Machine Learning 10-601

Tom M. Mitchell
Machine Learning Department
Carnegie Mellon University

September 27, 2017

This section:
- Backpropagation
- Representation learning
- Convolutional nets

Reading:
- Goodfellow: Convolutional nets

Training Deep Nets

1. Choose loss function J(θ) to optimize
   - sum of squared errors for y continuous: $\Sigma\ (y - h(x; \theta))^2$
   - maximize conditional likelihood: $\Sigma \log P(y|x; \theta)$
   - MAP estimate: $\Sigma \log P(y|x; \theta)\ P(\theta)$
   - ~~0/1 loss. Sum of classification errors: $\Sigma\ \delta(y = h(x; \theta)$~~
   - …
2. Design network architecture
   - Network of layers (ReLU's, sigmoid, convolutions, ...)
   - Widths of layers
   - Fully or partly interconnected
   - ...
3. Training algorithm
   - Derive gradient components $\frac{\partial J(\theta)}{\partial \theta_i}$
   - Choose gradient descent method, including stopping condition
   - Experiment with alternative architectures

Example: Learn probabilistic XOR

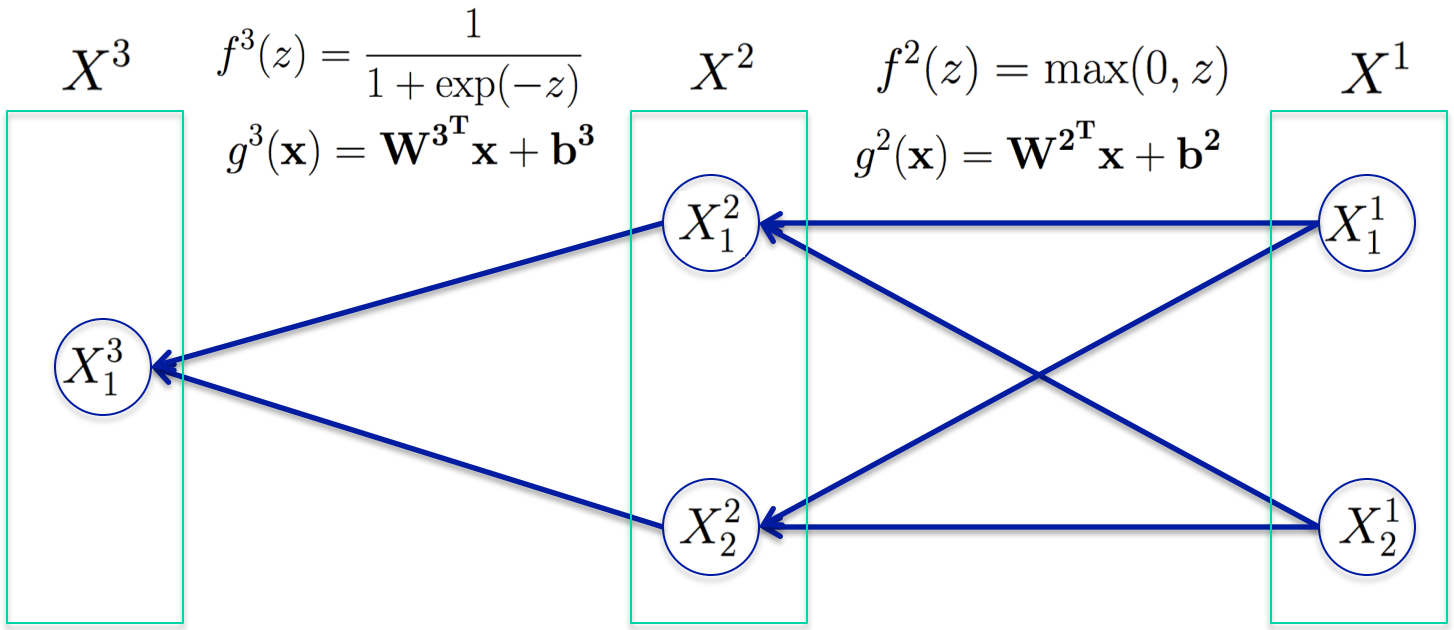
- Given boolean Y, X_1, X_2 learn $P(Y|X_1,X_2)$, where

$$P(Y = 0|X_1 = X_2) = 0.9$$

$$P(Y = 1|X_1 \neq X_2) = 0.9$$

- Can we learn this with logistic regression?

Sigmoid unit $f^3(g^3(\mathbf{x}))$ ReLU units $f^2(g^2(\mathbf{x}))$



Loss function $J(\theta)$ to be minimized: negative log likelihood

$$J(\theta) = \sum_{\langle \mathbf{x}, y \rangle \in D} -\log P(Y = y | X = \mathbf{x})$$

where $X_1^3 = P(Y = 1 | X = \mathbf{X^1})$, $\theta = \{\mathbf{W^3}, \mathbf{b^3}, \mathbf{W^2}, \mathbf{b^2}\}$

Example: Learn probabilistic XOR

- Given boolean Y, X_1, X_2 learn $P(Y|X_1,X_2)$, where

$$P(Y = 0|X_1 = X_2) = 0.9$$

$$P(Y = 1|X_1 \neq X_2) = 0.9$$

- Choose max conditional likelihood,
  equally, minimize negative log conditional likelihood

$$\begin{aligned} J(\theta) &= -\sum_k \log P(Y = y_k|X = \mathbf{x}_k) \\ &= -\sum_k y_k \log P(Y = 1|X = x_k) + (1 - y_k)(1 - P(Y = 1|X = x_k)) \end{aligned}$$

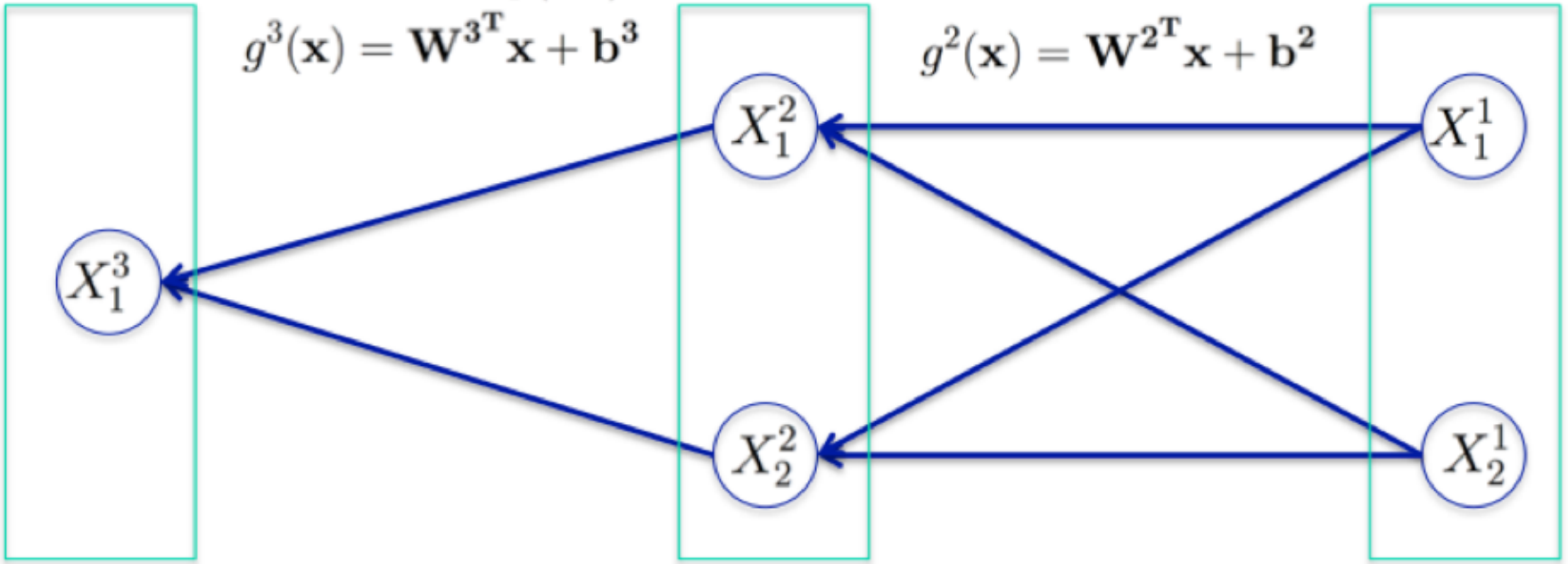
Example

Feed Forward

Sigmoid unit $f^3(g^3(\mathbf{x}))$ ReLU units $f^2(g^2(\mathbf{x}))$

X^3 $f^3(z) = \frac{1}{1+\exp(-z)}$ X^2 $f^2(z) = \max(0, z)$ X^1

$g^3(\mathbf{x}) = \mathbf{W^{3T}x} + \mathbf{b^3}$ $g^2(\mathbf{x}) = \mathbf{W^{2T}x} + \mathbf{b^2}$



X^3	$g^3(X^2)$	W^{3T}		b^3	X^2	$g^2(X^1)$	W^{2T}		b^2	X^1
0.53	0.12	0.10	-0.09	0.10	0.20	0.20	0.10	-0.10	0.10	1
					0.00	-0.15	-0.20	0.10	0.05	0
					1					*1*

Derive the gradient we need

$$
\begin{aligned}
J(\theta) &= -\sum_k \log P(Y = y_k | X = \mathbf{x}_k) \\
&= -\sum_k y_k \log P(Y = 1 | X = x_k) + (1 - y_k)(1 - P(Y = 1 | X = x_k))
\end{aligned}
$$

simplify notation by considering just one training example

$$
\begin{aligned}
\frac{\partial J(\theta)}{\partial X_1^3} &= \frac{\partial \left(-Y \log P(Y = 1|X) - (1 - Y)(1 - P(Y = 1|X))\right)}{\partial X_1^3} \\
&= \frac{\partial \left(-Y \log X_1^3 - (1 - Y) \log(1 - X_1^3)\right)}{\partial X_1^3} \\
&= \frac{-Y}{X_1^3} - (1 - Y)\frac{1}{1 - X_1^3}(-1) \\
&= \frac{-Y}{X_1^3} + \frac{(1 - Y)}{1 - X_1^3}
\end{aligned}
$$

recall $\frac{\partial \ln z}{\partial z} = \frac{1}{z}$

Derive the gradient we need

$$J(\theta) = -\sum_k \log P(Y = y_k | X = \mathbf{x}_k)$$

$$= -\sum_k y_k \log P(Y = 1 | X = x_k) + (1 - y_k)(1 - P(Y = 1 | X = x_k))$$

simplify notation by considering just one training example:

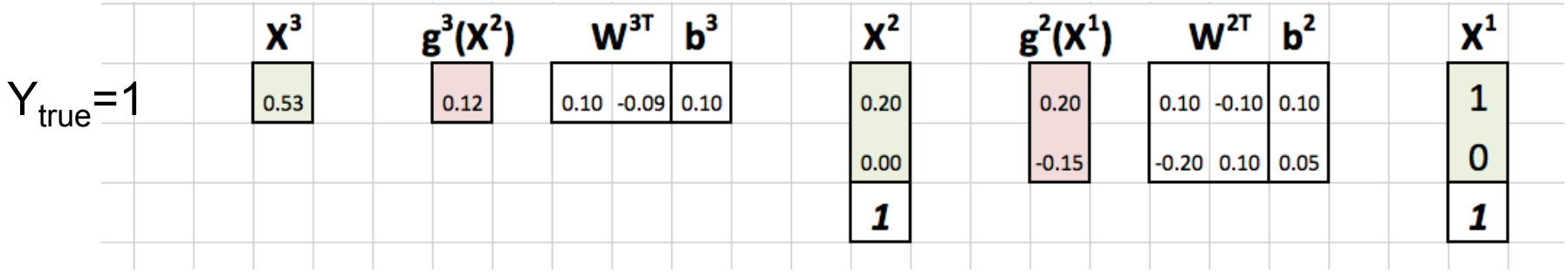
$$\frac{\partial J(\theta)}{\partial X_1^3} = \frac{\partial \left(-Y \log P(Y = 1|X) - (1 - Y)(1 - P(Y = 1|X))\right)}{\partial X_1^3}$$

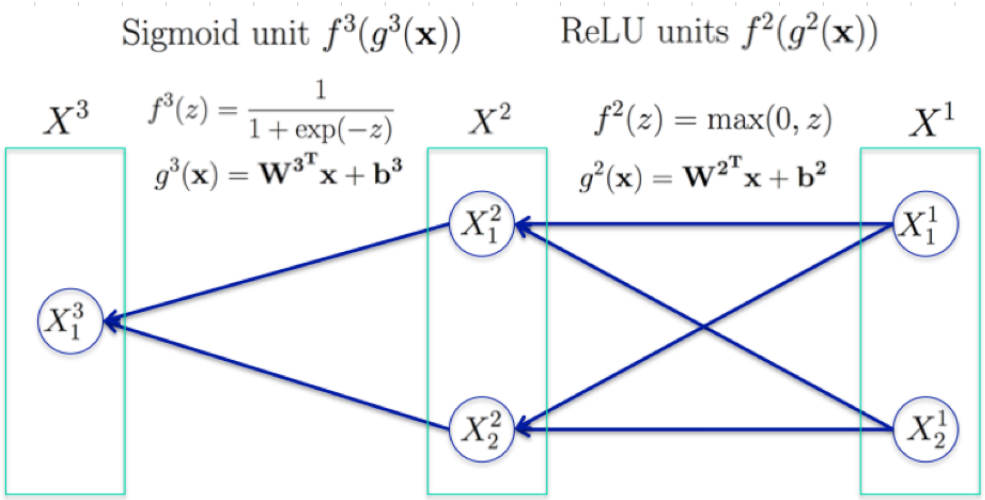
$$= \frac{\partial \left(-Y \log X_1^3 - (1 - Y) \log(1 - X_1^3)\right)}{\partial X_1^3}$$

$$= \frac{-Y}{X_1^3} - (1 - Y)\frac{1}{1 - X_1^3}(-1)$$

$$= \frac{-Y}{X_1^3} + \frac{(1 - Y)}{1 - X_1^3}$$

recall $\frac{\partial \ln z}{\partial z} = \frac{1}{z}$



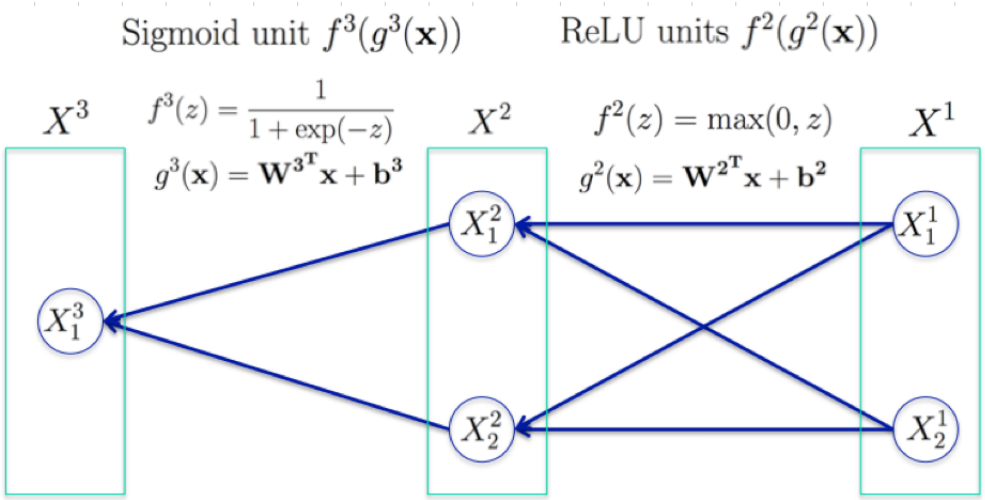


$$\frac{\partial J(\theta)}{\partial g_1^3} = \frac{\partial J(\theta)}{\partial X_1^3}\frac{\partial X_1^3}{\partial g^3} =$$

recall $\frac{\partial \sigma(z)}{\partial z} = \sigma(z)(1-\sigma(z))$
where $\sigma(z) = \frac{1}{1+\exp(-z)}$

$Y_{true}=1$

X^3	$g^3(X^2)$	W^{3T}		b^3	X^2	$g^2(X^1)$	W^{2T}		b^2	X^1
0.53	0.12	0.10	-0.09	0.10	0.20	0.20	0.10	-0.10	0.10	1
					0.00	-0.15	-0.20	0.10	0.05	0
					1					*1*

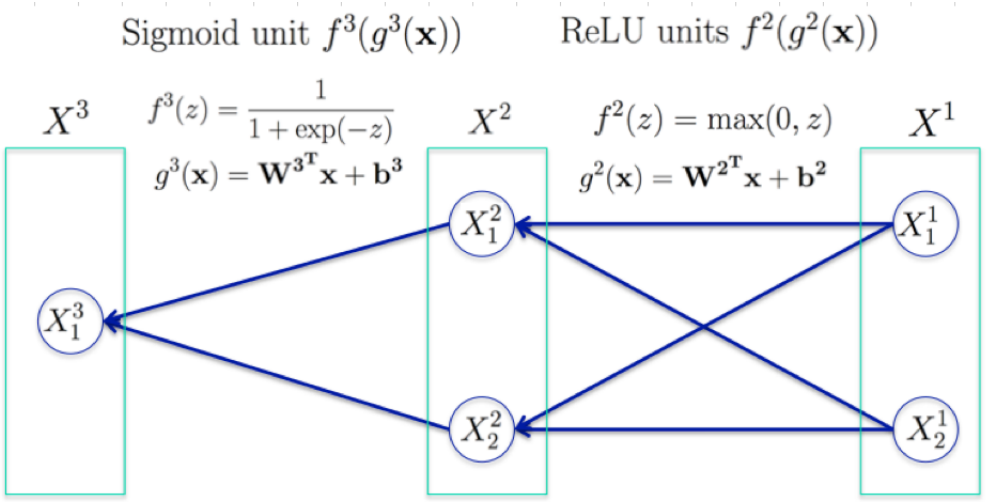


$$\frac{\partial J(\theta)}{\partial g_1^3} = \frac{\partial J(\theta)}{\partial X_1^3}\frac{\partial X_1^3}{\partial g^3} = \frac{\partial J(\theta)}{\partial X_1^3} X_1^3(1-X_1^3) =$$

recall $\frac{\partial \sigma(z)}{\partial z} = \sigma(z)(1-\sigma(z))$
where $\sigma(z) = \frac{1}{1+\exp(-z)}$

$Y_{true}=1$

$\mathbf{X^3}$	$\mathbf{g^3(X^2)}$	$\mathbf{W^{3T}}$		$\mathbf{b^3}$	$\mathbf{X^2}$	$\mathbf{g^2(X^1)}$	$\mathbf{W^{2T}}$		$\mathbf{b^2}$	$\mathbf{X^1}$
0.53	0.12	0.10	-0.09	0.10	0.20	0.20	0.10	-0.10	0.10	1
					0.00	-0.15	-0.20	0.10	0.05	0
					1					***1***

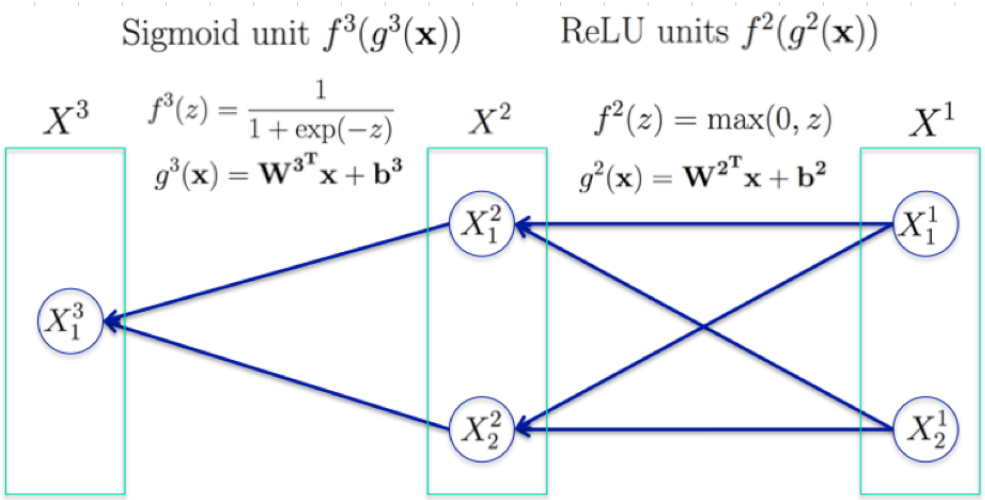


$$\frac{\partial J(\theta)}{\partial g_1^3} = \frac{\partial J(\theta)}{\partial X_1^3}\frac{\partial X_1^3}{\partial g^3} = \frac{\partial J(\theta)}{\partial X_1^3} X_1^3(1-X_1^3) \quad = -0.47$$

$$\frac{\partial J(\theta)}{\partial w_i^3} = \frac{\partial J(\theta)}{\partial g^3}\frac{\partial g^3}{\partial w_i^3} = \frac{\partial J(\theta)}{\partial g^3} X_i^2$$

$Y_{true}=1$

X^3	$g^3(X^2)$	W^{3T}		b^3	X^2	$g^2(X^1)$	W^{2T}		b^2	X^1
0.53	0.12	0.10	-0.09	0.10	0.20	0.20	0.10	-0.10	0.10	1
					0.00	-0.15	-0.20	0.10	0.05	0
					1					*1*



$$\frac{\partial J(\theta)}{\partial g_1^3} = \frac{\partial J(\theta)}{\partial X_1^3}\frac{\partial X_1^3}{\partial g^3} = \frac{\partial J(\theta)}{\partial X_1^3} X_1^3(1-X_1^3)$$

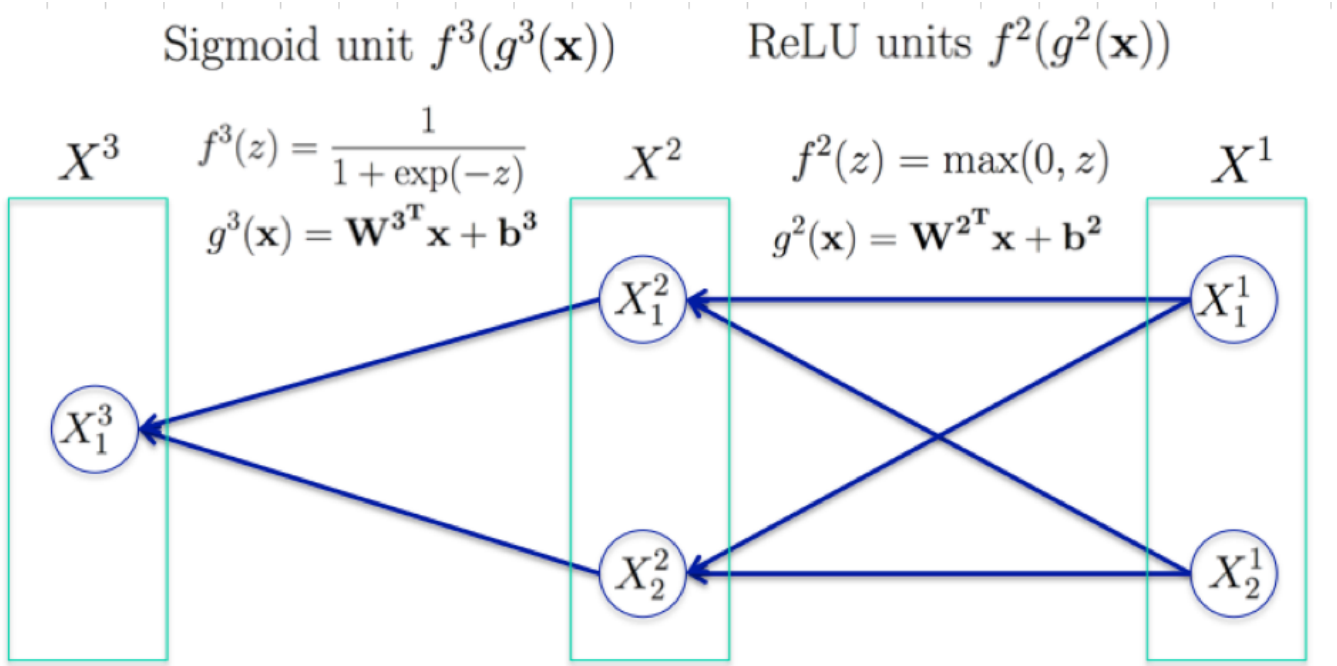
$$\frac{\partial J(\theta)}{\partial w_i^3} = \frac{\partial J(\theta)}{\partial g^3}\frac{\partial g^3}{\partial w_i^3} = \frac{\partial J(\theta)}{\partial g^3} X_i^2$$

$$\frac{\partial J(\theta)}{\partial X_i^2} = \frac{\partial J(\theta)}{\partial g^3}\frac{\partial g^3}{\partial X_i^2} = \frac{\partial J(\theta)}{\partial g^3} w_i^3$$

$$\frac{\partial J(\theta)}{\partial g_i^2} = \frac{\partial J(\theta)}{\partial X_i^2}\frac{\partial X_i^2}{\partial g_i^2} = \frac{\partial J(\theta)}{\partial X_i^2} \times [\text{if } g_i^2 > 0 \text{ then 1 else 0}]$$

$$\frac{\partial J(\theta)}{\partial w_{ik}^2} = \frac{\partial J(\theta)}{\partial g_i^2}\frac{\partial g_i^2}{\partial X_k^1} = \frac{\partial J(\theta)}{\partial g^3} X_k^1$$

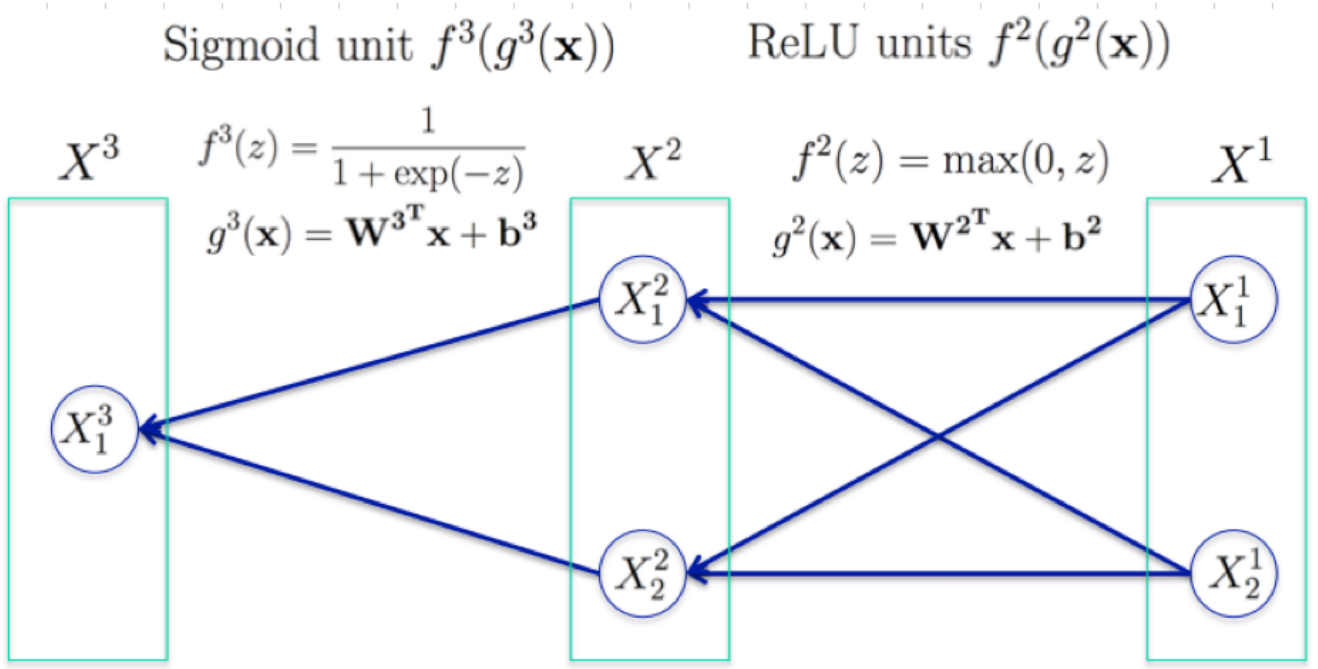
Back propagation



Ytrue	1		
dJ/dX3	-1.89		
dJ/dg3	-0.47		
dJ/dW3T	-0.09	0.00	-0.47
dJ/dX2	-0.05		
	0.04		
dJ/dg2	-0.05		
	0.00		
dJ/dW2T	-0.05	0.00	-0.05
	0.00	0.00	0.00

X^3	$g^3(X^2)$	W^{3T}		b^3	X^2	$g^2(X^1)$	W^{2T}		b^2	X^1
0.53	0.12	0.10	-0.09	0.10	0.20	0.20	0.10	-0.10	0.10	1
					0.00	-0.15	-0.20	0.10	0.05	0
					1					***1***

update each parameter according to $\theta_i \leftarrow \quad \theta_i - \eta \frac{\partial J(\theta)}{\partial \theta_i}$



Ytrue	1		
dJ/dX3	-1.89		
dJ/dg3	-0.47		
dJ/dW3T	-0.09	0.00	-0.47
dJ/dX2	-0.05		
	0.04		
dJ/dg2	-0.05		
	0.00		
dJ/dW2T	-0.05	0.00	-0.05
	0.00	0.00	0.00

X^3	$g^3(X^2)$	W^{3T}		b^3	X^2	$g^2(X^1)$	W^{2T}		b^2	X^1
0.53	0.12	0.10	-0.09	0.10	0.20	0.20	0.10	-0.10	0.10	1
					0.00	-0.15	-0.20	0.10	0.05	0
					1					***1***

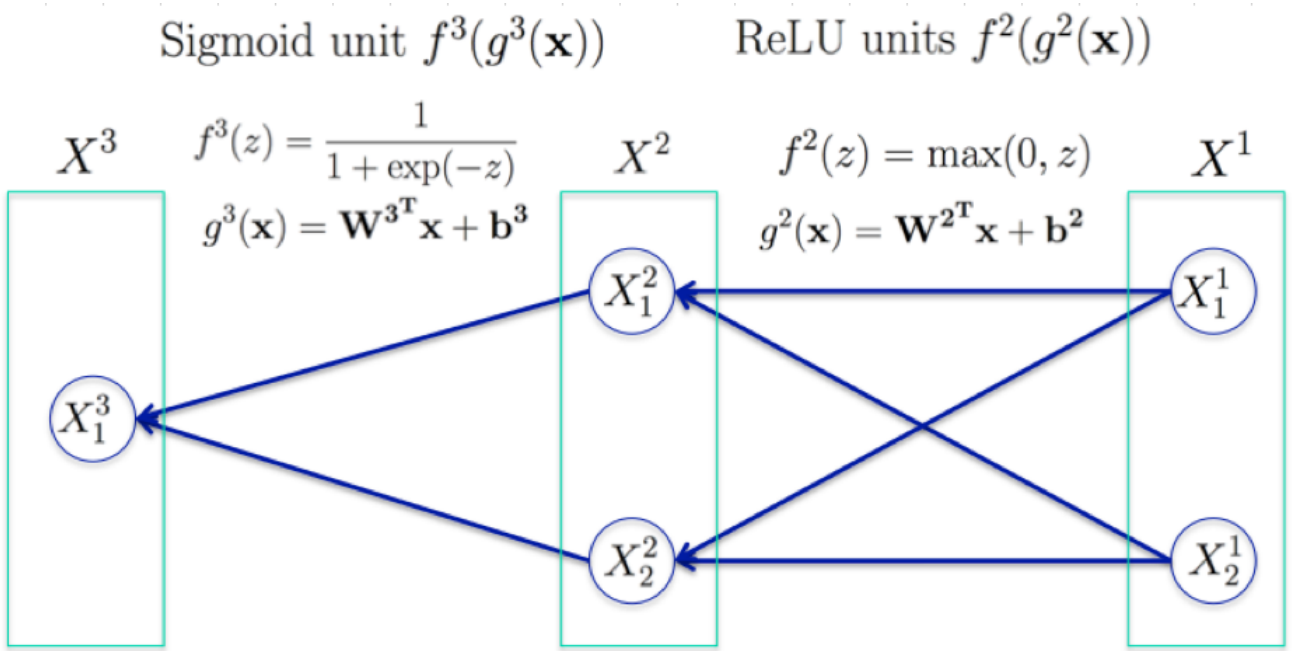
update each parameter according to $\theta_i \leftarrow \quad \theta_i - \eta \frac{\partial J(\theta)}{\partial \theta_i}$

using $\eta = 1$:

X^3	$g^3(X^2)$	W^{3T}		b^3	X^2	$g^2(X^1)$	W^{2T}		b^2	X^1
0.65	0.63	0.19	-0.09	0.57	0.30	0.30	0.15	-0.10	0.15	1
					0.00	-0.15	-0.20	0.10	0.05	0
					1					***1***

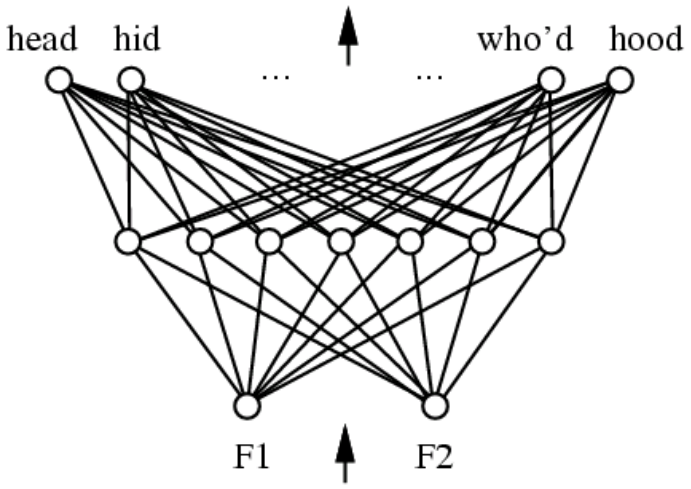
Next training example.. next stochastic gradient step…

Ytrue	1		
dJ/dX3	-1.53		
dJ/dg3	-0.35		
dJ/dW3T	-0.10	0.00	-0.35
dJ/dX2	-0.07		
	0.03		
dJ/dg2	-0.07		
	0.00		
dJ/dW2T	-0.07	0.00	-0.07
	0.00	0.00	0.00



X^3	$g^3(X^2)$	W^{3T}		b^3	X^2	$g^2(X^1)$	W^{2T}		b^2	X^1
0.65	0.63	0.19	-0.09	0.57	0.30	0.30	0.15	-0.10	0.15	1
					0.00	-0.15	-0.20	0.10	0.05	0
					1					*1*

Backpropagation Algorithm for sigmoid netwk, minimizing $\Sigma_d (t_d - o_d)^2$



Initialize all weights to small random numbers.
Until satisfied, Do

- For each training example, Do
  1. Input the training example to the network and compute the network outputs
  2. For each output unit k

$$\delta_k \leftarrow o_k(1 - o_k)(t_k - o_k)$$

  3. For each hidden unit h

$$\delta_h \leftarrow o_h(1 - o_h) \sum_{k \in outputs} w_{h,k}\delta_k$$

  4. Update each network weight $w_{i,j}$

$$w_{i,j} \leftarrow w_{i,j} + \Delta w_{i,j}$$

  where

$$\Delta w_{i,j} = \eta \delta_j x_i$$

x_d = input
o_i = observed i^{th} unit output
t_i = target output
w_{ij} = wt from i to j
δ_i= error term backpropagated to unit k:
dJ(θ) / $dg^k(X^{k-1})$

Many modifications to gradient descent

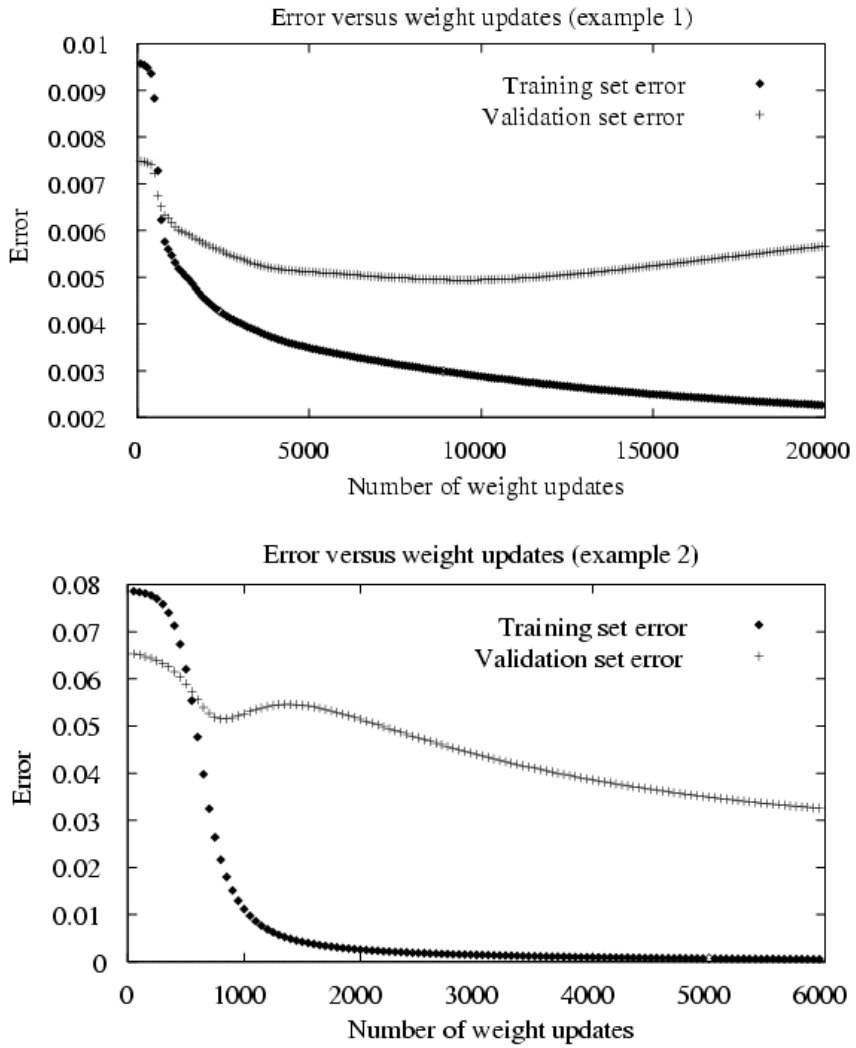
- Stochastic vs. Batch gradient descent (and mini-batches)
- Momentum
- Weight decay (MAP estimate with zero-mean prior)
- Gradient clipping
- Batch normalization
- Dropout
- Adagrad
- Adam
- …. no end in sight

See ML Department course on Optimization Methods

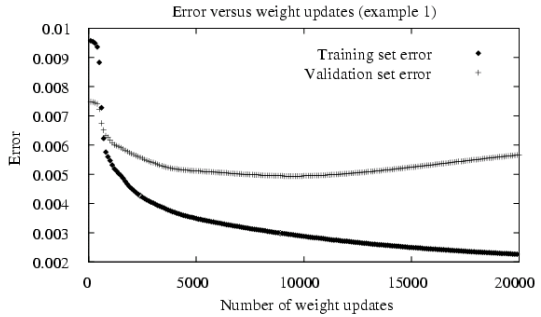
Gradient Descent and Backpropagation

- Updates every network parameter simultaneously, each iteration
- Easily generalized to arbitrary directed acyclic graphs
- Finds local minimum in J(θ), not necessarily global min
- Minimizes J(θ) over training examples, not necessarily future…
- Training can require hours, days, weeks, GPUs
- Applying network after training is relatively very fast

Overfitting in ANNs



Dealing with Overfitting



Our learning algorithm involves a parameter

n=number of gradient descent iterations

How do we choose n to optimize future error or loss?

- Separate available data into <u>training</u> and <u>validation</u> set
- Use <u>training</u> to perform gradient descent
- n ← number of iterations that optimizes <u>validation</u> set error

→ *gives unbiased estimate of optimal n*

(but still an optimistically <u>biased</u> estimate of true error)

Expressive Capabilities of ANNs

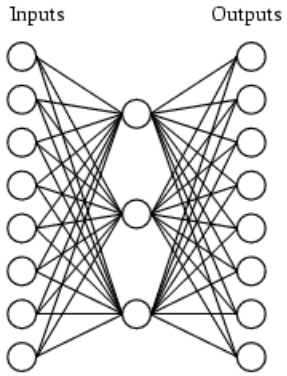
Boolean functions:

- Every boolean function can be represented by network with single hidden layer
- but might require exponential (in number of inputs) hidden units

Continuous functions:

- Every bounded continuous function can be approximated with arbitrarily small error, by network with one hidden layer [Cybenko 1989; Hornik et al. 1989]
- Any function can be approximated to arbitrary accuracy by a network with two hidden layers [Cybenko 1988].

Learning Hidden Layer Representations



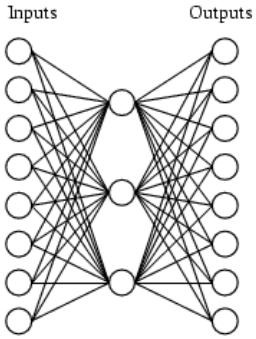
A target function:

Input		Output
10000000	$\rightarrow$	10000000
01000000	$\rightarrow$	01000000
00100000	$\rightarrow$	00100000
00010000	$\rightarrow$	00010000
00001000	$\rightarrow$	00001000
00000100	$\rightarrow$	00000100
00000010	$\rightarrow$	00000010
00000001	$\rightarrow$	00000001

Can this be learned??

Learning Hidden Layer Representations

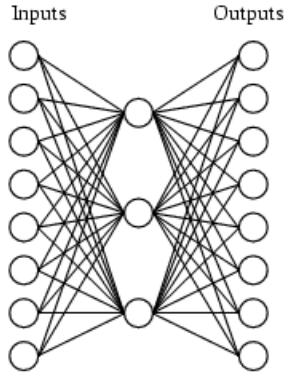
A network:

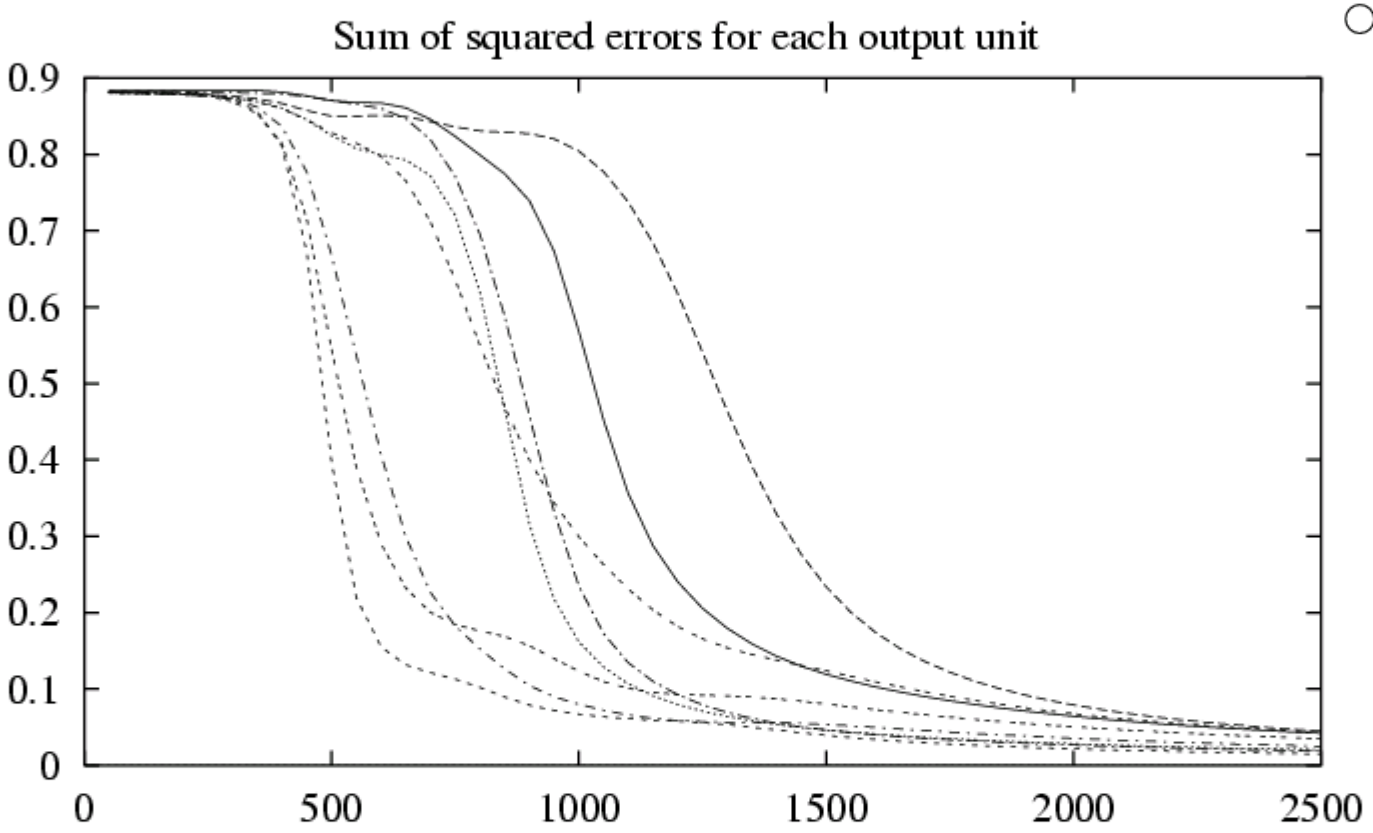


Learned hidden layer representation:

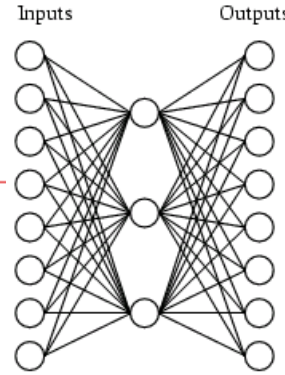
Input		Hidden Values				Output
10000000	$\rightarrow$	.89	.04	.08	$\rightarrow$	10000000
01000000	$\rightarrow$	.01	.11	.88	$\rightarrow$	01000000
00100000	$\rightarrow$	.01	.97	.27	$\rightarrow$	00100000
00010000	$\rightarrow$	.99	.97	.71	$\rightarrow$	00010000
00001000	$\rightarrow$	.03	.05	.02	$\rightarrow$	00001000
00000100	$\rightarrow$	.22	.99	.99	$\rightarrow$	00000100
00000010	$\rightarrow$	.80	.01	.98	$\rightarrow$	00000010
00000001	$\rightarrow$	.60	.94	.01	$\rightarrow$	00000001

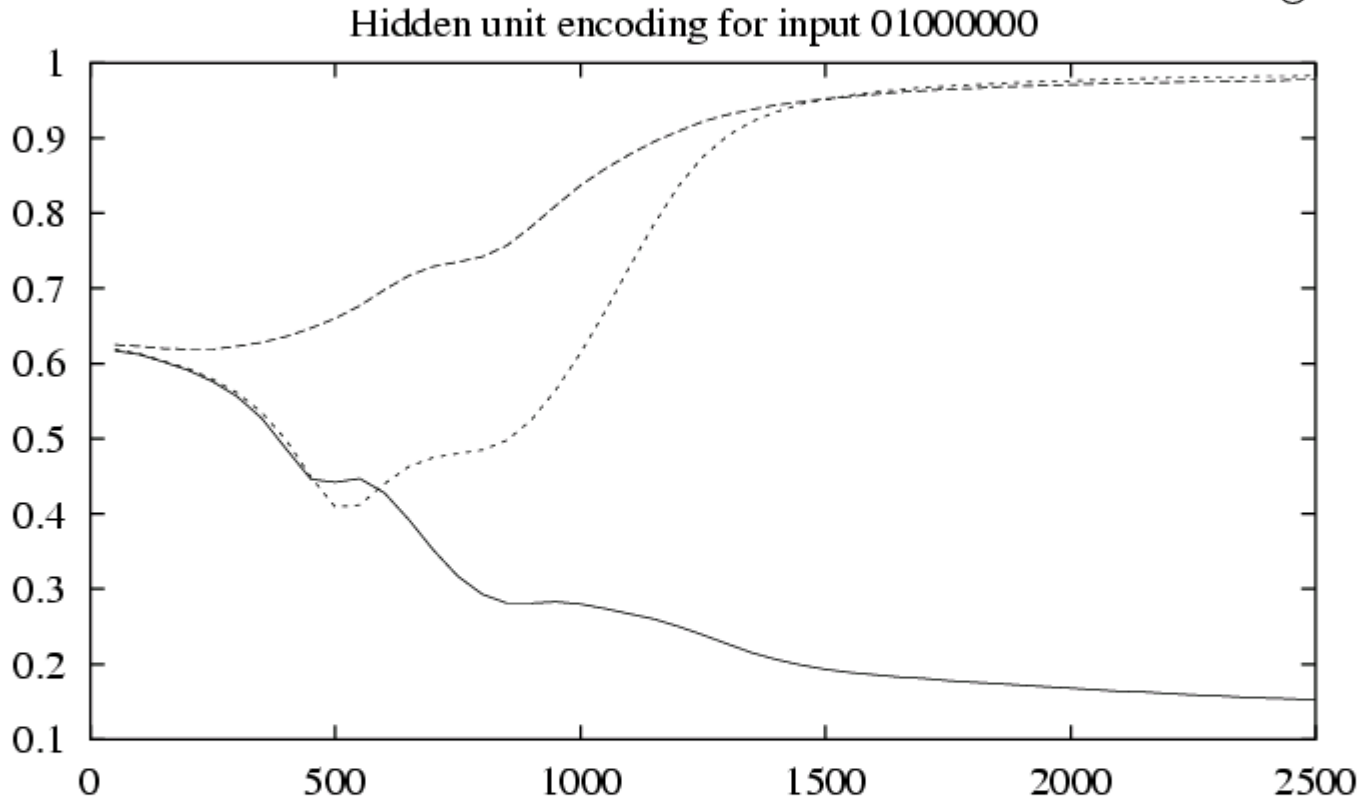
Training


Inputs
Outputs

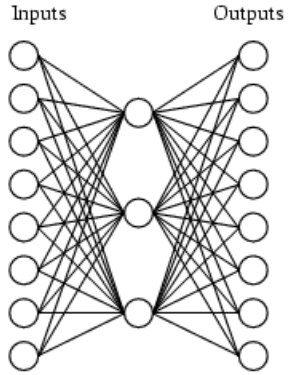

Sum of squared errors for each output unit
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
0
500
1000
1500
2000
2500

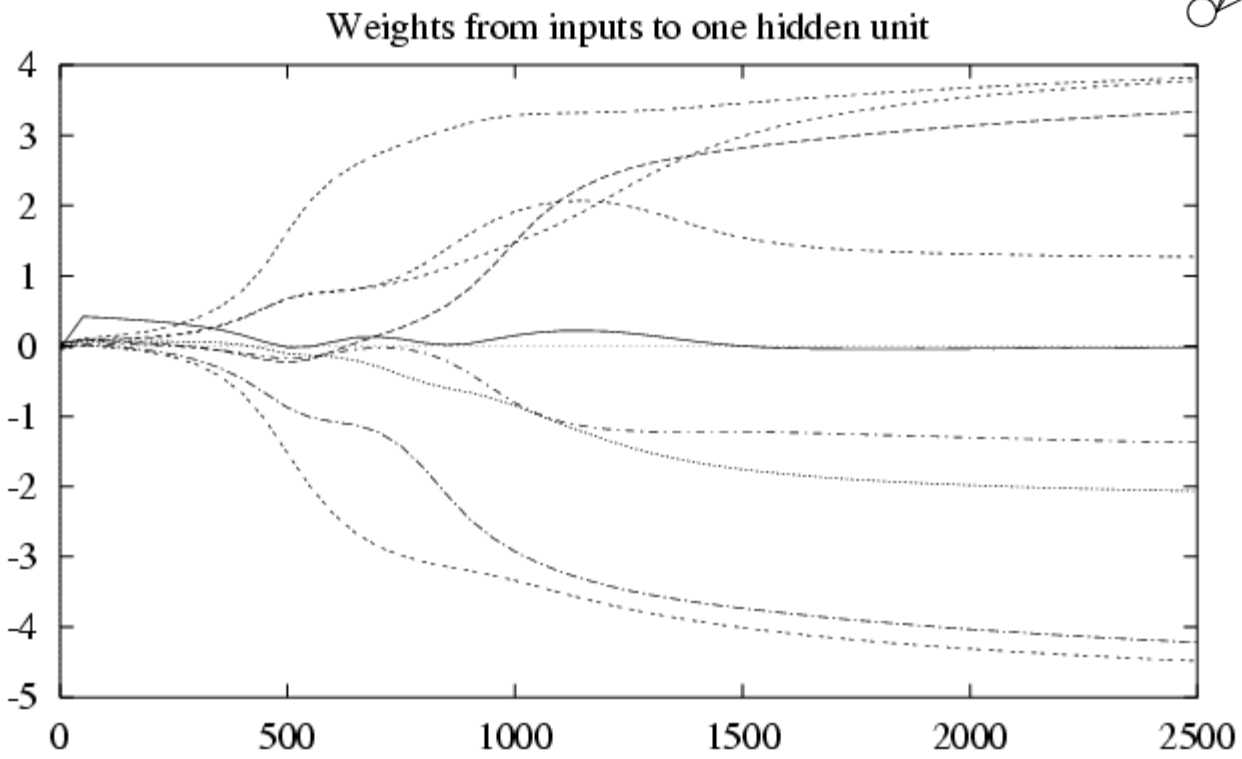
Training


Inputs
Outputs

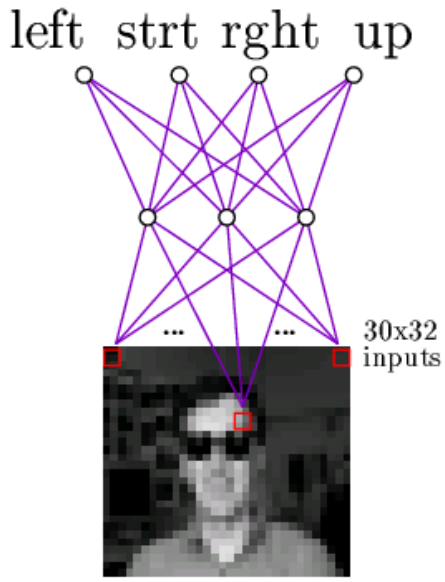

Hidden unit encoding for input 01000000
1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
500
1000
1500
2000
2500

Training


Inputs
Outputs


Weights from inputs to one hidden unit
4
3
2
1
0
-1
-2
-3
-4
-5
0
500
1000
1500
2000
2500

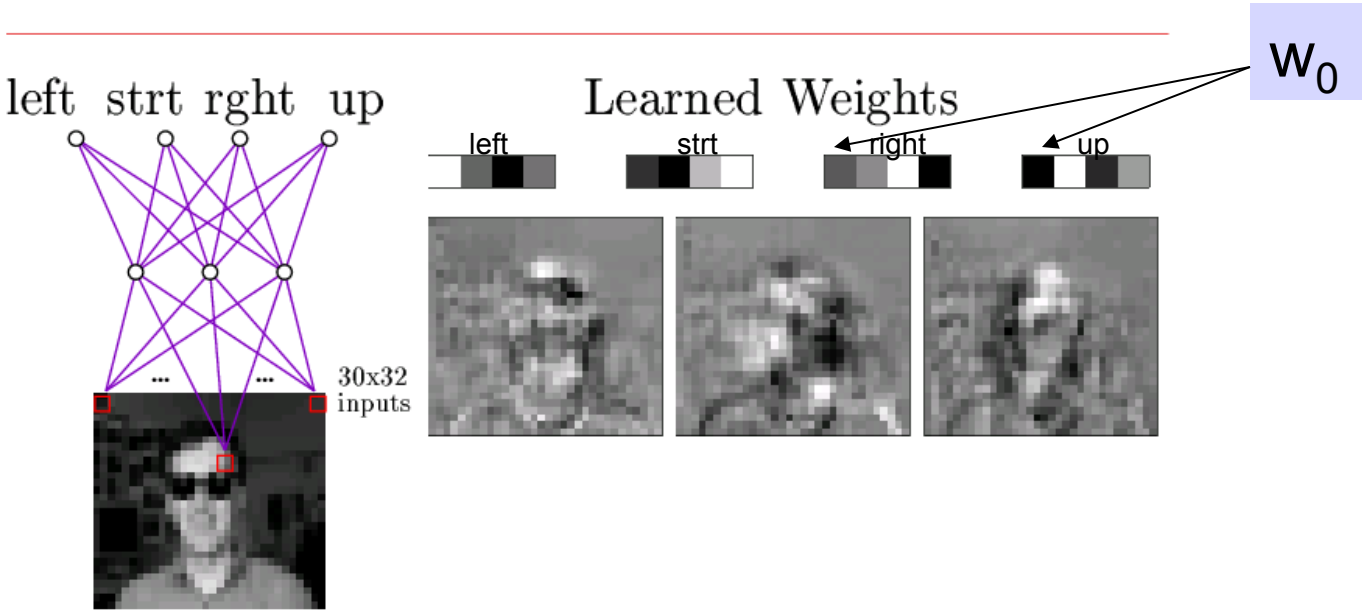
Neural Nets for Face Recognition



Typical input images

90% accurate learning head pose, and recognizing 1-of-20 faces

Learned Hidden Unit Weights

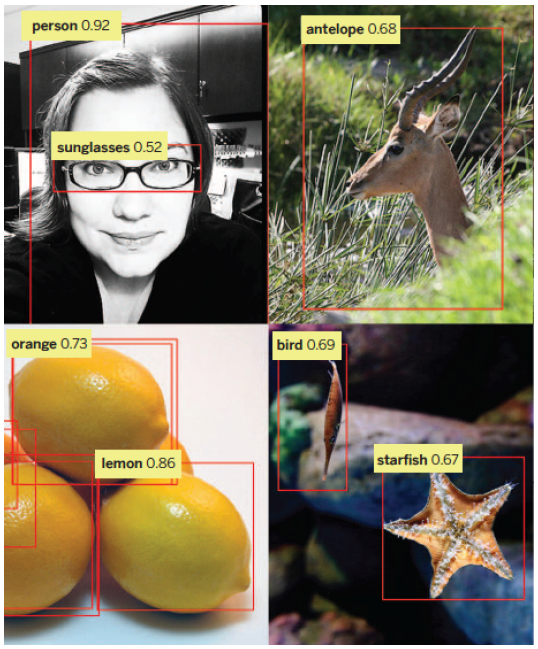


Typical input images

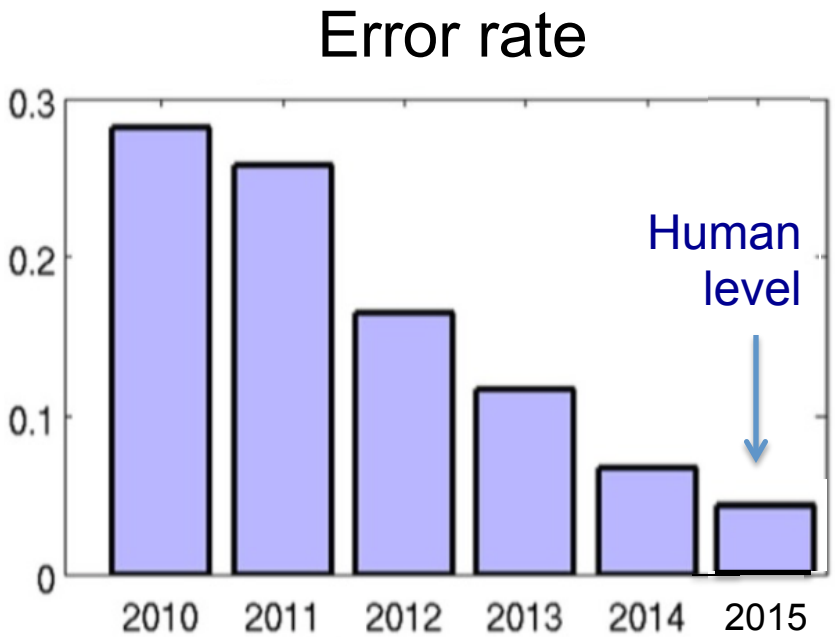
http://www.cs.cmu.edu/~tom/faces.html

Convolutional Neural Nets

Computer Vision



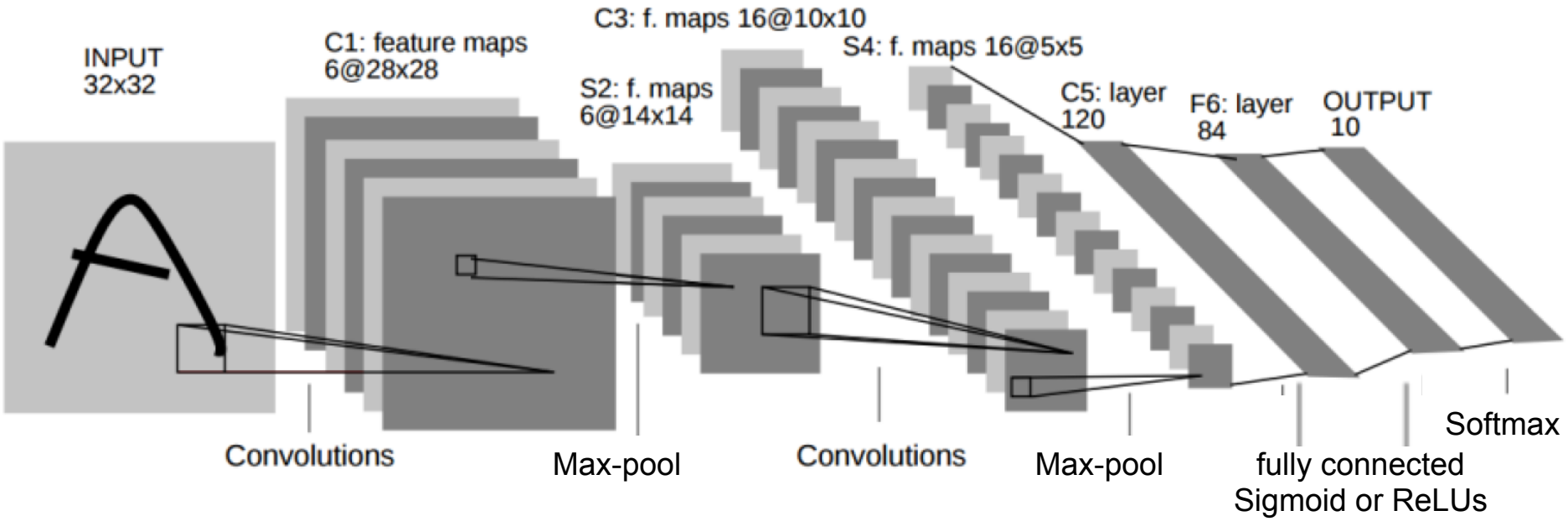
Imagenet Visual Recognition



Challenges:

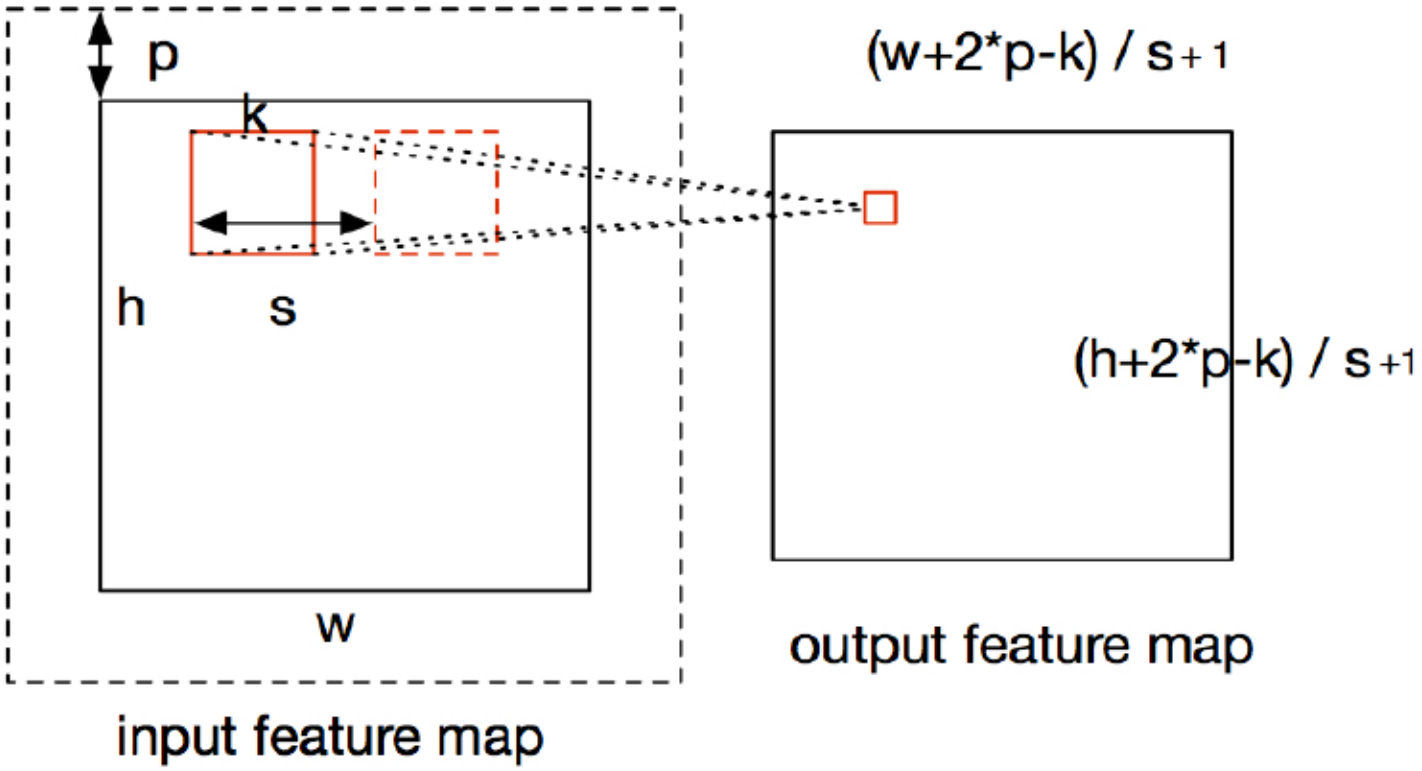
- invariance to translation, lighting, scaling, …
- manual engineering of low-level image features
- ...

A Convolutional Neural Net for Handwritten Digit recognition: LeNet



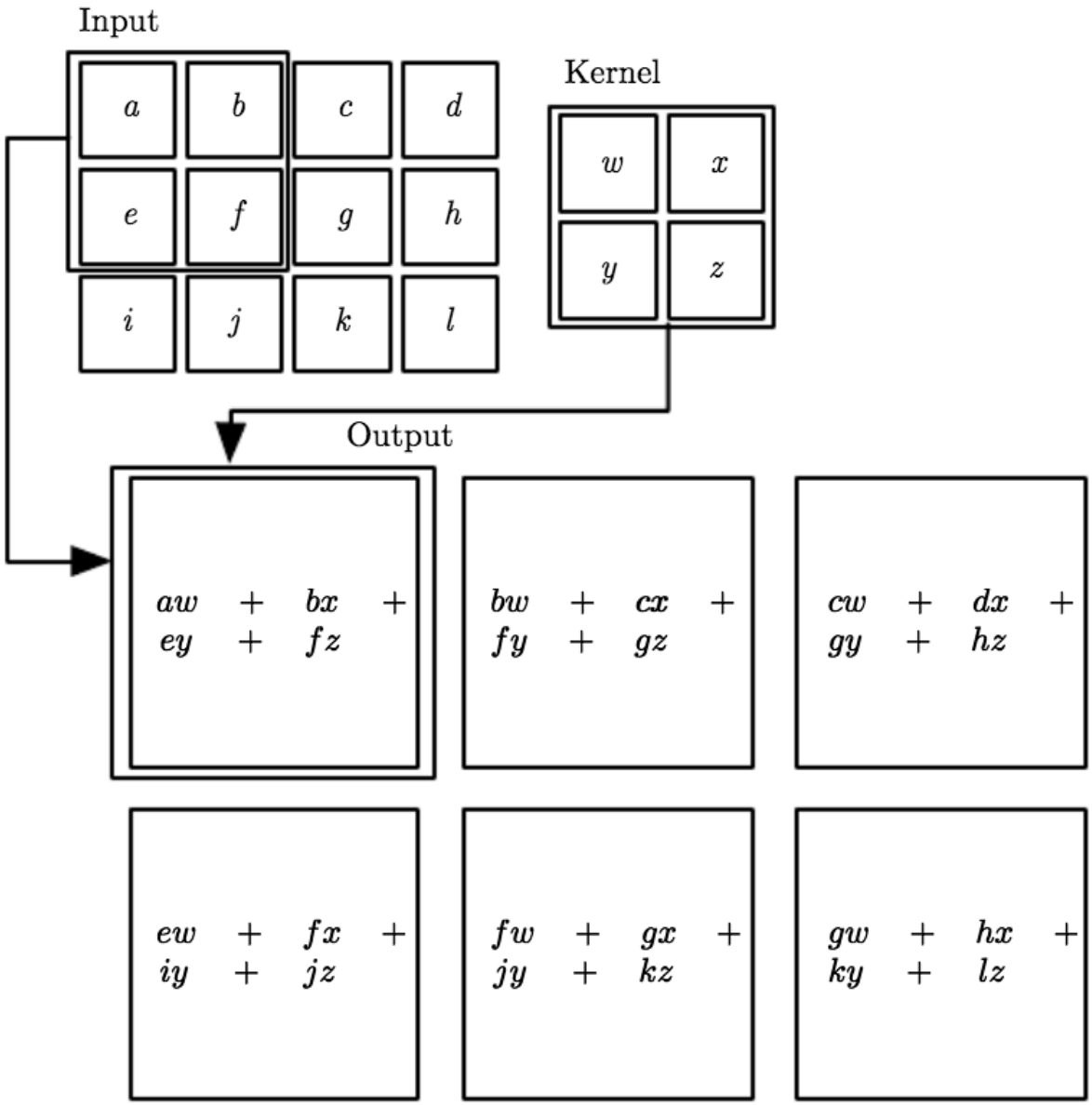
Convolution layer

p = padding
s = stride



$$S(i,j) = (I * K)(i,j) = \sum_m \sum_n I(i+m, j+n) K(m,n)$$

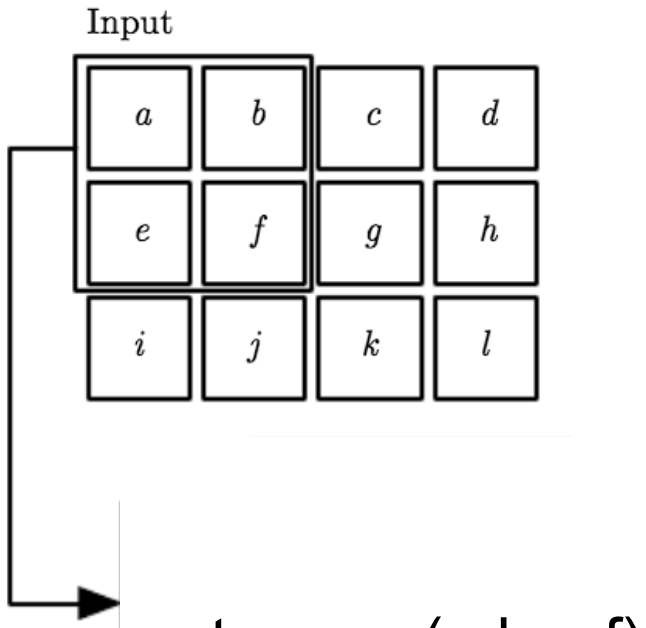
Convolution



$$S(i,j) = (I * K)(i,j) = \sum_m \sum_n I(i+m, j+n) K(m,n)$$

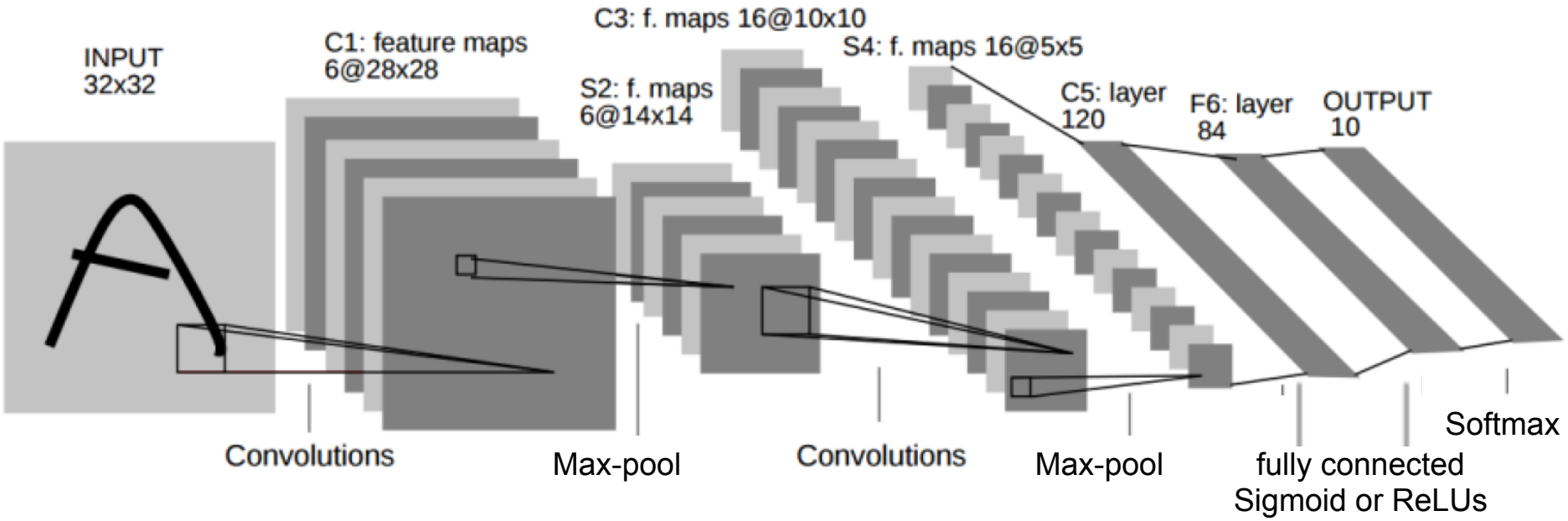
[from Goodfellow et al.]

Maxpool



[from Goodfellow et al.]

A Convolutional Neural Net for Handwritten Digit recognition: LeNet



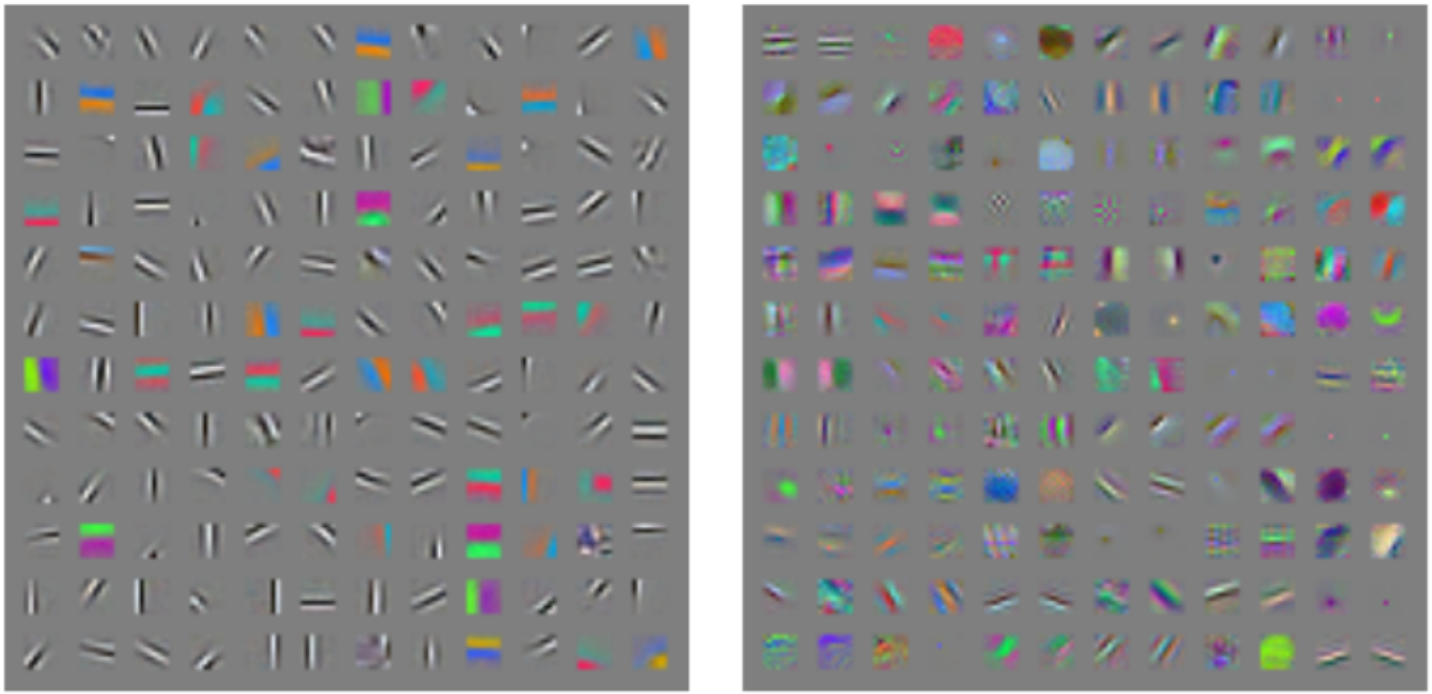


Figure 9.19: Many machine learning algorithms learn features that detect edges or specific colors of edges when applied to natural images. These feature detectors are reminiscent of the Gabor functions known to be present in primary visual cortex. *(Left)*Weights learned by an unsupervised learning algorithm (spike and slab sparse coding) applied to small image patches. *(Right)*Convolution kernels learned by the first layer of a fully supervised convolutional maxout network. Neighboring pairs of filters drive the same maxout unit.

[from Goodfellow et al.]

What you should know:

- Backpropagation algorithm
  - training network with gradient descent
  - how to derive gradient for simple cost functions, units
  - relationship to logistic regression
- Neural nets learn internal representations
- Convolutional neural networks
  - convolution operation
  - see new Homework

Many types of parameterized units

- Sigmoid activation units
- tanh activation units $\tanh(x) = \frac{(e^{2x}-1)}{(e^{2x}+1)}$
- ReLU
- Leaky ReLU (fixed non-zero slope for input<0)
- Parametric ReLU (trainable slope for input<0)
- Max Pool
- Inner Product
- GRU's
- LSTM's
- Matrix multiply
- …. no end in sight

Any unit h(X; W) that is differentiable w.r.t. X and W