Feed-forward Neural Networks

$$\underline{f} = g^{(3)}(\underbrace{W^{(3)}\underline{h}^{(2)} + \underline{b}^{(3)}}_{\underline{a}^{(3)}})$$

$$\uparrow$$

$$\underline{h}^{(2)} = g^{(2)}(W^{(2)}\underline{h}^{(1)} + \underline{b}^{(2)})$$

$$\uparrow$$

$$\underline{h}^{(1)} = g^{(1)}(W^{(1)}\underline{x} + \underline{b}^{(1)})$$

$$\uparrow$$

$\underline{x}$, inputs

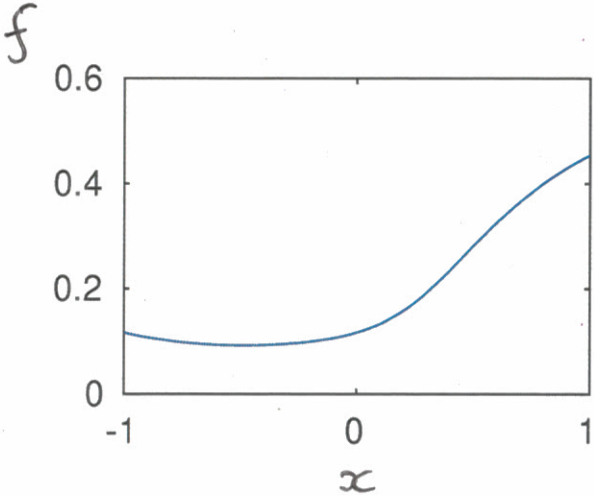
$$h_k^{(2)} = g^{(2)}\left(\sum_\ell W_{k\ell}^{(2)} h_\ell^{(1)} + b_k^{(2)}\right)$$

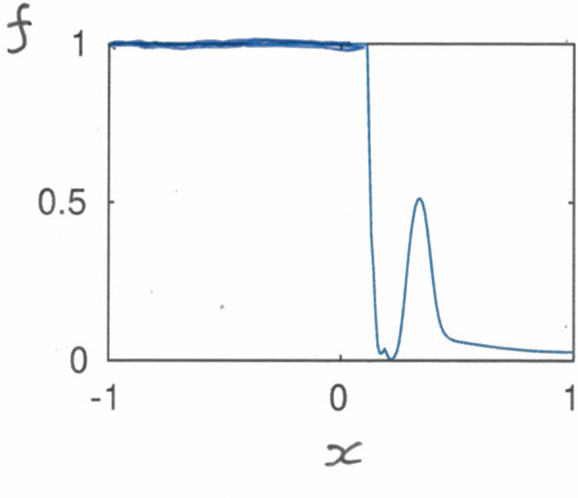
Homework: Try plotting for random weights.

Change: scale W, # layers, g, without $\underline{b}$

```
gg = @(a) 1./(1 + exp(-a));
X = (-1:0.01:1)'; % Nx1
H1 = gg(X * randn(1, 100)); % Nx100
H2 = gg(H1 * randn(100, 50)); % Nx50
F = gg(H2 * randn(50, 1)); % Nx1
plot(X, F);
```

```
gg = @(a) 1./(1 + exp(-a));
X = (-1:0.01:1)'; % Nx1
H1 = gg(X * randn(1, 100) * 10); % Nx100
H2 = gg(H1 * randn(100, 50) * 10); % Nx50
F = gg(H2 * randn(50, 1) * 10); % Nx1
plot(X, F);
```


f
0.6
0.4
0.2
0
-1
0
1
x


f
1
0.5
0
-1
0
1
x

Initialization

Don't set all weights to zero or the same value

Naive $W_{ij}^{(l)} \sim N(0, 1)$

Use "randn"

[Revise background notes expectations]

Sum of K random values

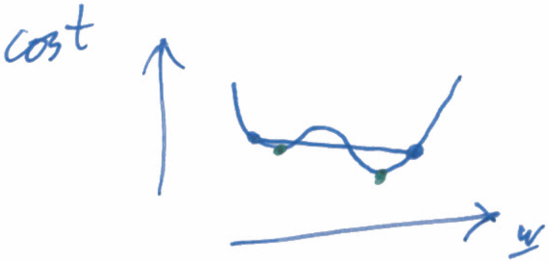
They are typically $\sim \pm 1$ each

Typically sum $\sim \pm \sqrt{K}$

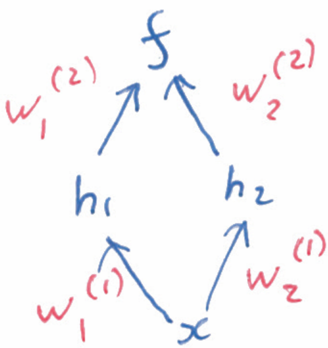
Example initialization $w \sim N\left(0, \left(\frac{1}{\sqrt{K}}\right)^2\right)$

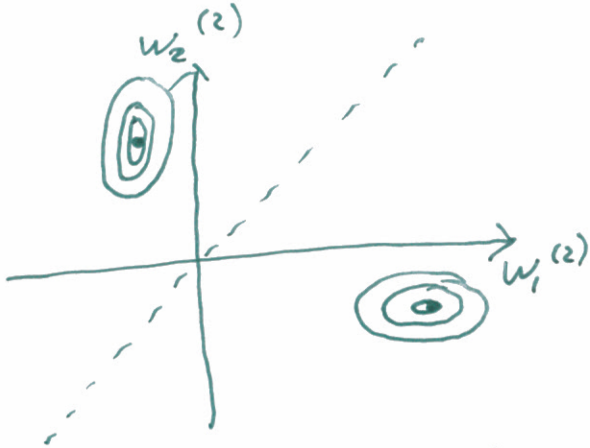
MLP: Has more sophisticated ideas.

NN don't have a convex cost



No unique optimum
$\Rightarrow$ not convex.





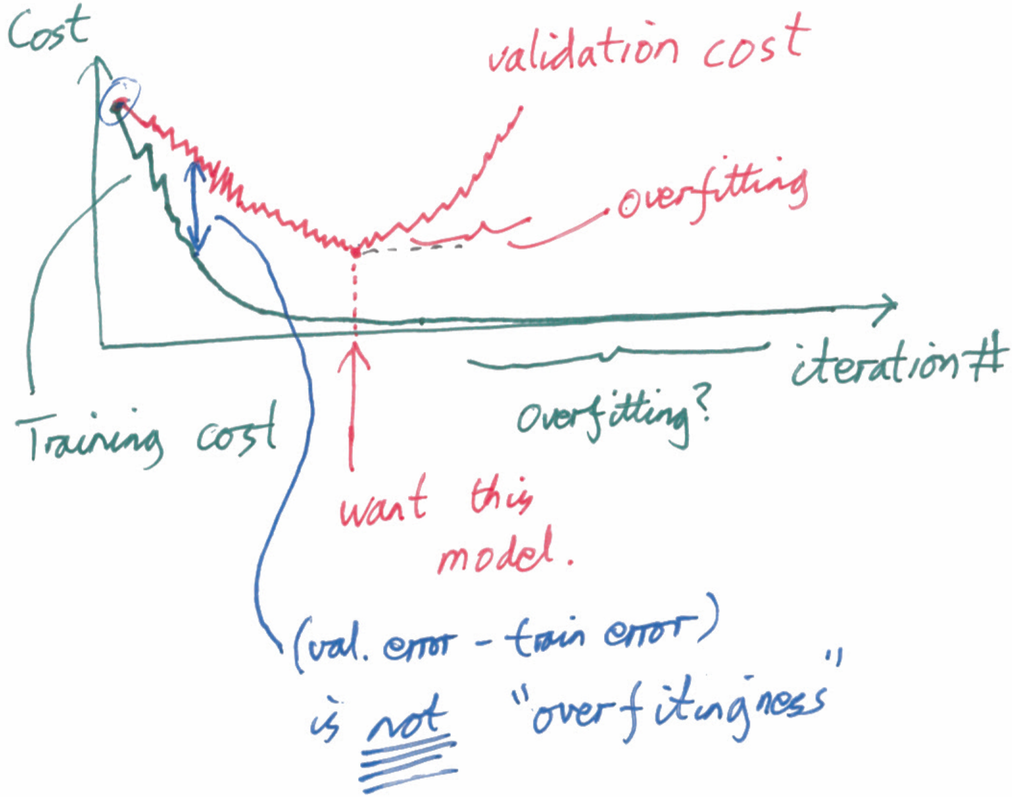
Set $\underline{w}^{(1)}$ to best values
for $\underline{w}^{(2)}$

- These local optima, don't matter
  $\rightarrow$ because the functions (predictions) are the same.
- Not all optima are equivalent

Regularization – Early-stopping

Could do L2 regularization

Set λ ... cross-validate



Every R updates:

If val. cost is the smallest I've seen:

Store: weights & val cost.

If val. cost hasn't improved in 20 ~~updates:~~ looks at val

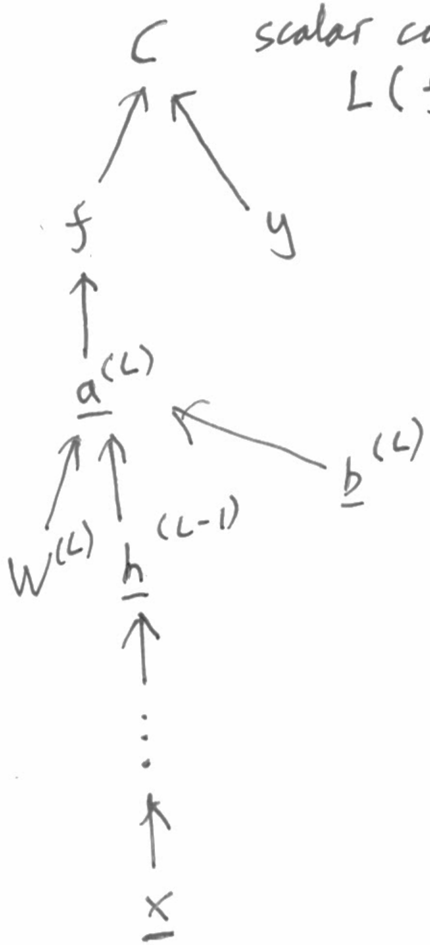
Stop. Return weights we last stored.

or reduce η

Learning rates η_t

Getting gradients – Reverse-mode differentiation

"Back propagation"



scalar cost, $L(f, y)$

Graph
DAG

Strategy:

- For every intermediate θ

get $\bar{\theta} = \frac{\partial c}{\partial \theta}$

$$\bar{f} = \frac{\partial c}{\partial f}$$

$$\overline{a_i^{(L)}} = \frac{\partial c}{\partial a_i^{(L)}}$$

$$\overline{W_{ij}^{(L)}} = \frac{\partial c}{\partial W_{ij}^{(L)}}$$