Back Propagation

Mladen Kolar and Rob McCulloch

1. Back Propagation

1. Back Propagation

Backpropation is the basic algorithm for computing the gradient vector for a neural net model.

For a given (x, y) we need the partial derivatives of the ultimate loss with respect to all the weights and biases.

To evaluate the model we start at x and go *forward* through the layers, ending up at the ouput layer.

To evaluate the gradient we go *backward*, starting at the output layer and iterating back to the coefficients connecting x to the first hidden layer.

We will need a general notation for the neural net model.

Let's start by letting ℓ index the layers.

ℓ goes from 1 to L where $\ell = 1$ is the input layer (x) and L is the final output layer.

To keep things simple, we will have just one outcome with associated activation function g^L. For a single numeric outcome, g^L would typically be the identity function $I(x) = x$.

We will use the same activation function g at all the interior units (neurons).

Let p_ℓ be the number of neurons at layer ℓ.
Note that $p_1 = p$ where p is the dimension of x since that is the input layer.

Lots of Notation !!!!:

$Z_k^{(\ell)}$: the Z value at the k^{th} unit of layer (ℓ), $k = 1, 2, \ldots, p_\ell$.

We have $Z_{\text{unit}}^{(\text{layer})}$. Similary, we have $a_k^{(\ell)}$ with,

$a_k^{(\ell)} = g(Z_k^{(\ell)})$.

$w_{kj}^{(\ell)}$ = weight from $a_j^{(\ell)}$ (at layer ℓ) to $Z_k^{(\ell+1)}$ (at layer $(\ell + 1)$).

Think of w as $w_{kj}^{(\ell)} = w_{k \leftarrow j}^{(\ell)}$.

$b_k^{(\ell)}$ = intercept for $Z_k^{(\ell+1)}$ (at layer $(\ell + 1)$).

$$Z_k^{(\ell)} = b_k^{(\ell-1)} + \sum_{j=1}^{p_{(\ell-1)}} w_{kj}^{(\ell-1)} \, a_j^{(\ell-1)}, \quad k = 1, 2, \ldots, p_\ell.$$

$$Z_k^{(\ell)} = b_k^{(\ell-1)} + \sum_{j=1}^{p_{(\ell-1)}} w_{kj}^{(\ell-1)} a_j^{(\ell-1)}, \quad k = 1, 2, \ldots, p_\ell.$$

Matrix/Vector version:

$$Z^{(\ell)} = (Z_1^{(\ell)}, Z_2^{(\ell)}, \ldots, Z_{p_\ell}^{(\ell)})'$$

$$a^{(\ell)} = g(Z^{(\ell)})$$

$$b^{(\ell)} = (b_1^{(\ell)}, b_2^{(\ell)}, \ldots, b_{p_{(\ell+1)}}^{(\ell)})'$$

$$W^{(\ell)} = \left[w_{kj}^{(\ell)} \right], \quad p_{(\ell+1)} \times p_\ell$$

Then,

$$Z^{(\ell)} = b^{(\ell-1)} + W^{(\ell-1)} a^{(\ell-1)}$$

Begin:

$$a^{(1)} = x, \in R^p.$$

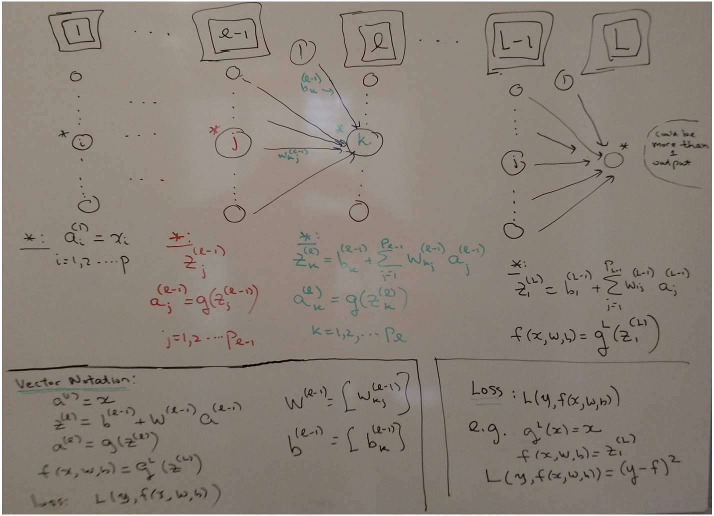
Iterate through the layers:

$$Z^{(\ell)} = b^{(\ell-1)} + W^{(\ell-1)} a^{(\ell-1)}, \;\; a^{(\ell)} = g(Z^{(\ell)}).$$

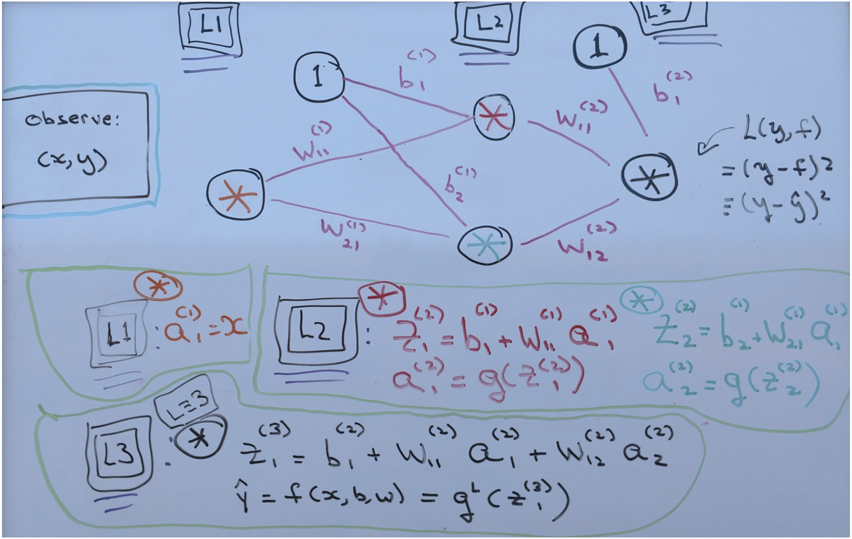
Final output layer and Loss:

$$f(x, W, b) = g^L(Z^L), \text{ Loss: } L(y, f(x, W, b)).$$

Here is the general model:



Simplest interesting case, just the model.



Note:

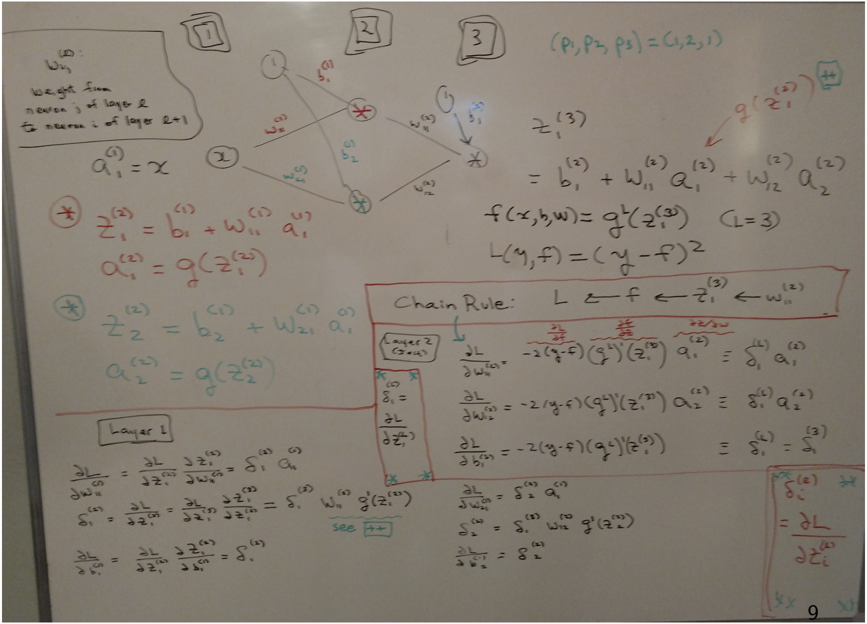
Backpropagation will work by computing:

$$\delta_i^{(\ell)} = \frac{\partial L}{\partial Z_i^{(\ell)}}$$

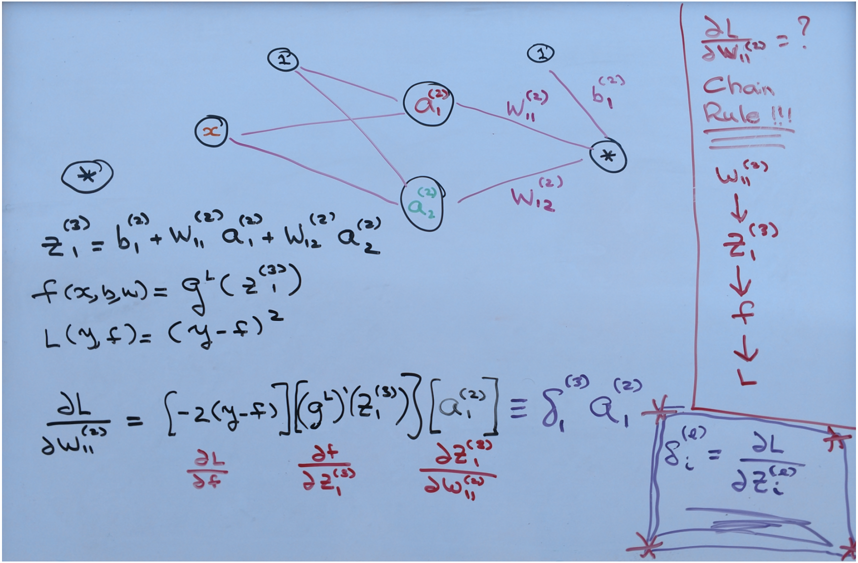
The differential effect of a change in $Z_i^{(\ell)}$ on *the ultimate loss* L.

Simplest interesting case, everything.

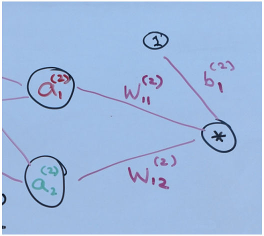
One x, one hidden layer with 2 neurons, one output.



Simplest interesting case, just $\frac{\partial L}{\partial w_{11}^{(2)}}$.



Same thing, just using δ:



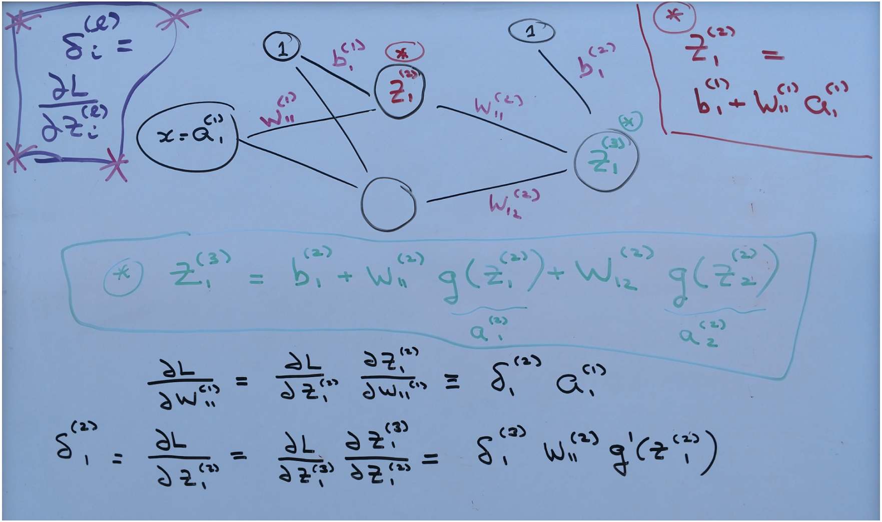
$$Z_1^{(3)} = b_1^{(2)} + w_{11}^{(2)}\, a_1^{(2)} + w_{12}^{(2)}\, a_2^{(2)}.$$

$$\frac{\partial L}{\partial w_{11}^{(2)}} = \frac{\partial L}{\partial Z_1^{(3)}} \frac{\partial Z_1^{(3)}}{\partial w_{11}^{(2)}} = \delta_1^{(3)}\, a_1^{(2)}.$$

Similarly,

$$\frac{\partial L}{\partial w_{12}^{(2)}} = \delta_1^{(3)}\, a_2^{(2)}, \quad \frac{\partial L}{\partial b_1^{(2)}} = \delta_1^{(3)}.$$

Simplest interesting case, just $\frac{\partial L}{\partial w_{11}^{(1)}}$.



$$Z_2^{(2)} = b_2^{(1)} + w_{21}^{(1)} a_1^{(1)}.$$

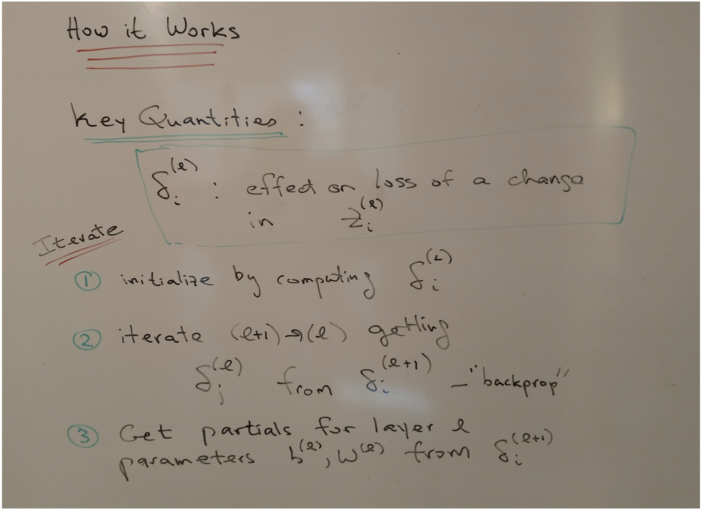
Similarly,

$$\delta_2^{(2)} = \frac{\partial L}{\partial Z_2^{(2)}} = \frac{\partial L}{\partial Z_1^{(3)}} \frac{\partial Z_1^{(3)}}{\partial Z_2^{(2)}} = \delta_1^{(3)} \, w_{12}^{(2)} \, g'(Z_2^{(2)}).$$

$$\frac{\partial L}{\partial w_{21}^{(1)}} = \frac{\partial L}{\partial Z_2^{(2)}} \frac{\partial Z_2^{(2)}}{\partial w_{21}^{(1)}} = \delta_2^{(2)} \, a_1^{(1)}.$$

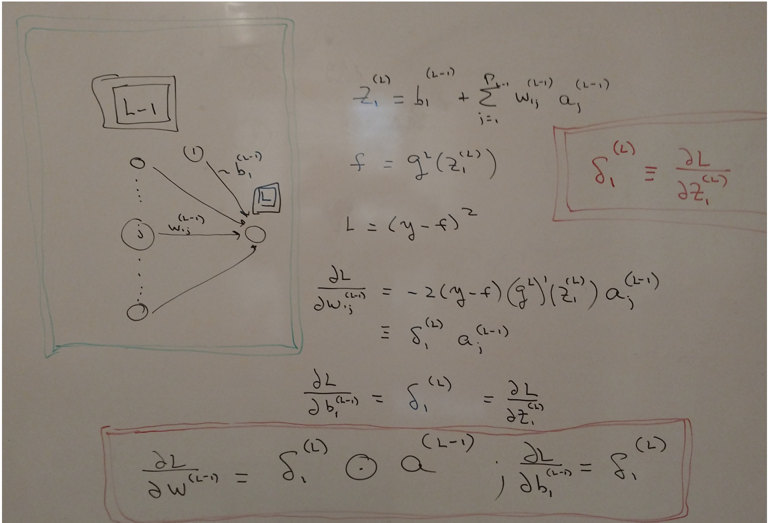
And,

$$\frac{\partial L}{\partial b_1^{(1)}} = \delta_1^{(2)}, \quad \frac{\partial L}{\partial b_2^{(1)}} = \delta_2^{(2)}.$$


How it Works
Key Quantities :
$\delta_i^{(\ell)}$: effect on loss of a change in $z_i^{(\ell)}$
Iterate
① initialize by computing $\delta_i^{(L)}$
② iterate $(\ell+1) \to (\ell)$ getting $\delta_j^{(\ell)}$ from $\delta_i^{(\ell+1)}$ — "backprop"
③ Get partials for layer ℓ parameters $b^{(\ell)}, W^{(\ell)}$ from $\delta_i^{(\ell+1)}$

Here are the partial derivatives associated with the parameters at layer $L-1$.

This will also initialize the back-progagation algorithm for computing the partials for parameters associated with the other layers.



Latex for previous hand written slide.

$$Z_1^{(L)} = b_1^{(L-1)} + \sum_{j=1}^{p_{L-1}} w_{1j}^{(L-1)}\, a_j^{(L-1)}.$$

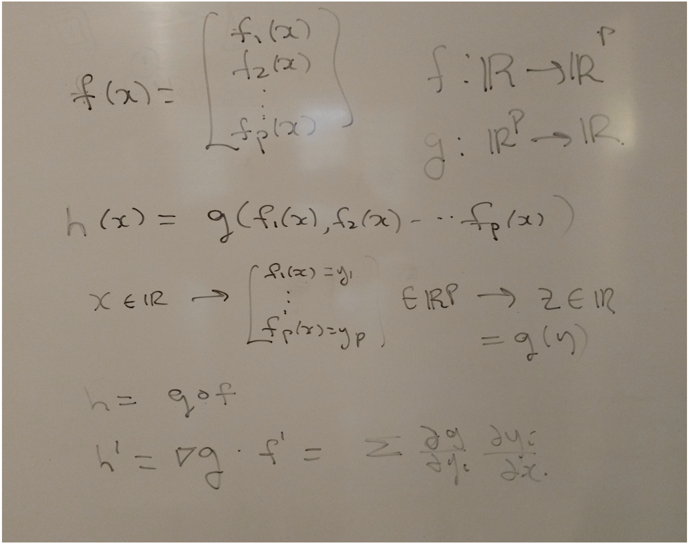
$$f(x, b, w) = g^L(Z_1^{(L)}), \;\; L(y, f) = (y - f)^2$$

$$\delta_1^{(L)} = \frac{\partial L}{\partial Z_1^{(L)}} = \frac{\partial L}{\partial f}\, \frac{\partial f}{\partial Z_1^{(L)}} = [-2(y-f)]\, \left[(g^L)'(Z_1^{(L)}) \right].$$

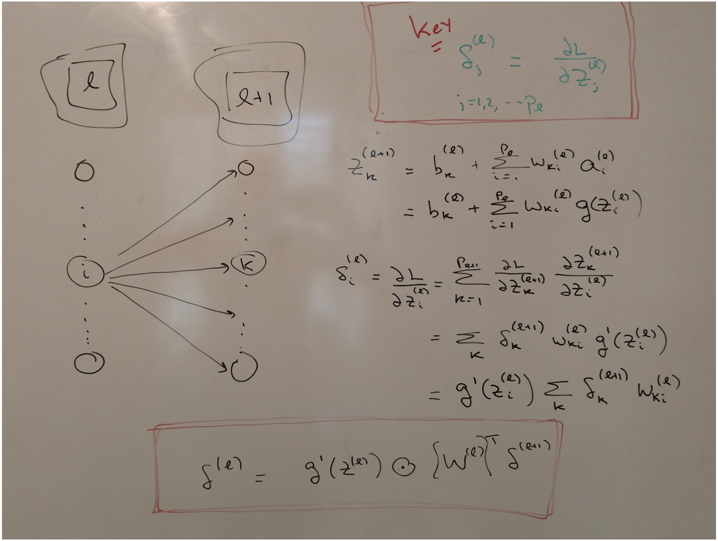
$$\frac{\partial L}{\partial w_{1j}^{(L-1)}} = \frac{\partial L}{Z_1^{(L)}}\, \frac{\partial Z_1^{(L)}}{w_{1j}^{(L-1)}} = \delta_1^{(L)}\, a_j^{(L-1)}.$$

$$\frac{\partial L}{\partial b_1^{(L-1)}} = \delta_1^{(L)}$$

Multivariate version of chain rule.



Here is the iteration for computing the key $\delta_j^{(\ell)}$ quantities.



$$
\begin{aligned}
Z_k^{(\ell+1)} &= b_k^\ell + \sum_{i=1}^{p_\ell} w_{ki}^{(\ell)}\, a_i^{(\ell)} \\
&= b_k^\ell + \sum_{i=1}^{p_\ell} w_{ki}^{(\ell)}\, g(Z_i^{(\ell)})
\end{aligned}
$$

$$
\begin{aligned}
\delta_i^{(\ell)} = \frac{\partial L}{\partial Z_i^{(\ell)}} &= \sum_{k=1}^{p_{\ell+1}} \frac{\partial L}{\partial Z_k^{(\ell+1)}} \frac{\partial Z_k^{(\ell+1)}}{\partial Z_i^{(\ell)}} \\
&= \sum_{k=1}^{p_{\ell+1}} \left[\delta_k^{(\ell+1)}\right] \left[w_{ki}^{(\ell)}\, g'(Z_i^{(\ell)})\right] \\
&= g'(Z_i^{(\ell)}) \sum_{k=1}^{p_{\ell+1}} \left[\delta_k^{(\ell+1)}\right] \left[w_{ki}^{(\ell)}\right]
\end{aligned}
$$

$$\delta^{(\ell)} = g'(Z^{(\ell)}) \odot \left[\left[W^{(\ell)} \right]' \delta^{(\ell+1)} \right]$$

where

$$a \odot b = (a_i b_i)$$

is *elementwise* multiplication, and

$g'(Z^{(\ell)})$ means apply $g' : R \to R$ to each element of $Z^{(\ell)}$.

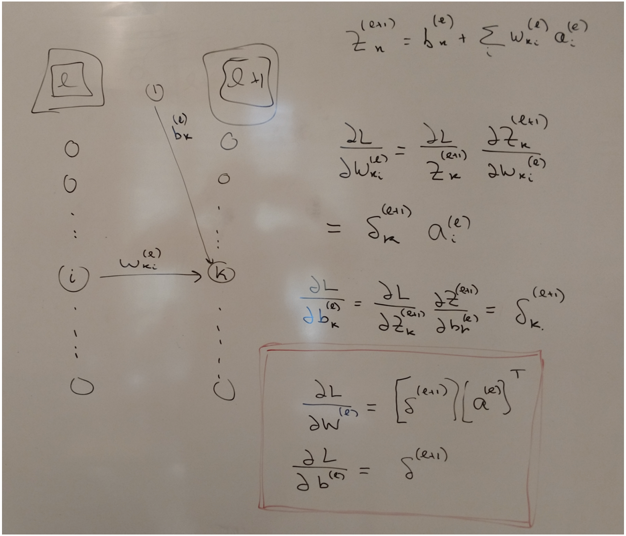
Note:

$Z^{(\ell)} \in R^{p_\ell}$. $g'(Z^{(\ell)}) \in R^{p_\ell}$. $\delta^{(\ell)} \in R^{p_\ell}$.

$\delta^{(\ell+1)} \in R^{(p_\ell+1)}$.

$W^{(\ell)}$ is $p_{(\ell+1)} \times p_\ell$.

Here are the partial derivatives in terms of the $\delta_j^{(\ell)}$.



$$Z_k^{(\ell+1)} = b_k^\ell + \sum_{i=1}^{p_\ell} w_{ki}^{(\ell)} \, a_i^{(\ell)}.$$

$$\begin{aligned} \frac{\partial L}{\partial w_{ki}^{(\ell)}} &= \frac{\partial L}{\partial Z_k^{(\ell+1)}} \frac{\partial Z_k^{(\ell+1)}}{\partial w_{ki}^{(\ell)}} \\ &= \delta_k^{(\ell+1)} \, a_i^{(\ell)} \end{aligned}$$

$$\frac{\partial L}{\partial b_k^{(\ell)}} = \delta_k^{(\ell+1)}$$

$$\frac{\partial L}{\partial W^{(\ell)}} = \left[\frac{\partial L}{\partial w_{ki}^{(\ell)}}\right] = \left[\delta^{(\ell+1)}\right]\left[a^{(\ell)}\right]'$$

$$\frac{\partial L}{\partial b^{(\ell)}} = \left[\frac{\partial L}{\partial b_k^{(\ell)}}\right] = \delta^{(\ell+1)}$$

Neural Nets in a Nutshell

Model and Loss

$$a^{(1)} = x \;;\quad z^{(\ell)} = b^{(\ell-1)} + W^{(\ell-1)} a^{(\ell-1)} \;;\quad a^{(\ell)} = g^{(\ell)}(z^{(\ell)})$$

$$f(x, b, w) = a^{(L)} \;;\quad \min_{b,w} \frac{1}{n} \sum_{i=1}^{n} L(y_i, f(x_i, b, w))$$

Gradient Computation (Backprop)

- $\delta_i^{(L)} = \frac{\partial L}{\partial f} (g^L)'(z_i^{(L)})$
- $\delta^{(\ell)} = (g^{(\ell)})'(z^{(\ell)}) \odot \left[W^{(\ell)}\right]^T \delta^{(\ell+1)}$
- $\frac{\partial L}{\partial W^{(\ell)}} = \left[\delta^{(\ell+1)}\right]\left[a^{(\ell)}\right]^T \qquad \frac{\partial L}{\partial b^{(\ell)}} = \delta^{(\ell+1)}$

*

- ε_k schedule
- Nesterov Momentum
- L^1, L^2 regularization
- Dropout
- ⋮

SGD: Stochastic Gradient Descent

Epochs: $k = 1, 2, \dots K$ (pass through data)

Minibatches: $\{x_i^b, y_i^b\}$ $i = 1, 2, \dots m$, $b = 1, 2, \dots B$

ε_k: learing rate

$\Theta = (b, w)$

for $k = 1, 2, \dots K$

for $b = 1, 2, \dots B$

$$\Theta \to \Theta - \varepsilon_k \frac{1}{m} \sum_{i=1}^{m} \nabla L(x_i^b, y_i^b, \Theta)$$ *