Lecture 17

Error backpropagation

goal: compute 1st & 2nd derivatives of $E(\vec{w})$ [wrt $\vec{w}$] efficiently.

Consider a general NN with arbitrary feed-forward topology, arbitrary activation f's and a general error f'n:

$$E(\vec{w}) = \sum_{n=1}^{N} E_n(\vec{w})$$

Consider first a linear model:

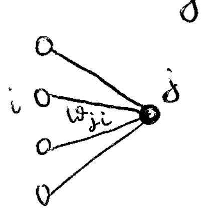
$$\begin{cases} y_k = \sum_i w_{ki} x_i \,, \\ E_n = \frac{1}{2} \sum_k (\underbrace{y_{nk}}_{y_k(\vec{x}_n, \vec{w})} - t_{nk})^2 . \end{cases}$$

Then
$$\frac{\partial E_n}{\partial w_{ji}} = \sum_k (y_{nk} - t_{nk}) \underbrace{\frac{\partial y_{nk}}{\partial w_{ji}}}_{\delta_{kj} x_{ni}} =$$

$$= \underbrace{(y_{nj} - t_{nj})}_{\text{"error signal"}} \underbrace{x_{ni}}_{\text{"input"}}$$

More generally,

$$a_j = \sum_i w_{ji} \underbrace{z_i}_{\text{activation of unit } i \text{ connected to unit } j}$$



Then $z_j = h(a_j)$
$\uparrow$ activation of unit j

Now consider $$\frac{\partial E_n}{\partial w_{ji}} = \underbrace{\frac{\partial E_n}{\partial a_j}}_{\equiv\, \delta_j} \underbrace{\frac{\partial a_j}{\partial w_{ji}}}_{=\, z_i} = \text{(no sum!)}$$

$$= \delta_j z_i .$$

($z_i = 1$ if we are dealing with bias)

If j is an output unit,

$$\delta_j = \frac{\partial E_n}{\partial a_j} = \underbrace{y_j - t_j}_{n \text{ index omitted for simplicity}} \quad \text{for regression OR classification}$$

Indeed, for regression

$y_k = a_k$ (unit activation function)

$\hookrightarrow$ $\frac{\partial E}{\partial a_k} = y_k - t_k$.

For $K=2$ classification,

$$\frac{\partial E_n}{\partial a_j} = -\left[\frac{t_j}{y_j} \underbrace{\frac{\partial y_j}{\partial a_j}}_{y_j(1-y_j)} + (1-t_j)(-1)\frac{1}{1-y_j}\frac{\partial y_j}{\partial a_j} \right] =$$

$y_j = \sigma(a_j)$, sigmoid

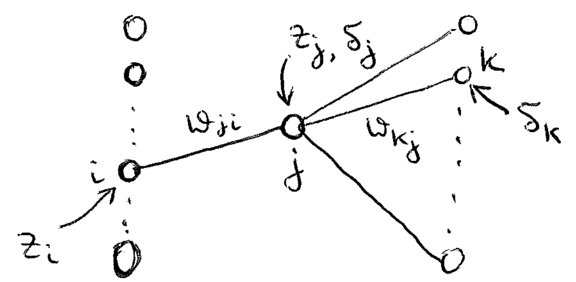
$$= -\left[t_j(1-y_j) - (1-t_j)y_j \right] = y_j - t_j .$$

Same for N $K=2$ classif's and $K>2$ classif'n.

If j is a hidden unit,

$$\delta_j = \frac{\partial E_n}{\partial a_j} = \sum_k \frac{\partial E_n}{\partial a_k}\frac{\partial a_k}{\partial a_j} \quad \langle = \rangle$$

↑ over all "future" units, hidden or output, to which j is connected



$$\langle = \rangle \quad \sum_k \delta_k \frac{\partial}{\partial a_j}\Big(\sum_i w_{ki} h(a_i)\Big) = \sum_{k,i} \delta_k w_{ki} \underbrace{\frac{\partial h(a_i)}{\partial a_j}}_{h'(a_j)\,\delta_{ij}} =$$

$$= h'(a_j) \sum_k w_{kj} \delta_k . \qquad (*)$$

We can use $(*)$ by starting from output units (for which δ_j is easily computed) and computing δ_j's for hidden units in a <u>backpropagation</u> <u>pass</u>.

Finally, use $\dfrac{\partial E_n}{\partial w_{ji}} = \delta_j z_i$ to compute all gradients.

If needed, finish with

$$\frac{\partial E}{\partial w_{ji}} = \sum_n \frac{\partial E_n}{\partial w_{ji}} .$$

The above derivation can be easily generalized to several types of activation functions $h(\cdot)$.

<u>Ex.</u> Consider a two-layer network as before, and focus on regression:

output units $\Rightarrow$ $y_k = a_k$

hidden units $\Rightarrow$ $h(a_k) = \tanh(a_k)$

Note that $\dfrac{d\tanh(a)}{da} = 1 - \tanh^2(a)$

For pattern n, $E_n = \frac{1}{2} \sum_{k=1}^{K} (y_k - t_k)^2$ ← # output units

<u>Forward pass</u>:

$$\begin{cases} a_j = \sum_{i=0}^{D} w_{ji}^{(1)} x_i \\ z_j = \tanh(a_j) \\ y_k = \sum_{j=0}^{M} w_{kj}^{(2)} z_j \end{cases}$$

← # input units (D); x_i ← inputs; ← # hidden units (M)

Next, compute $\delta_k = y_k - t_k$

Use (*) to find

$\delta_j = (1 - z_j^2) \sum_{k=1}^{K} w_{kj}^{(2)} \delta_k$ for hidden units

Finally, compute

$$\frac{\partial E_n}{\partial w_{ji}^{(1)}} = \delta_j x_i \quad , \quad \frac{\partial E_n}{\partial w_{kj}^{(2)}} = \delta_k z_j \qquad (**)$$

and sum over n.

Note that backpropagation is $\Theta(W)$, where W is the total # weights & biases. Most effort goes into evaluating

$$a_j = \sum_i w_{ji} z_i$$

Forward propagation (computing E_n) is $\Theta(W)$ as well. Note that finite difference:

$$\frac{\partial E_n}{\partial w_{ji}} = \frac{E_n(w_{ji}+\xi) - E_n(w_{ji}-\xi)}{2\xi} + \Theta(\xi^2)$$

is $\Theta(W^2)$ since each E_n computation is $\Theta(W)$ & one needs W of those. However, it can be used for numerical checks.

We can check these results by evaluating derivatives of E_n without using backpropagation:

$$E_n = \frac{1}{2} \sum_{k=1}^{K} \underbrace{(y_k - t_k)^2}_{\text{index } n \text{ omitted for clarity}} , \text{ where}$$

$$y_k = \sum_{j=0}^{M} w_{kj}^{(2)} \tanh\left(\sum_{i=0}^{D} w_{ji}^{(1)} x_i\right).$$

Then $$\frac{\partial E_n}{\partial w_{k'j'}^{(2)}} = \sum_{k=1}^{K} (y_k - t_k) \frac{\partial y_k}{\partial w_{k'j'}^{(2)}} =$$

$$= \sum_{k=1}^{K} (y_k - t_k) \sum_{j=0}^{M} \underbrace{\frac{\partial w_{kj}^{(2)}}{\partial w_{k'j'}^{(2)}}}_{\delta_{kk'}\delta_{jj'}} \underbrace{\tanh(\underbrace{\sum_{i=0}^{D} w_{ji}^{(1)} x_i}_{a_j})}_{z_j} \ominus$$

$$\ominus \underbrace{(y_{k'} - t_{k'})}_{\delta_{k'}} z_{j'} = \underline{\underline{\delta_{k'} z_{j'}}} , \quad \text{same as} \quad (**)$$

Next, $$\frac{\partial E_n}{\partial w_{j'i'}^{(1)}} = \sum_{k=1}^{K} (y_k - t_k) \frac{\partial y_k}{\partial w_{j'i'}^{(1)}} =$$

$$= \sum_{k=1}^{K} \underbrace{(y_k - t_k)}_{\delta_k} \sum_{j=0}^{M} w_{kj}^{(2)} \underbrace{\tanh'(a_j)}_{1 - z_j^2} \sum_{i=0}^{D} x_i \underbrace{\frac{\partial w_{ji}^{(1)}}{\partial w_{j'i'}^{(1)}}}_{\delta_{jj'}\delta_{ii'}} \boxed{=}$$

$\boxed{=}$ $\sum_{k=1}^{K} \delta_k \, w_{kj'}^{(2)} \, (1 - z_{j'}^2) \, x_{i'} =$

$= x_{i'} \underbrace{(1 - z_{j'}^2) \sum_{k=1}^{K} \delta_k \, w_{kj'}^{(2)}}_{\delta_{j'}} = \underline{\underline{\delta_{j'} \, x_{i'}}}$, same as (**)

Jacobian matrix

$$J_{ki} = \frac{\partial y_k}{\partial x_i}$$ local sensitivity of outputs to changes in inputs

$$\Delta y_k \simeq \sum_i \underbrace{\frac{\partial y_k}{\partial x_i}}_{J_{ki}} \underbrace{\Delta x_i}_{\text{small input perturbations}} \qquad \left|\frac{\Delta x_i}{x_i}\right| \ll 1$$

J_{ki} can be evaluated using backpropagation:

$$J_{ki} = \frac{\partial y_k}{\partial x_i} = \sum_j \frac{\partial y_k}{\partial a_j} \underbrace{\frac{\partial a_j}{\partial x_i}}_{w_{ji}} \qquad (*)$$

$\sum_j$ over all "future" units j connected to unit i

Further, $$\frac{\partial y_k}{\partial a_j} = \sum_l \frac{\partial y_k}{\partial a_l} \frac{\partial a_l}{\partial a_j} \;=$$

$\sum_l$ over all "future" units l connected to unit j

$$= \sum_l \frac{\partial y_k}{\partial a_l} \frac{\partial}{\partial a_j} \sum_i w_{li} \underbrace{h(a_i)}_{z_i} =$$

$$= \sum_{l,i} \frac{\partial y_k}{\partial a_l} w_{li} h'(a_i) \delta_{ij} = h'(a_j) \sum_l w_{lj} \frac{\partial y_k}{\partial a_l}.$$

Start from output units, e.g. with sigmoid activation functions:

$$\frac{\partial y_k}{\partial a_j} = \sigma'(a_k) \underbrace{\frac{\partial a_k}{\partial a_j}}_{\delta_{kj}} = \sigma(a_j)(1-\sigma(a_j))\,\delta_{kj} .$$

<u>Algorithm</u>:

1. Choose $\vec{x}_n$ and forward-propagate
2. For each k, start with output units and backpropagate $\Rightarrow$ find $\frac{\partial y_k}{\partial a_j}$, $\forall j$
3. Find J_{ki} using $(*)$

Can be checked against finite difference:

$$\frac{\partial y_k}{\partial x_i} = \frac{y_k(x_i+\varepsilon) - y_k(x_i-\varepsilon)}{2\varepsilon} + \mathcal{O}(\varepsilon^2)$$

needs $2D$ forward propagations ($D = \#$ inputs)

<u>Hessian matrix</u>

$$H_{ij} = \frac{\partial^2 E}{\partial w_i \, \partial w_j}$$

$i,j = 1, \ldots, W$ ← total # weights & biases

all weights / biases relabeled using one consecutive index.

(1) H_{ij} used in some non-linear optimization algorithms

(2) H_{ij} is used in Bayesian neural networks

Exact evaluation:

Consider a two-layer network for simplicity.

Define $\underbrace{\delta_k = \frac{\partial E_n}{\partial a_k}}_{\text{as before}}$, $M_{kk'} = \frac{\partial^2 E_n}{\partial a_k \partial a_{k'}}$

here, k & k' can be hidden or output nodes

(a) Both weights in the second (hidden) layer:

$$\frac{\partial^2 E_n}{\partial w^{(2)}_{kj} \partial w^{(2)}_{k'j'}} = \frac{\partial^2 E_n}{\partial a_k \partial a_{k'}} \frac{\partial a_k}{\partial w^{(2)}_{kj}} \frac{\partial a_{k'}}{\partial w^{(2)}_{k'j'}} =$$

outputs hidden units

$$= M_{kk'} z_j z_{j'} .$$

(b) Both weights in the first (input) layer:

$$\frac{\partial^2 E_n}{\partial w^{(1)}_{ji} \partial w^{(1)}_{j'i'}} = \frac{\partial^2 E_n}{\partial a_j \partial a_{j'}} \underbrace{\frac{\partial a_j}{\partial w^{(1)}_{ji}}}_{x_i} \underbrace{\frac{\partial a_{j'}}{\partial w^{(1)}_{j'i'}}}_{x_{i'}} =$$

inputs hidden units

$$= M_{jj'} x_i x_{i'} .$$

(c) One weight in each layer:

$$\frac{\partial^2 E_n}{\partial w^{(1)}_{ji} \partial w^{(2)}_{kj'}} = \frac{\partial}{\partial w^{(1)}_{ji}} \left[\frac{\partial E_n}{\partial a_k} \underbrace{\frac{\partial a_k}{\partial w^{(2)}_{kj'}}}_{z_{j'}} \right] =$$

hidden input output

$$= \overbrace{\frac{\partial^2 E_n}{\partial a_j \partial a_k}}^{M_{jk} = M_{kj}} x_i z_{j'} + \underbrace{\frac{\partial E_n}{\partial a_k}}_{\delta_k} \frac{\partial z_{j'}}{\partial w^{(1)}_{ji}} =$$

$$= M_{jk} x_i z_{j'} + \delta_k h'(a_j) \delta_{jj'} x_i .$$

Thus, we need to backpropagate $M_{kk'}$.

Indeed,
$$\underset{\text{hidden}}{\frac{\partial E_n}{\partial a_j}} = \sum_{\substack{k \\ \text{downstream of} \\ \text{\& connected to } j}} \frac{\partial E_n}{\partial a_k} \underbrace{\frac{\partial \overbrace{a_k}^{=\sum_l w_{kl} h(a_l)}}{\partial a_j}}_{\sum_l w_{kl} h'(a_l) \underbrace{\frac{\partial a_l}{\partial a_j}}_{\delta_{lj}} = w_{kj} h'(a_j)} = h'(a_j) \sum_k \frac{\partial E_n}{\partial a_k} w_{kj} .$$

Now,
$$\underset{\text{hidden}}{\frac{\partial^2 E_n}{\partial a_i \partial a_j}} = h''(a_i)\, \delta_{ij} \sum_k \frac{\partial E_n}{\partial a_k} w_{kj} + h'(a_j) \sum_{\substack{k, k' \\ k \text{ downstream of } j \\ k' \text{ downstream of } i}} \frac{\partial^2 E_n}{\partial a_k \partial a_{k'}} w_{kj} \underbrace{\frac{\partial \overbrace{a_{k'}}^{\sum_l w_{k'l} h(a_l)}}{\partial a_i}}_{\sum_l h'(a_l) \delta_{li} w_{k'l} = h'(a_i) w_{k'i}} =$$

$$= h''(a_i)\, \delta_{ij} \sum_k \underbrace{\frac{\partial E_n}{\partial a_k}}_{\delta_k} w_{kj} + h'(a_i) h'(a_j) \sum_{k,k'} \underbrace{\frac{\partial^2 E_n}{\partial a_k \partial a_{k'}}}_{M_{kk'}} w_{kj} w_{k'i} . \qquad (**)$$

Compute M_{ij} recursively starting from output nodes. For ex., for regression we get:

output nodes
$$\begin{cases} \dfrac{\partial E_n}{\partial a_j} = \dfrac{\partial E_n}{\partial y_j} = y_j - t_j , \\[2ex] \dfrac{\partial^2 E_n}{\partial a_i \partial a_j} \underset{a_i = y_i \text{ here}}{=} \delta_{ij} . \end{cases}$$

Then we can compute M_{ij} for hidden nodes using $(**)$.

Finally, in the "mixed" case:

$$\underset{\substack{\uparrow \\ \text{hidden}}}{\frac{\partial}{\partial a_i}} \underset{\substack{\uparrow \\ \text{output}}}{\frac{\partial E_n}{\partial y_j}} = \frac{\partial}{\partial a_i}(y_j - t_j) = \underset{\substack{\uparrow \\ \text{output}}}{\sum_k} \underbrace{\frac{\partial (y_j - t_j)}{\partial a_k}}_{y_k} \underbrace{\frac{\partial a_k}{\partial a_i}}_{w_{ki} h'(a_i)} =$$

$$= \sum_k \delta_{jk} w_{ki} h'(a_i) = h'(a_i) w_{ji}.$$

Thus we can compute all M_{ij} & therefore all H_{ij}. This is an $\theta(W^2)$ computation.

This can be checked against finite differences:

$$\frac{\partial^2 E}{\partial w_{ji} \partial w_{lk}} = \frac{1}{4\varepsilon^2}\Big[E(w_{ji}+\varepsilon, w_{lk}+\varepsilon) -$$

$$- E(w_{ji}+\varepsilon, w_{lk}-\varepsilon) - E(w_{ji}-\varepsilon, w_{lk}+\varepsilon) +$$

$$+ E(w_{ji}-\varepsilon, w_{lk}-\varepsilon) \Big] + \theta(\varepsilon^2)$$

↑ 4 forward propagations with $\theta(W)$ operations each $\times$ W^2 Hessian elements $\Rightarrow$

$\Rightarrow$ $\theta(W^3)$ computation.

However, we can use a mixed approach:

$$\frac{\partial^2 E}{\partial w_{ji} \partial w_{lk}} = \frac{1}{2\varepsilon}\Bigg[\underbrace{\frac{\partial E}{\partial w_{ji}}\bigg|_{w_{lk}+\varepsilon}} - \underbrace{\frac{\partial E}{\partial w_{ji}}\bigg|_{w_{lk}-\varepsilon}} \Bigg] + \theta(\varepsilon^2).$$

compute using backpropagation in $\theta(W)$ steps $\times$ W weights to perturbed $(w_{lk} \pm \varepsilon)$: $\theta(W^2)$ computation just as with explicit backpropagation above.