Neural Networks: Gradient Descent and Backpropagation

Mark Craven and David Page
Computer Sciences 760
Spring 2019

Goals for the lecture

you should understand the following concepts

- gradient descent with a linear output unit + squared error
- gradient descent with a sigmoid output unit + cross entropy
- backpropagation

Taking derivatives in neural nets

recall the chain rule from calculus

$$y = f(u)$$
$$u = g(x)$$

$$\frac{\partial y}{\partial x} = \frac{\partial y}{\partial u}\frac{\partial u}{\partial x}$$

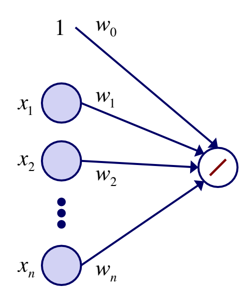
we'll make use of this as follows

$$\frac{\partial E}{\partial w_i} = \frac{\partial E}{\partial o}\frac{\partial o}{\partial net}\frac{\partial net}{\partial w_i} \qquad net = w_0 + \sum_{i=1}^{n} w_i x_i$$

Gradient descent: simple case #1

Consider a simple case of a network with one <u>linear</u> output unit and no hidden units:

$$o^{(d)} = net^{(d)} = w_0 + \sum_{i=1}^{n} w_i x_i^{(d)}$$



let's learn w_i's that minimize squared error

$$E(\mathbf{w}) = \frac{1}{2}\sum_{d \in D}\left(y^{(d)} - o^{(d)}\right)^2$$

batch case

$$\frac{\partial E}{\partial w_i} = \frac{\partial}{\partial w_i}\frac{1}{2}\sum_{d \in D}\left(y^{(d)} - o^{(d)}\right)^2$$

online case

$$\frac{\partial E^{(d)}}{\partial w_i} = \frac{\partial}{\partial w_i}\frac{1}{2}\left(y^{(d)} - o^{(d)}\right)^2$$

Gradient descent: simple case #1

let's focus on the online case (stochastic gradient descent):

$$\frac{\partial E^{(d)}}{\partial w_i} = \frac{\partial E^{(d)}}{\partial o^{(d)}} \frac{\partial o^{(d)}}{\partial net^{(d)}} \frac{\partial net^{(d)}}{\partial w_i}$$

$$\frac{\partial E^{(d)}}{\partial o^{(d)}} = -\left(y^{(d)} - o^{(d)}\right)$$

$$\frac{\partial o^{(d)}}{\partial net^{(d)}} = 1$$

(linear output unit)

$$\frac{\partial net^{(d)}}{\partial w_i} = x_i^{(d)}$$

$$\frac{\partial E^{(d)}}{\partial w_i} = \left(o^{(d)} - y^{(d)}\right) x_i^{(d)}$$

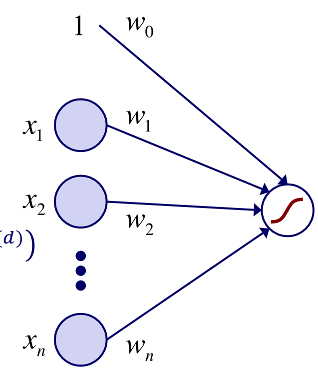
Gradient descent: simple case #2

Now let's consider the case in which we have a sigmoid output unit, no hidden units, and cross-entropy loss function:

$$net^{(d)} = w_0 + \sum_{i=1}^{n} w_i x_i^{(d)}$$

$$o^{(d)} = \frac{1}{1 + e^{-net^{(d)}}}$$

$$E(\boldsymbol{w}) = \sum_{d \in D} -y^{(d)} \ln\left(o^{(d)}\right) - \left(1 - y^{(d)}\right) \ln\left(1 - o^{(d)}\right)$$



useful property:

$$\frac{\partial o^{(d)}}{\partial net^{(d)}} = o^{(d)}(1 - o^{(d)})$$

Gradient descent: simple case #2

$$\frac{\partial E^{(d)}}{\partial w_i} = \frac{\partial E^{(d)}}{\partial o^{(d)}} \frac{\partial o^{(d)}}{\partial net^{(d)}} \frac{\partial net^{(d)}}{\partial w_i}$$

$$\frac{\partial E^{(d)}}{\partial o^{(d)}} = \frac{o^{(d)} - y^{(d)}}{o^{(d)}(1 - o^{(d)})}$$

$$\frac{\partial o^{(d)}}{\partial net^{(d)}} = o^{(d)}\left(1 - o^{(d)}\right)$$

$$\frac{\partial net^{(d)}}{\partial w_i} = x_i^{(d)}$$

$$\frac{\partial E^{(d)}}{\partial w_i} = \left(o^{(d)} - y^{(d)}\right) x_i^{(d)}$$

Backpropagation

- now we've covered how to do gradient descent for single-layer networks
- how can we calculate $\frac{\partial E}{\partial w_i}$ for every weight in a multilayer network?

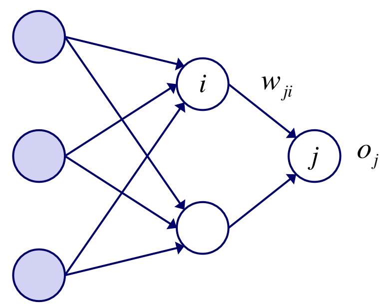
  ➔ backpropagate errors from the output units to the hidden units

Backpropagation notation

let's consider the online case, but drop the $^{(d)}$ superscripts for simplicity

we'll use

- subscripts on y, o, net to indicate which unit they refer to
- subscripts to indicate the unit a weight emanates from and goes to



Backpropagation

each weight is changed by $$\Delta w_{ji} = -\eta \frac{\partial E}{\partial w_{ji}}$$

$$= -\eta \frac{\partial E}{\partial net_j} \frac{\partial net_j}{\partial w_{ji}}$$

$$= \eta \, \delta_j \, o_i$$

this term is x_i if i is an input unit

where $$\delta_j = -\frac{\partial E}{\partial net_j}$$

Backpropagation

each weight is changed by $\Delta w_{ji} = \eta \ \delta_j \ o_i$

where $\delta_j = -\dfrac{\partial E}{\partial net_j}$

suppose we're using sigmoids and cross-entropy

$\delta_j = y_j - o_j$ if j is an output unit

$\delta_j = o_j(1-o_j)\sum_k \delta_k w_{kj}$ if j is a hidden unit

same as single-layer net

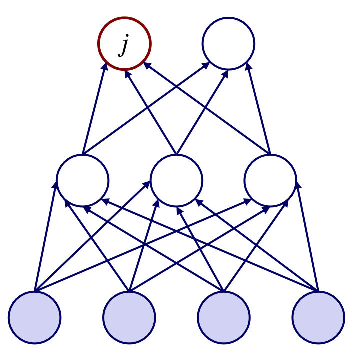
Backpropagation illustrated

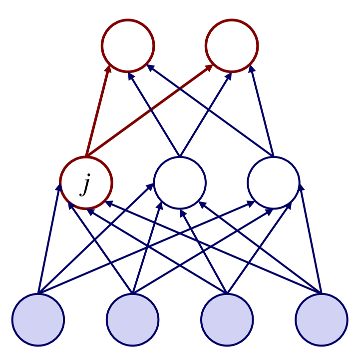
1. calculate error of output units

$\delta_j = y_j - o_j$

2. calculate error for hidden units

$\delta_j = o_j(1-o_j)\sum_k \delta_k w_{kj}$

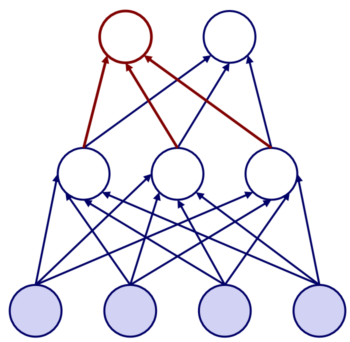




Backpropagation illustrated

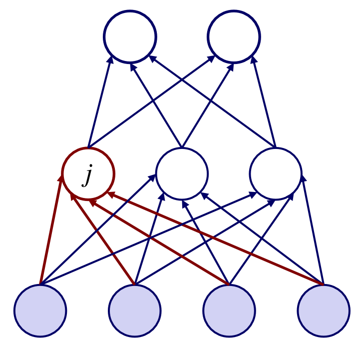
3. determine updates for weights going to output units

$$\Delta w_{ji} = \eta \ \delta_j \ o_i$$



4. determine updates for weights to hidden units using hidden-unit errors

$$\Delta w_{ji} = \eta \ \delta_j \ o_i$$



Backpropagation

- particular derivatives depend on **loss** and **activation** functions
- here we show derivatives for **cross entropy** and **sigmoid** functions
- gradient descent and backprop generalize to other cases in which these functions are differentiable