Week 8 Neural Net

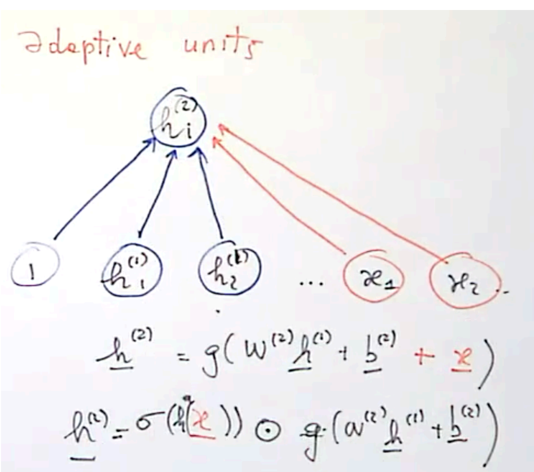
define neural net: Rather than placing basis functions by hand,
pick the family of basis functions, “learn” the locations and any other parameters from data

Regularization: L2, L1, input noise (data aug), dropout, batch norm, skip connection, early stopping

Train more epoch: could be seen as model getting more complex (weights larger, decision boundary more complex)
train & val loss w.r.t epoch VS # of parameters

Adaptive unit:
Skip connection but put x inside activation (make sure dimension match)
=> input participate into this transformation
e.g. hidden units depends on input; want input being processed by all hidden layers

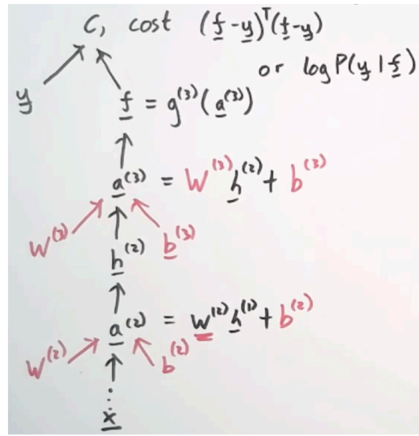
OR:
=> input activate the layer (set layer as 1 or 0 using sigmoid)



Forward Prop:
have a Directed Acyclic Graph (DAG), e.g.:
OR right graph: Cost C depends on training labels y & neural net functions f
activation a depends on w and b…

$$z = \exp(\sin(u^2)\log(u^2))$$

$$\begin{tikzcd}
 & & w \arrow[dr] & & \\
u \arrow[r] & v \arrow[ur] \arrow[dr] & & y \arrow[r] & z \\
 & & x \arrow[ur] & &
\end{tikzcd}$$



Forward derivative theta^dot: any intermediate quantity theta w.r.t leftmost element u

from left to right

= derivatives of the elementary function used at that stage * already computed value (use the chain rule)

$$\dot{\theta} = \frac{\partial \theta}{\partial u}$$

Back Prop:

from right to left

any intermediate quantity theta **w.r.t final value z**

use the chain rule (this = derivative of current function * already calculated)

$$\bar{\theta} = \frac{\partial z}{\partial \theta}$$

If Z is a matrix:

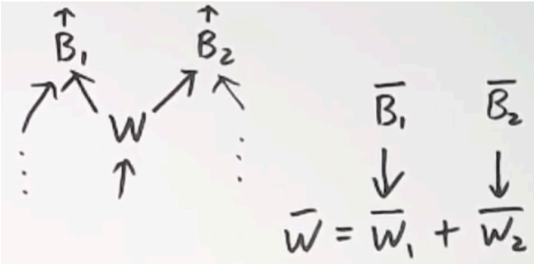
derivative of cost c (final element) w.r.t every elements i,j in matrix Z

(or represent derivative via matrices)

$$\bar{Z}_{ij} = \frac{\partial c}{Z_{ij}}$$

when have multiple children (to final cost C) (e.g. at point v)

do BP individually, add those up



Forward derivative at rightmost node = backward derivative at leftmost node

$$\dot{z} = \bar{u} = \frac{\partial z}{\partial u}$$

Goal of BP: Neural net fitting wants derivative of final cost c, w.r.t all free parameters in the graph(a,h,w,b),

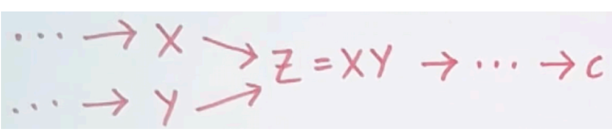
s.t. we could optimize those parameters

Computationally efficient: get derivative of the cost w.r.t everything, with only the cost of evaluating the NN 2-3 times

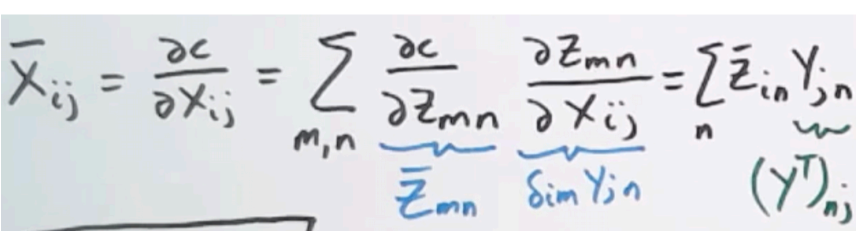
forward prob: have to perturb (change a bit) each parameter one at a time, have to re evaluate cost w.r.t every parameters one a a time

Efficient BP for matrix

For Z = XY:



General equation for matrix BP:



But if breaking down one element of output matrix Z, = one row of X * one column of Y

since derivative w.r.t X i j, only the j th term involving the j th column of X is relevant (underlined, not constant as long as m=I; this term = chronic delta)
all other terms are constant w.r.t X i j

$$
\begin{aligned}
Z_{mn} &= \sum_p X_{mp} Y_{pn} \\
&= X_{m1} Y_{1n} + X_{m2} Y_{2n} + \dots \\
&\quad \dots \underbrace{X_{mj} Y_{jn}}_{\delta_{im} Y_{jn}} + \dots
\end{aligned}
$$

so BP partial derivative = computational cost of one element of output matrix,

$$
\boxed{\bar{X} = \bar{Z} Y^{T}}
$$

$M \times P \qquad M \times N \quad N \times P$

Other rules:(memorize!)

matrix product: $C = AB \quad \Rightarrow \quad \bar{A} = \bar{C} B^{\top}$ and $\bar{B} = A^{\top} \bar{C}$,

matrix addition: $C = A + B \quad \Rightarrow \quad \bar{A} = \bar{C}$ and $\bar{B} = \bar{C}$,

$C = AB^{\top} \quad \Rightarrow \quad \bar{A} = \bar{C} B$ and $\bar{B} = \bar{C}^{\top} A$,