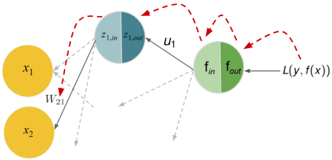
Deep Learning

Basic Backpropagation 2



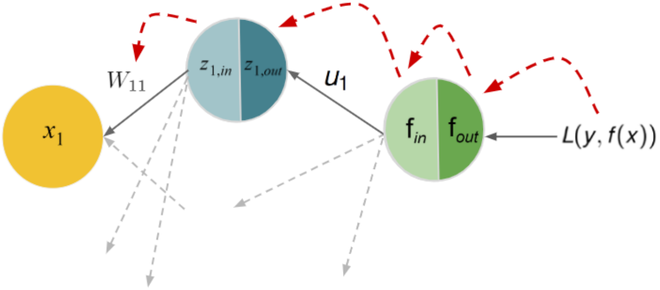
Learning goals

- Backprop formalism and recursion

BACKWARD COMPUTATION AND CACHING

In the XOR example, we computed:

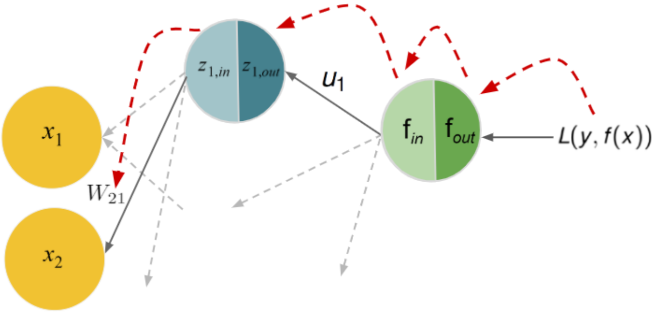
$$\frac{\partial L(y, f(\mathbf{x}))}{\partial W_{11}} = \frac{\partial L(y, f(\mathbf{x}))}{\partial f_{out}} \cdot \frac{\partial f_{out}}{\partial f_{in}} \cdot \frac{\partial f_{in}}{\partial z_{1,out}} \cdot \frac{\partial z_{1,out}}{\partial z_{1,in}} \cdot \frac{\partial z_{1,in}}{\partial W_{11}}$$



BACKWARD COMPUTATION AND CACHING

Next, let us compute:

$$\frac{\partial L\left(y, f(\mathbf{x})\right)}{\partial W_{21}} = \frac{\partial L\left(y, f(\mathbf{x})\right)}{\partial f_{out}} \cdot \frac{\partial f_{out}}{\partial f_{in}} \cdot \frac{\partial f_{in}}{\partial z_{1,out}} \cdot \frac{\partial z_{1,out}}{\partial z_{1,in}} \cdot \frac{\partial z_{1,in}}{\partial W_{21}}$$



BACKWARD COMPUTATION AND CACHING

- Examining the two expressions:

$$\frac{\partial L(y, f(\mathbf{x}))}{\partial W_{11}} = \frac{\partial L(y, f(\mathbf{x}))}{\partial f_{out}} \cdot \frac{\partial f_{out}}{\partial f_{in}} \cdot \frac{\partial f_{in}}{\partial z_{1,out}} \cdot \frac{\partial z_{1,out}}{\partial z_{1,in}} \cdot \frac{\partial z_{1,in}}{\partial W_{11}}$$

$$\frac{\partial L(y, f(\mathbf{x}))}{\partial W_{21}} = \frac{\partial L(y, f(\mathbf{x}))}{\partial f_{out}} \cdot \frac{\partial f_{out}}{\partial f_{in}} \cdot \frac{\partial f_{in}}{\partial z_{1,out}} \cdot \frac{\partial z_{1,out}}{\partial z_{1,in}} \cdot \frac{\partial z_{1,in}}{\partial W_{21}}$$

- Significant overlap / redundancy in the two expressions.
- **Again**: Let's call this subexpression δ_1 and cache it.

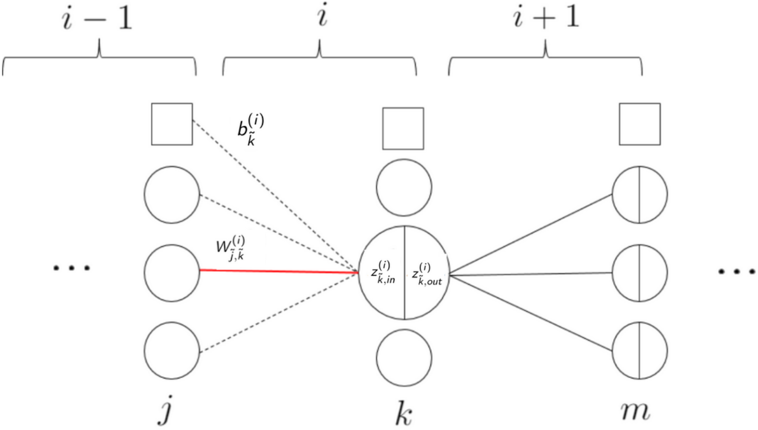
$$\delta_1 = \frac{\partial L(y, f(\mathbf{x}))}{\partial z_{1,in}} = \frac{\partial L(y, f(\mathbf{x}))}{\partial f_{out}} \cdot \frac{\partial f_{out}}{\partial f_{in}} \cdot \frac{\partial f_{in}}{\partial z_{1,out}} \cdot \frac{\partial z_{1,out}}{\partial z_{1,in}}$$

$$\frac{\partial L(y, f(\mathbf{x}))}{\partial W_{11}} = \delta_1 \cdot \frac{\partial z_{1,in}}{\partial W_{11}} \quad \text{and} \quad \frac{\partial L(y, f(\mathbf{x}))}{\partial W_{21}} = \delta_1 \cdot \frac{\partial z_{1,in}}{\partial W_{21}}$$

- δ_1 can also be seen as an **error signal** that represents how much the loss L changes when the input $z_{1,in}$ changes.

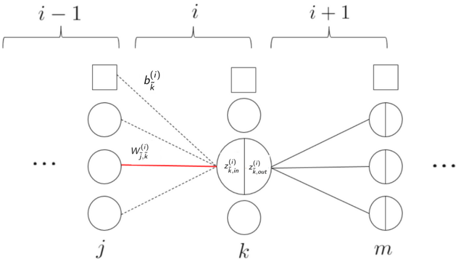
BACKPROP: RECURSION

- Let us now derive a general formulation of backprop.
- The neurons in layers $i-1$, i and $i+1$ are indexed by j, k and m, respectively.
- The output layer will be referred to as layer O.



Credit: Erik Hallström

BACKPROP: RECURSION



- Let $\delta_{\tilde{k}}^{(i)}$ (also: error signal) for a neuron $\tilde{k}$ in layer i represent how much the loss L changes when the input $z_{\tilde{k},in}^{(i)}$ changes:

$$\delta_{\tilde{k}}^{(i)} = \frac{\partial L}{\partial z_{\tilde{k},in}^{(i)}} = \frac{\partial L}{\partial z_{\tilde{k},out}^{(i)}} \frac{\partial z_{\tilde{k},out}^{(i)}}{\partial z_{\tilde{k},in}^{(i)}} = \sum_m \left(\frac{\partial L}{\partial z_{m,in}^{(i+1)}} \frac{\partial z_{m,in}^{(i+1)}}{\partial z_{\tilde{k},out}^{(i)}} \right) \frac{\partial z_{\tilde{k},out}^{(i)}}{\partial z_{\tilde{k},in}^{(i)}}$$

- Note: The sum in the expression above is over all the neurons in layer $i+1$. This is simply an application of the (multivariate) chain rule.

BACKPROP: RECURSION

Using

$$\begin{aligned} z_{\tilde{k},out}^{(i)} &= \sigma(z_{\tilde{k},in}^{(i)}) \\ z_{m,in}^{(i+1)} &= \sum_k W_{k,m}^{(i+1)} z_{k,out}^{(i)} + b_m^{(i+1)} \end{aligned}$$

we get:

$$\begin{aligned} \delta_{\tilde{k}}^{(i)} &= \sum_m \left(\frac{\partial L}{\partial z_{m,in}^{(i+1)}} \frac{\partial z_{m,in}^{(i+1)}}{\partial z_{\tilde{k},out}^{(i)}} \right) \frac{\partial z_{\tilde{k},out}^{(i)}}{\partial z_{\tilde{k},in}^{(i)}} \\ &= \sum_m \left(\frac{\partial L}{\partial z_{m,in}^{(i+1)}} \frac{\partial \left(\sum_k W_{k,m}^{(i+1)} z_{k,out}^{(i)} + b_m^{(i+1)} \right)}{\partial z_{\tilde{k},out}^{(i)}} \right) \frac{\partial \sigma(z_{\tilde{k},in}^{(i)})}{\partial z_{\tilde{k},in}^{(i)}} \\ &= \sum_m \left(\frac{\partial L}{\partial z_{m,in}^{(i+1)}} W_{\tilde{k},m}^{(i+1)} \right) \sigma'(z_{\tilde{k},in}^{(i)}) = \sum_m \left(\delta_{m,in}^{(i+1)} W_{\tilde{k},m}^{(i+1)} \right) \sigma'(z_{\tilde{k},in}^{(i)}) \end{aligned}$$

Therefore, we now have a **recursive definition** for the error signal of a neuron in layer i in terms of the error signals of the neurons in layer $i+1$ and, by extension, layers {i+2, i+3 . . . , O}!

BACKPROP: RECURSION

- Given the error signal $\delta_{\tilde{k}}^{(i)}$ of neuron $\tilde{k}$ in layer i, the derivative of loss L w.r.t. to the weight $W_{\tilde{j},\tilde{k}}$ is simply:

$$\frac{\partial L}{\partial W_{\tilde{j},\tilde{k}}^{(i)}} = \frac{\partial L}{\partial z_{\tilde{k},in}^{(i)}} \frac{\partial z_{\tilde{k},in}^{(i)}}{\partial W_{\tilde{j},\tilde{k}}^{(i)}} = \delta_{\tilde{k}}^{(i)} z_{\tilde{j},out}^{(i-1)}$$

because $z_{\tilde{k},in}^{(i)} = \sum_j W_{j,\tilde{k}}^{(i)} z_{j,out}^{(i-1)} + b_{\tilde{k}}^{(i)}$

- Similarly, the derivative of loss L w.r.t. bias $b_{\tilde{k}}^{(i)}$ is:

$$\frac{\partial L}{\partial b_{\tilde{k}}^{(i)}} = \frac{\partial L}{\partial z_{\tilde{k},in}^{(i)}} \frac{\partial z_{\tilde{k},in}^{(i)}}{\partial b_{\tilde{k}}^{(i)}} = \delta_{\tilde{k}}^{(i)}$$

BACKPROP: RECURSION

- It is not hard to show that the error signal δ^{i} for an entire layer i is ($\odot$ = element-wise product):
  - $\delta^{(O)} = \nabla_{f_{out}} L \odot \tau'(f_{in})$
  - $\delta^{(i)} = W^{(i+1)}\delta^{(i+1)} \odot \sigma'(z_{in}^{(i)})$
- Therefore, backpropagation works by computing and storing the error signals **backwards**. That is, starting at the output layer and ending at the first hidden layer. This way, the error signals of later layers **propagate backwards** to the earlier layers.
- The derivative of the loss L w.r.t. a given weight is computed efficiently by plugging in the cached error signals, thereby avoiding expensive and redundant computations.