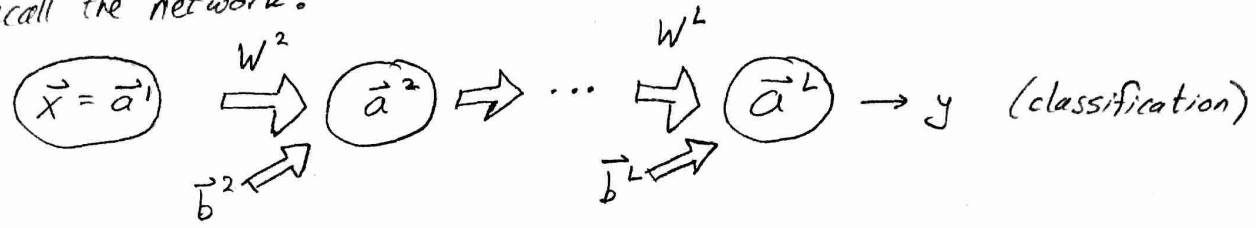
Backpropagation ("learning") algorithm derivation

Recall the network:



We will have the network "learn" by algorithmically adjusting the weights and biases between each layer to make the output closer to the correct classification. Since the output is $y = \arg\max_i \vec{a}^L$, we want to make a_i^L closer to its maximum value 1, for i = index of correct classification.

We will do this by minimizing the "error" * of the last layer of activations $\vec{a}^L$ from the optimal out put $\hat{e}_i$ (again i is index of correct classification). The heuristic chosen for error largely determines how effective the learning is; here, we will use Euclidean square distance, in accordance with Nielson's first two chapters:

$$C = \frac{1}{2} \| \vec{y} - \vec{a}^L \|^2$$

where $\vec{y} = \hat{e}_i$. (We use C to stand for "cost" in accordance with Nielson, though we will often refer to it as "error".)

* Sometimes also referred to as "loss" or "cost"

So we give the network an "example" $\vec{x}$ with a known classification y (and corresponding correct last layer $\vec{y}$). Then adjust weights and biases a bit in the direction that decreases C (i.e. opposite the direction of the derivative of C w.r.t the weights and w.r.t the biases).

Thus we need to compute the derivative of C w.r.t each weight w and bias b in the network. It turns out that it's much easier to do this by taking many derivatives at once in parallel — i.e. gradients and derivative matrices. So we will approach it this way.

Before we get to the derivation, we have to settle two items: first, I will use a non-standard notation for derivatives, since I believe it makes it more clear what type of derivative it is once you get used to it (e.g. gradient of C w.r.t the biases $\vec{b}$ in a particular layer). Refer to the Appendix A for the description of this notation.

Second, we must recall a few rules of derivatives and observe some properties specific to our purposes:

Proposition 1 (Chain Rule)

For a function $f = f(\vec{x})$, where each component of $\vec{x}$ itself depends on a vector $\vec{u}$ (i.e. $x_i = x_i(u_1, \dots, u_k) \quad 1 \le i \le n$):

$$\vec{d}_{\vec{u}} f = D_{\vec{u}} \vec{x} \cdot \vec{d}_{\vec{x}} f$$

Recall the reason behind this is that:

$$\vec{d}_{\vec{u}} f = \begin{bmatrix} \partial_{u_1} f \\ \vdots \\ \partial_{u_k} f \end{bmatrix} \quad \text{where} \quad \begin{aligned} \partial_{u_i} f &= \partial_{x_1} f \cdot \partial_{u_i} x_1 + \cdots + \partial_{x_n} f \cdot \partial_{u_i} x_n \\ &= \vec{d}_{\vec{x}} f \cdot \vec{\partial}_{u_i} \vec{x} \end{aligned}$$

Thus,

$$\vec{d}_{\vec{u}} f = \begin{bmatrix} \vec{d}_{\vec{x}} f \cdot \vec{\partial}_{u_1} \vec{x} \\ \vdots \\ \vec{d}_{\vec{x}} f \cdot \vec{\partial}_{u_k} \vec{x} \end{bmatrix} = \begin{bmatrix} \vec{\partial}_{u_1} \vec{x} \\ \vdots \\ \vec{\partial}_{u_k} \vec{x} \end{bmatrix} \cdot \vec{d}_{\vec{x}} f = D_{\vec{u}} \vec{x} \cdot \vec{d}_{\vec{x}} f \qquad \square$$

But in a special case (which will come up frequently in this derivation), we can simplify this substantially:

Proposition 2 ("Parallel" Chain Rule)

Given the same conditions as Prop. 1, except $\vec{x}$ and $\vec{u}$ have same length, and each x_i depends only on the corresponding u_i, i.e. $x_i = x_i(u_i)$:

$$\vec{d}_{\vec{u}} f = \vec{d}_{\vec{x}} f \odot \vec{d}_{\vec{u}} \vec{x}$$

(recall $\odot$ is the Hadamard Product, and $\vec{d}_{\vec{u}} \vec{x} = (\partial_{u_1} x_1, \dots, \partial_{u_n} x_n)$)

Pf

Since $x_i = x_i(u_i)$, then $\partial_{u_j} x_i = 0$ for $i \neq j$. Thus:

$$\vec{\partial}_{u_i} \vec{x} = \begin{bmatrix} 0 \\ \vdots \\ \partial_{u_i} x_i \\ \vdots \\ 0 \end{bmatrix}$$

and putting together all these columns $1 \leq i \leq n$, we form a <u>diagonal matrix</u>:

$$D_{\vec{u}} \vec{x} = \begin{bmatrix} \partial_{u_1} x_1 & & 0 \\ & \ddots & \\ 0 & & \partial_{u_n} x_n \end{bmatrix}$$

So by Prop. 1:

$$\vec{d}_{\vec{u}} f = D_{\vec{u}} \vec{x} \cdot \vec{d}_{\vec{x}} f = \begin{bmatrix} \partial_{u_1} x_1 & & 0 \\ & \ddots & \\ 0 & & \partial_{u_n} x_n \end{bmatrix} \begin{bmatrix} \partial_{x_1} f \\ \vdots \\ \partial_{x_n} f \end{bmatrix} = \begin{bmatrix} \partial_{x_1} f \cdot \partial_{u_1} x_1 \\ \vdots \\ \partial_{x_n} f \cdot \partial_{u_n} x_n \end{bmatrix}$$

$$= \vec{d}_{\vec{x}} f \odot \vec{d}_{\vec{u}} \vec{x} \quad \square$$

In fact, we can extend this statement more generally:

<u>Proposition 2E</u> (Extended)

Given the same conditions as Prop. 2, except $\vec{x}$ is a f'n of a matrix U of values:

$$U = \begin{bmatrix} u_{11} & \cdots & u_{1n} \\ \vdots & \ddots & \vdots \\ u_{m1} & \cdots & u_{mn} \end{bmatrix}$$

where we denote rows by $\vec{r}_i$ $1 \leq i \leq m$, and cols $\vec{c}_j$ $1 \leq j \leq n$.

If each x_i depends only on the corresponding row $\vec{r}_i$, i.e. $x_i = x_i(\vec{r}_i)$, then:

$$\vec{d}_{\vec{c}_j} f = \vec{d}_{\vec{x}} f \odot \vec{d}_{\vec{c}_j} \vec{x}$$

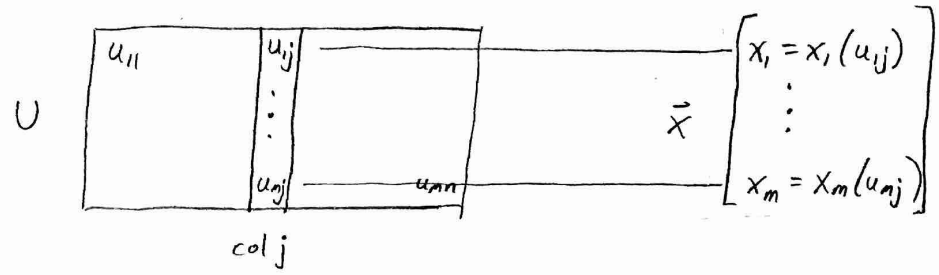
The proof is based on the same idea as in Prop. 2.

(See next page)

(4)

Pf

Note that for any component x and matrix entry u, $\partial_u x \neq 0$ only when u is in the same row as x. Furthermore, if we restrict to only derivatives w.r.t. a single column $\vec{c}$, then u must also be in the column $\vec{c}$.



Only one element is both in the row $\vec{r}_i$ corresponding to x_i and in the column $\vec{c}_j$ — namely, u_{ij}. Thus, for each x_i, only one derivative in $\vec{d}_{\vec{c}_j} x_i$ will be nonzero, $\partial_{u_{ij}} x_i$:

$$\vec{d}_{\vec{c}_j} x_i = \begin{bmatrix} 0 \\ \vdots \\ \partial_{u_{ij}} x_i \\ \vdots \\ 0 \end{bmatrix}$$

So again, when we put these column vectors together for $1 \leq i \leq m$, we get a diagonal matrix:

$$D_{\vec{c}_j} \vec{x} = \begin{bmatrix} \partial_{u_{1j}} x_1 & & 0 \\ & \ddots & \\ 0 & & \partial_{u_{mj}} x_m \end{bmatrix}$$

And by Prop. 1:

$$\vec{d}_{\vec{c}_j} f = D_{\vec{c}_j} \vec{x} \cdot \vec{d}_{\vec{x}} f = \begin{bmatrix} \partial_{u_{1j}} x_1 & & 0 \\ & \ddots & \\ 0 & & \partial_{u_{mj}} x_m \end{bmatrix} \begin{bmatrix} \partial_{x_1} f \\ \vdots \\ \partial_{x_m} f \end{bmatrix} = \begin{bmatrix} \partial_{x_1} f \cdot \partial_{u_{1j}} x_1 \\ \vdots \\ \partial_{x_m} f \cdot \partial_{u_{mj}} x_m \end{bmatrix}$$

$$= \vec{d}_{\vec{x}} f \odot \vec{d}_{\vec{c}_j} \vec{x}$$

□

With these propositions established, we can begin the derivation.

So we wish to find $\partial_b C$ and $\partial_w C$ for each bias and weight in the network. The expression will depend on which layer those biases and weights correspond to, so we'll start in the last layer (this turns out to be easiest). Also we'll find the entire vector $\vec{d}_{\vec{b}} C$ and matrix $D_W C$ at once in a much cleaner way than finding each component individually. (Until further notice I will use the symbols $\vec{b}$, W, $\vec{a}$, $\vec{z}$, etc. all referencing those values in the last layer L.)

Let's begin with something more straight-forward:

$\vec{d}_{\vec{a}} C$

$$\vec{d}_{\vec{a}} C = \vec{d}_{\vec{a}} \left(\tfrac{1}{2} \| \vec{y} - \vec{a} \|^2 \right)$$

If we let $\vec{\delta} = \vec{y} - \vec{a}$, then $\vec{\delta}$ satisfies properties of Prop. 2:

$$\delta_i = \delta_i(a_i) = y_i - a_i$$

So:

$$= \vec{d}_{\vec{\delta}} \left(\tfrac{1}{2} \| \vec{\delta} \|^2 \right) \odot \vec{d}_{\vec{a}} \vec{\delta}$$

By converting the first term of this product into a dot product $\vec{\delta} \cdot \vec{\delta}$ and applying the product rule, we get just $\vec{\delta}$. For the second term:

$$\vec{d}_{\vec{a}} \vec{\delta} = \vec{d}_{\vec{a}} (\vec{y} - \vec{a}) = -\vec{1}$$

(I informally use $\vec{1}$ to denote the vector of 1's.)
Together, we get:

$$\vec{d}_{\vec{a}} C = \vec{\delta} \odot -\vec{1}$$

$$= (\vec{y} - \vec{a}) \odot -\vec{1}$$

$$= \vec{a} - \vec{y}$$

Fact 1: $\vec{d}_{\vec{a}} C = \vec{a} - \vec{y}$

Recalling that $\vec{a} = \vec{\sigma}(\vec{z})$, we can take this to the next step:

$\vec{\partial}_{\vec{z}} C$

$$C = C(\vec{a}(\vec{z}))$$

Note that by definition, $\vec{a}$ also satisfies the conditions for Prop. 2, since $a_i = \sigma(z_i)$. Thus, we can also parallelize this derivative:

$$\vec{\partial}_{\vec{z}} C = \vec{\partial}_{\vec{a}} C \odot \vec{\partial}_{\vec{z}} \vec{a}$$

Expanding the second term, we get:

$$\vec{\partial}_{\vec{z}} \vec{a} = \vec{\partial}_{\vec{z}} \vec{\sigma}(\vec{z}) = \begin{bmatrix} \partial_{z_1} \sigma(z_1) \\ \vdots \\ \partial_{z_n} \sigma(z_n) \end{bmatrix} = \begin{bmatrix} \sigma'(z_1) \\ \vdots \\ \sigma'(z_n) \end{bmatrix}$$

We will label this vector $\vec{\sigma}'(\vec{z})$. So we have our next result:

Fact 2: $\vec{\partial}_{\vec{z}} C = \vec{\partial}_{\vec{a}} C \odot \vec{\sigma}'(\vec{z})$

Armed with these two facts, we can go for the derivative w.r.t the biases, recalling $\vec{z} = W\vec{a}' + \vec{b}$ (here I use $\vec{a}'$ to denote activations in the previous layer, as in the Feedforward derivation).

$\vec{\partial}_{\vec{b}} C$

$$C = C(\vec{z}(\vec{b}))$$

Again, by the parallel nature of the network, $\vec{z}$ satisfies the conditions of Prop. 2: $z_i = z_i(b_i) = \sum w a' + b_i$. So:

$$\vec{\partial}_{\vec{b}} C = \vec{\partial}_{\vec{z}} C \odot \vec{\partial}_{\vec{b}} \vec{z}$$

Examining the second term (since we have the first from Fact 2):

$$\vec{\partial}_{\vec{b}} \vec{z} = \vec{\partial}_{\vec{b}} (W\vec{a}' + \vec{b}) = \vec{1}$$

Thus:

Fact 3: $\vec{\partial}_{\vec{b}} C = \vec{\partial}_{\vec{z}} C$

And at last, we can go for the derivative w.r.t weights.

$D_W C$

We'll tackle the matrix by examining just one column at a time:

$$D_W C = \left[\vec{d}_{\vec{w}_1} C, \cdots, \vec{d}_{\vec{w}_n} C \right]$$

Let's try the general column $\vec{d}_{\vec{w}_i} C$ using the same techniques:

$$C = C(\vec{z}(W))$$

But this is not quite like the other cases. We cannot parallelize w/ Prop. 2 because each $z_i = \sum w a' + b_i$ depends on multiple w's. Which w's though? This sum is a dot product over the <u>i^{th} row of W</u> and $\vec{a}'$. So <u>each z_i depends on row i of W.</u>

These are the conditions for Prop. 2E, and since we're differentiating w.r.t. a column of W, we can apply Prop. 2E to use the same technique:

$$\vec{d}_{\vec{w}_i} C = \vec{d}_{\vec{z}} C \odot \vec{d}_{\vec{w}_i} \vec{z}$$

We compute the second term:

$$\vec{d}_{\vec{w}_i} \vec{z} = \vec{d}_{\vec{w}_i} \left(W a' + \vec{b} \right)$$

$$= \vec{d}_{\vec{w}_i} \left(\sum_k a'_k \vec{w}_k + \vec{b} \right)$$

Differentiating columns $\vec{w}_k$ for $k \neq i$ will give $\vec{0}$, and likewise $\vec{b}$ will give $\vec{0}$, as none of those columns depend on $\vec{w}_i$. However, differentiating the $\vec{w}_i$ term will give us back the coefficient:

$$= a_i' \vec{I}$$

Putting this together, we have:

<u>Fact 4</u>: $\vec{d}_{\vec{w}_i} C = a_i' \vec{d}_{\vec{z}} C$

To get the full matrix of derivatives, we put together all the columns:

$$D_w C = \left[\vec{d}_{\vec{w}_1} C, \cdots, \vec{d}_{\vec{w}_n} C \right]$$

$$= \left[a_1' \vec{d}_{\vec{z}} C, \cdots, a_n' \vec{d}_{\vec{z}} C \right]$$

This fits the definition of the outer product: $\vec{u} \otimes \vec{v} = \vec{u}\vec{v}^t = \left[v_1 \vec{u}, \ldots, v_n \vec{u} \right]$

$$= \vec{d}_{\vec{z}} C \otimes \vec{a}'$$

Fact 5: $D_w C = \vec{d}_{\vec{z}} C \otimes \vec{a}'$

Remember that these all apply to the last layer L. Here we will recap the formulas, putting back in the usual layer superscript notation:

1) $\vec{d}_{\vec{a}^L} C = \vec{a}^L - \vec{y}$

2) $\vec{d}_{\vec{z}^L} C = \vec{d}_{\vec{a}^L} C \odot \vec{\sigma}'(\vec{z}^L)$

3) $\vec{d}_{\vec{b}^L} C = \vec{d}_{\vec{z}^L} C$

4) $\vec{d}_{\vec{w}_i^L} C = a_i^{L-1} \vec{d}_{\vec{z}^L} C$

5) $D_{W^L} C = \vec{d}_{\vec{z}^L} C \otimes \vec{a}^{L-1}$

Notice that each term (except the first) can be expressed in terms of a previous term. Also note the relationship between terms from this layer L and the activations from the previous layer $L-1$. This suggests that there may be a recurrence relation in general for all layers l, and perhaps we'll have the same dependence on earlier terms with a different "base" term $\vec{d}_{\vec{a}^l} C$. In fact, this is the case, and we will prove it by induction backwards on l, from L down to 1. We use the last layer L and these five facts as our base case, and extend to the general layer l next.

First, we must derive the expression for the new "base" term $\vec{d}_{\vec{a}} C$ for the second to last layer $L-1$ to see the pattern, and then we can inductively prove the general case for this term.

Throughout this section, we will adopt the notation using primes to indicate layer numbers. For instance, no primes ($\vec{a}$) denotes the current layer, one prime ($\vec{a}'$) denotes the previous layer, two primes ($\vec{a}''$) denotes two layers before the current layer, and so on. This keeps superscripts clean.

In the following analysis, we treat the last layer as the "current" one.

$\underline{\vec{d}_{\vec{a}'} C}$

$$C = C(\vec{z}(\vec{a}')) \quad \text{recall} \quad \vec{z} = W\vec{a}' + \vec{b}$$

$$\vec{d}_{\vec{a}'} C = D_{\vec{a}'} \vec{z} \cdot \vec{d}_{\vec{z}} C \quad \text{by Prop. 1 (Chain Rule)}$$

Expanding the first term:

$$D_{\vec{a}'} \vec{z} = D_{\vec{a}'} (W\vec{a}' + \vec{b}) = W^t$$

The fact that this is transposed W is not totally obvious. It's a consequence of how we define the D operator in this context, which is:

$$D_{\vec{u}} \vec{x} = \left[\vec{d}_{\vec{u}} x_1, \ldots, \vec{d}_{\vec{u}} x_n \right] = \begin{bmatrix} \partial_{u_1} x_1 & \cdots & \partial_{u_1} x_n \\ \vdots & \ddots & \vdots \\ \partial_{u_m} x_1 & \cdots & \partial_{u_m} x_n \end{bmatrix}$$

Working this out substituting $\vec{a}'$ for $\vec{u}$ and $W\vec{a}' + \vec{b}$ for $\vec{x}$, we get W^t. Thus:

$$\vec{d}_{\vec{a}'} C = W^t \vec{d}_{\vec{z}} C \quad \text{where } \vec{d}_{\vec{z}} C \text{ is as in Fact 2.}$$

We claim this is a recurrence relation that extends to any choice of "current" layer. In fact, using this relation with layer L as our "current" layer as a base case, we now assume this holds for any "current" layer $\ell \leq L$. We now show it still holds going back one more layer, between layers $\ell-1$ and $\ell-2$:

$$C = C(\vec{z}'(\vec{a}'')), \quad \vec{z}' = W'\vec{a}'' + \vec{b}'$$

$$\vec{d}_{\vec{a}''} C = D_{\vec{a}''} \vec{z}' \cdot \vec{d}_{\vec{z}'} C \quad \text{as above}$$

$$D_{\vec{a}''} \vec{z}' = D_{\vec{a}''} (W'\vec{a}'' + \vec{b}) = (W')^t$$

Note: This turns out to be the same argument as above, substituting one prime for no primes, and two primes for one.

Thus:

$$\vec{d}_{\vec{a}''} C = (W')^t \vec{d}_{\vec{z}'} C \quad \square$$

Fact 6: $\vec{d}_{\vec{a}'} C = W^t \vec{d}_{\vec{z}} C$

Having established the base term $\vec{d}_{\vec{a}} C$ for any layer, we can now verify the other formulas for any layer. Using Facts 2-5 at layer L as a base case, we assume they hold at any layer $l \leq L$ and show that they hold at layer $l-1$ (one prime):

$\vec{d}_{\vec{z}'} C$

$C = C(\vec{a}'(\vec{z}'))$, $\vec{a}' = \vec{\sigma}(\vec{z}')$ where $a_i' = \sigma(z_i')$

$\Rightarrow \vec{d}_{\vec{z}'} C = \vec{d}_{\vec{a}'} C \odot \vec{d}_{\vec{z}'} \vec{a}' = \vec{d}_{\vec{a}'} C \odot \vec{\sigma}'(\vec{z}')$ □

$\vec{d}_{\vec{b}'} C$

$C = C(\vec{z}'(\vec{b}'))$, $\vec{z}' = W' \vec{a}'' + \vec{b}'$ where $z_i' = \sum w' a'' + b_i'$

$\Rightarrow \vec{d}_{\vec{b}'} C = \vec{d}_{\vec{z}'} C \odot \vec{d}_{\vec{b}'} \vec{z}' = \vec{d}_{\vec{z}'} C \odot \vec{1} = \vec{d}_{\vec{z}'} C$ □

$D_{W'} C$

$C = C(\vec{z}'(W'))$, $\vec{z}' = W' \vec{a}'' + \vec{b}'$

$D_{W'} C = \left[\vec{d}_{\vec{w}_1'} C, \ldots, \vec{d}_{\vec{w}_n'} C \right]$ for column vectors $\vec{d}_{\vec{w}_i'} C$

$\vec{d}_{\vec{w}_i'} C = \vec{d}_{\vec{z}'} C \odot \vec{d}_{\vec{w}_i'} \vec{z}'$ by Prop 2E (as in Fact 4's analysis)

$= \vec{d}_{\vec{z}'} C \odot a_i'' \vec{1}$

$= a_i'' \vec{d}_{\vec{z}'} C$ This verifies general case for Fact 4

Finally,

$D_{W'} C = \left[a_1'' \vec{d}_{\vec{z}'} C, \ldots, a_n'' \vec{d}_{\vec{z}'} C \right]$

$= \vec{d}_{\vec{z}'} C \otimes \vec{a}''$ This verifies general case for Fact 5 □.

Thus, we have the general case formulas at current layer l (no primes):

- $\vec{d}_{\vec{a}} C = W^t \vec{d}_{\vec{z}} C$, with base case $\vec{d}_{\vec{a}^L} C = \vec{a}^L - \vec{y}$
- $\vec{d}_{\vec{z}} C = \vec{d}_{\vec{a}} C \odot \vec{\sigma}'(\vec{z})$
- $\vec{d}_{\vec{b}} C = \vec{d}_{\vec{z}} C$
- $D_W C = \vec{d}_{\vec{z}} C \otimes \vec{a}'$

So once the network has been given an example $\vec{x}$ with its correct classification y, we feedforward $\vec{x}$ to obtain the activations $\vec{a}$ at each layer. Then we use them to compute the derivatives $\vec{d}_{\vec{b}} C$ and $D_W C$ at each layer. Finally, we adjust the bias vector $\vec{b}$ and weight matrix W at each layer by a small amount in the direction opposite those derivatives. This "small amount" is computed by scaling $\vec{d}_{\vec{b}} C$ and $D_W C$ by a small factor η, often called the "learning rate" (since it determines how much biases and weights change).

Note: $\sigma'(x) = \sigma(x)(1-\sigma(x))$ we use this substitution in the algorithm below.

Here are the formal algorithms for backpropagation and training.

Backpropagate (y, A)

- Given weight matrices $W^2, \ldots, W^L$ and bias vectors $\vec{b}^2, \ldots, \vec{b}^L$ for each layer, and learning rate η,
- Declare vars:
  - $\vec{d}_{\vec{a}} C$, the derivative of cost w.r.t current layer's activations
  - $\vec{d}_{\vec{z}} C$, same w.r.t current layer's $\vec{z}$ values
  - $\Delta \vec{b}$, change in $\vec{b}$ at current layer
  - ΔW, change in W at current layer
  - $\vec{a}$, current layer's activations
  - $\vec{a}'$, previous layer's activations
  - $\vec{y}$, vector corresponding to classification y
- $\vec{y} = \hat{e}_y$
- $\vec{a}$ = last element of A
- $\vec{d}_{\vec{a}} C = \vec{a} - \vec{y}$
- $\vec{d}_{\vec{z}} C = \vec{d}_{\vec{a}} C \odot \left(\vec{a}\,(\vec{1} - \vec{a}')\right)$
- For each layer l from L to 2:
  - $\vec{a}$ = l^{th} elem. of A
  - $\vec{a}'$ = $(l-1)^{th}$ elem. of A
  - $\Delta \vec{b} = -\eta\, \vec{d}_{\vec{z}} C$
  - $\Delta W = -\eta\, \vec{d}_{\vec{z}} C \otimes \vec{a}'$
  - $\vec{d}_{\vec{a}} C = (W^l)^t\, \vec{d}_{\vec{z}} C$
  - $\vec{d}_{\vec{z}} C = \vec{d}_{\vec{a}} C \odot \left(\vec{a}'(\vec{1} - \vec{a}')\right)$
  - $\vec{b}$ += $\Delta \vec{b}$
  - W += ΔW

Train $(\vec{x}_1, \ldots, \vec{x}_n, y_1, \ldots, y_n)$

- Declare vars:
  - A, the list of activations returned by Feedforward
- For each $\vec{x}, y$:
  - A = Feedforward $(\vec{x})$
  - Backpropagate (y, A)