

2. 在l-1层调用backward函数开头部分,再计算:

$$\delta^{l-1} = delta^{l-1} \odot \sigma^{'}\left(z^{l-1}\right) \qquad \text{(P2)}$$

参数更新

$$w = w - \alpha \nabla C \qquad \text{(update-1)}$$

引入动量的参数更新：

$$v_t = \gamma v_{t-1} \qquad \text{(update-2)}$$

$$w = w - v_t - \alpha \nabla C \qquad \text{(update-3)}$$

$$w = w - \frac{\lambda}{m} w \qquad \text{(update-4)}$$