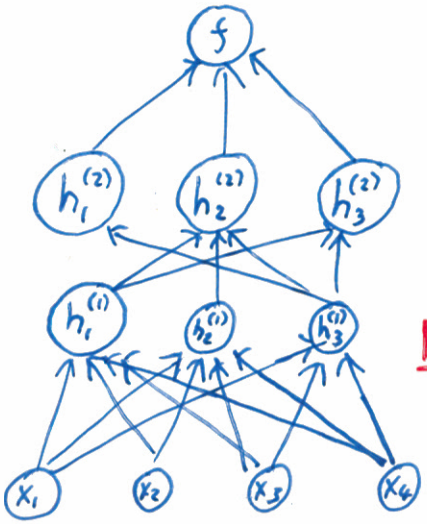
Feed-forward Neural Networks



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

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

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

$$\underline{x}$$

Ouput(s)

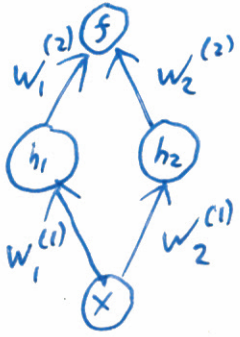
Hidden Layer 2

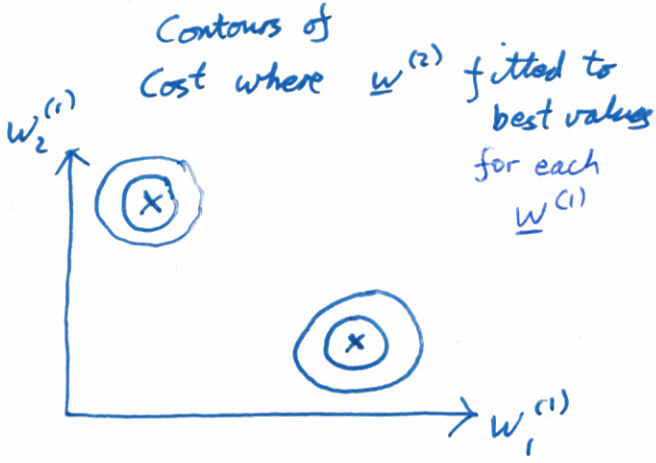
Hidden Layer 1

Visible Layer / Inputs

- When f is a scalar, $W^{(3)}\underline{h}^{(2)} = \underline{w}^{(3)T}\underline{h}^{(2)}$
- Other architectures possible, e.g. "skip connections"
- Specialist layers exist for images / audio:
  Convolutional neural nets or "ConvNets"
  ... and others

Why are NN's not convex?



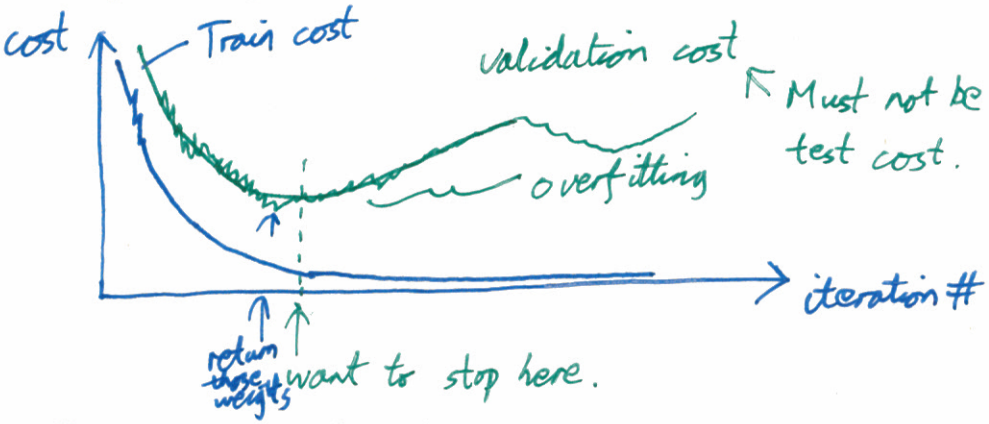


- Local optima are a problem
- Random restarts could help
  - expensive on big problems.

Regularization by early stopping

Can do L2 regularization... cross-validate λ?

Early stopping:



Every K updates:

If val. cost the smallest we've seen:

Store the weights & val cost

If val. cost hasn't improved in 20 updates

terminate.

Regularization by adding noise

Add noise to weights

When doing each SGD update

add Gaussian noise to the weights

For linear models the average effect is the same as L2 regularization.

Add noise to data

Randomly corrupt features, for example set to zero.

Reading: "Nightmare at test-time"

"Denoising Autoencoders"

See also: Drop-Out

We will need gradients

Cost function $E = \sum_{n=1}^{N} L(y, f)$

For Gradient methods we need

$\frac{\partial E}{\partial W_{ij}^{(l)}}$ and $\frac{\partial E}{\partial b_i^{(l)}}$ and any other free parameters.

A computation could be: