

Machine Learning

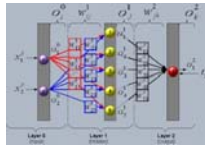
10-701/15-781, Fall 2011

Artificial Neural Networks And Deep Learning

Eric Xing

Lecture 5, September 26, 2011

Reading: Chap. 5 CB



© Eric Xing @ CMU, 2006-2011

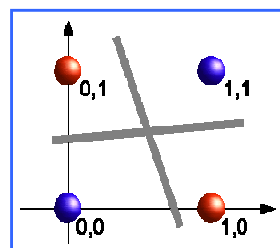
1

Learning highly non-linear functions

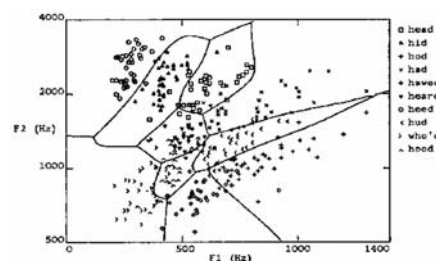
$$f: X \rightarrow Y$$

- f might be non-linear function
- X (vector of) continuous and/or discrete vars
- Y (vector of) continuous and/or discrete vars

The XOR gate



Speech recognition

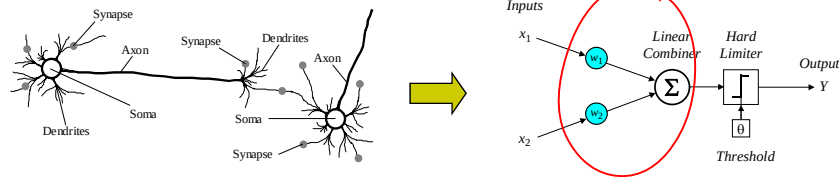


© Eric Xing @ CMU, 2006-2011

2

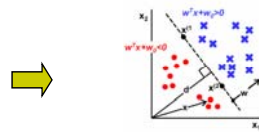
Perceptron and Neural Nets

- From biological neuron to artificial neuron (perceptron)



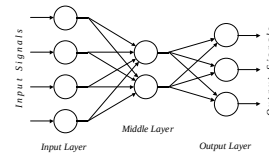
- Activation function

$$X = \sum_{i=1}^n x_i w_i \quad Y = \begin{cases} +1, & \text{if } X \geq \omega_0 \\ -1, & \text{if } X < \omega_0 \end{cases}$$



- Artificial neuron networks

- supervised learning
- gradient descent



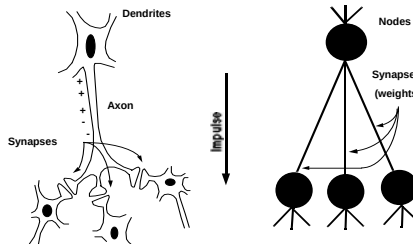
© Eric Xing @ CMU, 2006-2011

3

Connectionist Models

- Consider humans:

- Neuron switching time
~ 0.001 second
- Number of neurons
~ 10^{10}
- Connections per neuron
~ 10^4 - 10^5
- Scene recognition time
~ 0.1 second
- 100 inference steps doesn't seem like enough
→ much parallel computation



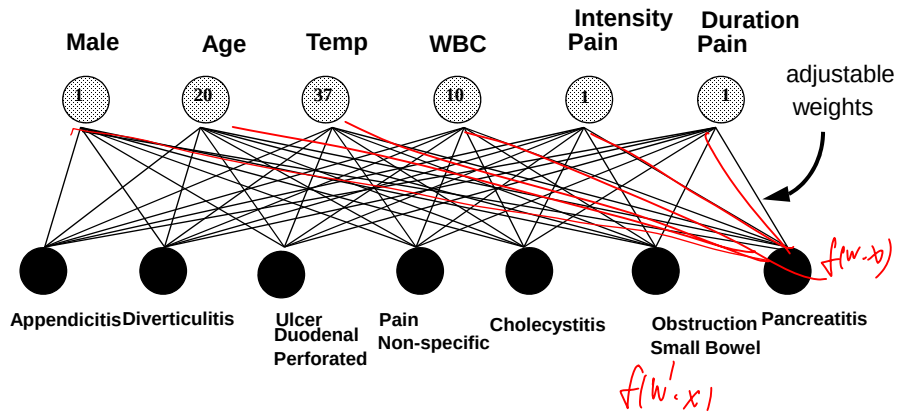
- Properties of artificial neural nets (ANN)

- Many neuron-like threshold switching units
- Many weighted interconnections among units
- Highly parallel, distributed processes

© Eric Xing @ CMU, 2006-2011

4

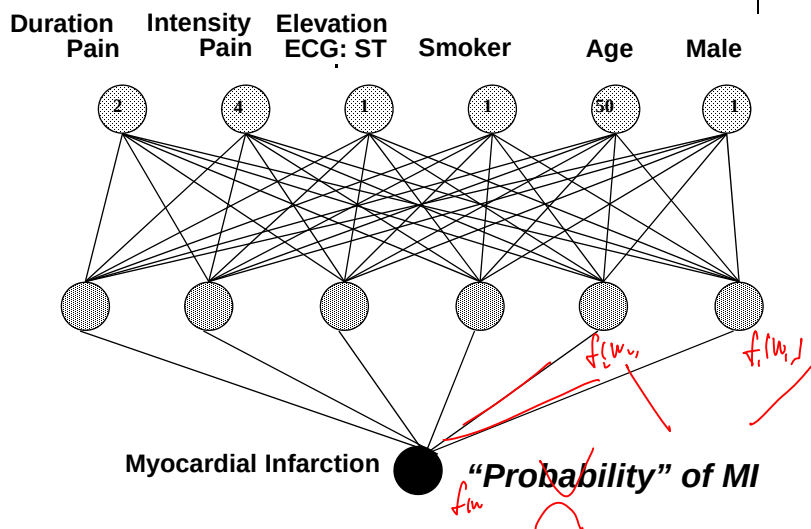
Abdominal Pain Perceptron



© Eric Xing @ CMU, 2006-2011

5

Myocardial Infarction Network



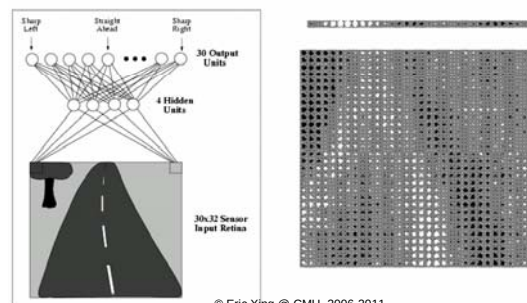
© Eric Xing @ CMU, 2006-2011

6

The "Driver" Network



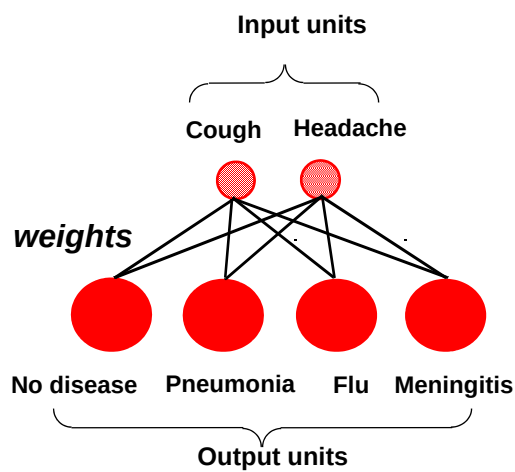
ALVINN
[Pomerleau 1993]



© Eric Xing @ CMU, 2006-2011

7

Perceptrons



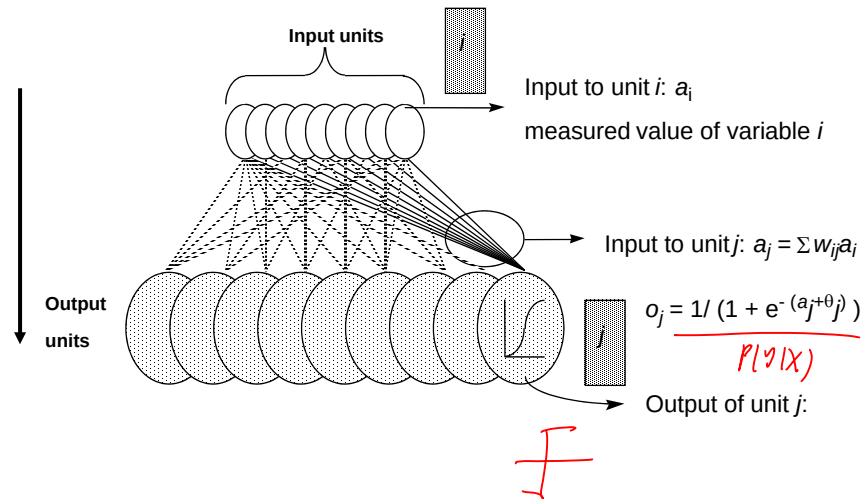
Δ rule
change weights to
decrease the error

$$- \frac{\text{what we got} - \text{what we wanted}}{\text{error}}$$

© Eric Xing @ CMU, 2006-2011

8

Perceptrons



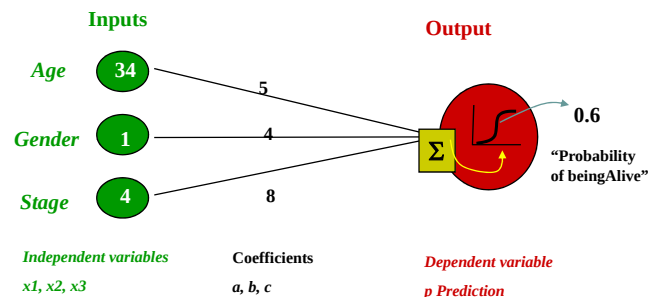
© Eric Xing @ CMU, 2006-2011

9

Jargon Pseudo-Correspondence

- Independent variable = input variable
- Dependent variable = output variable
- Coefficients = "weights"
- Estimates = "targets"

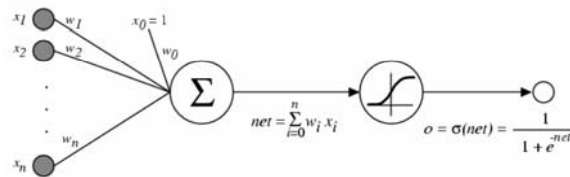
Logistic Regression Model (the sigmoid unit)



© Eric Xing @ CMU, 2006-2011

10

The perceptron learning algorithm



- Recall the nice property of sigmoid function $\frac{d\sigma}{dt} = \sigma(1 - \sigma)$
- Consider regression problem $f: X \rightarrow Y$, for scalar Y : $y = f(x) + \epsilon$
- We used to maximize the conditional data likelihood

$$\vec{w} = \arg \max_{\vec{w}} \ln \prod_i P(y_i | x_i; \vec{w})$$

- Here ...

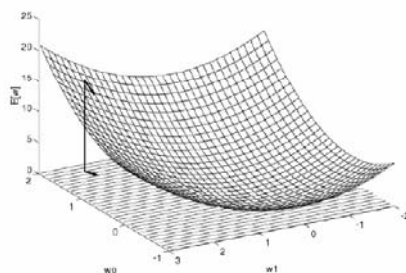
$$\vec{w} = \arg \min_{\vec{w}} \sum_i \frac{1}{2} (y_i - \hat{f}(x_i; \vec{w}))^2$$

© Eric Xing @ CMU, 2006-2011

11

Gradient Descent

x_d = input
 t_d = target output
 o_d = observed unit output
 w_i = weight i



$$\begin{aligned} \frac{\partial E[\vec{w}]}{\partial w_j} &= \frac{\partial}{\partial w_j} \frac{1}{2} \sum (t_d - o_d)^2 \\ &= (t_d - o_d) o_d (1 - o_d) \frac{d \sigma}{d \text{net}} \end{aligned}$$

Gradient

$$\nabla E[\vec{w}] = \left[\frac{\partial E}{\partial w_0}, \frac{\partial E}{\partial w_1}, \dots, \frac{\partial E}{\partial w_n} \right]$$

Training rule:

$$\Delta \vec{w} = -\eta \nabla E[\vec{w}]$$

i.e.,

$$\Delta w_i = -\eta \frac{\partial E}{\partial w_i}$$

© Eric Xing @ CMU, 2006-2011

12

The perceptron learning rules

x_d = input
 t_d = target output
 o_d = observed unit
 output
 w_i = weight i

$$\begin{aligned}
 \frac{\partial E_D[\vec{w}]}{\partial w_j} &= \frac{\partial}{\partial w_i} \frac{1}{2} \sum_d (t_d - o_d)^2 \\
 &= \frac{1}{2} \sum_d 2(t_d - o_d) \frac{\partial}{\partial w_i} (t_d - o_d) \\
 &= \sum_d (t_d - o_d) \left(-\frac{\partial o_d}{\partial w_i} \right) \\
 &= - \sum_d (t_d - o_d) \frac{\partial o_d}{\partial \text{net}_i} \frac{\partial \text{net}_d}{\partial w_i} \\
 &= - \sum_d (t_d - o_d) o_d (1 - o_d) x_d^i
 \end{aligned}$$

Batch mode:

Do until converge:

1. compute gradient $\nabla E_D[\vec{w}]$
2. $\vec{w} = \vec{w} - \eta \nabla E_D[\vec{w}]$

Incremental mode:

Do until converge:

- For each training example d in D
 1. compute gradient $\nabla E_d[\vec{w}]$
 2. $\vec{w} = \vec{w} - \eta \nabla E_d[\vec{w}]$

where

$$\nabla E_d[\vec{w}] = -(t_d - o_d) o_d (1 - o_d) \vec{x}_d$$

© Eric Xing @ CMU, 2006-2011

13

MLE vs MAP



- Maximum conditional likelihood estimate

$$\vec{w} = \arg \max_{\vec{w}} \ln \prod_i P(y_i | x_i; \vec{w})$$

$$\vec{w} \leftarrow \vec{w} + \eta \sum_d (t_d - o_d) o_d (1 - o_d) \vec{x}_d$$

- Maximum a posteriori estimate

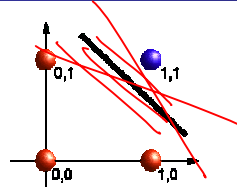
$$\vec{w} = \arg \max_{\vec{w}} \ln p(\vec{w}) \prod_i P(y_i | x_i; \vec{w})$$

$$\vec{w} \leftarrow \vec{w} + \eta \left(\sum_d (t_d - o_d) o_d (1 - o_d) \vec{x}_d - \lambda \vec{w} \right)$$

© Eric Xing @ CMU, 2006-2011

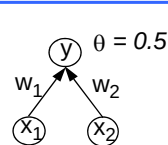
14

What decision surface does a perceptron define?



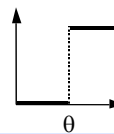
NAND

x	y	Z (color)
0	0	1
0	1	1
1	0	1
1	1	0



$$f(x_1w_1 + x_2w_2) = y$$

$$\begin{aligned} f(0w_1 + 0w_2) &= 1 \\ f(0w_1 + 1w_2) &= 1 \\ f(1w_1 + 0w_2) &= 1 \\ f(1w_1 + 1w_2) &= 0 \end{aligned}$$



$$f(a) = \begin{cases} 1, & \text{for } a > \theta \\ 0, & \text{for } a \leq \theta \end{cases}$$

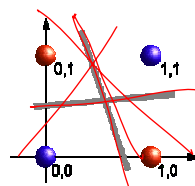
some possible values for w_1 and w_2

w_1	w_2
0.20	0.35
0.20	0.40
0.25	0.30
0.40	0.20

© Eric Xing @ CMU, 2006-2011

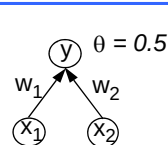
15

What decision surface does a perceptron define?



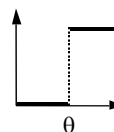
NAND

x	y	Z (color)
0	0	0
0	1	1
1	0	1
1	1	0



$$f(x_1w_1 + x_2w_2) = y$$

$$\begin{aligned} f(0w_1 + 0w_2) &= 0 \\ f(0w_1 + 1w_2) &= 1 \\ f(1w_1 + 0w_2) &= 1 \\ f(1w_1 + 1w_2) &= 0 \end{aligned}$$



$$f(a) = \begin{cases} 1, & \text{for } a > \theta \\ 0, & \text{for } a \leq \theta \end{cases}$$

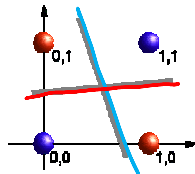
some possible values for w_1 and w_2

w_1	w_2

© Eric Xing @ CMU, 2006-2011

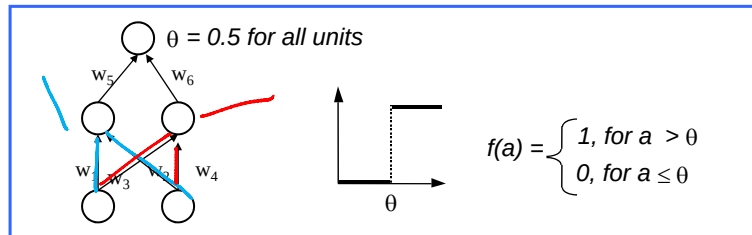
16

What decision surface does a perceptron define?



NAND

x	y	Z (color)
0	0	0
0	1	1
1	0	1
1	1	0



a possible set of values for $(w_1, w_2, w_3, w_4, w_5, w_6)$:
 $(0.6, -0.6, -0.7, 0.8, 1, 1)$

© Eric Xing @ CMU, 2006-2011

17

Non Linear Separation

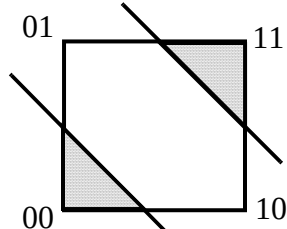


Meningitis

No cough
Headache

Flu

Cough
Headache



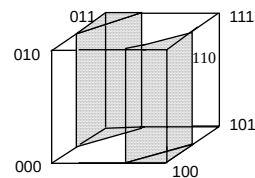
■ No treatment
□ Treatment

No disease

No cough
No headache

Pneumonia

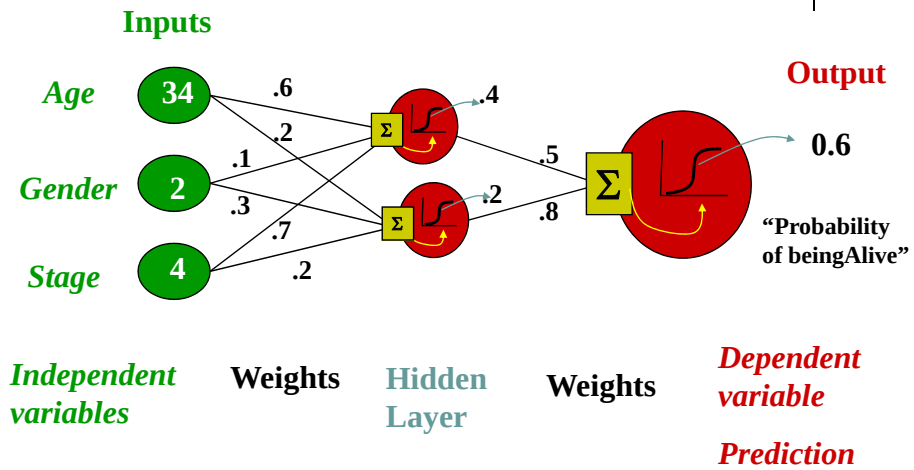
Cough
No headache



© Eric Xing @ CMU, 2006-2011

18

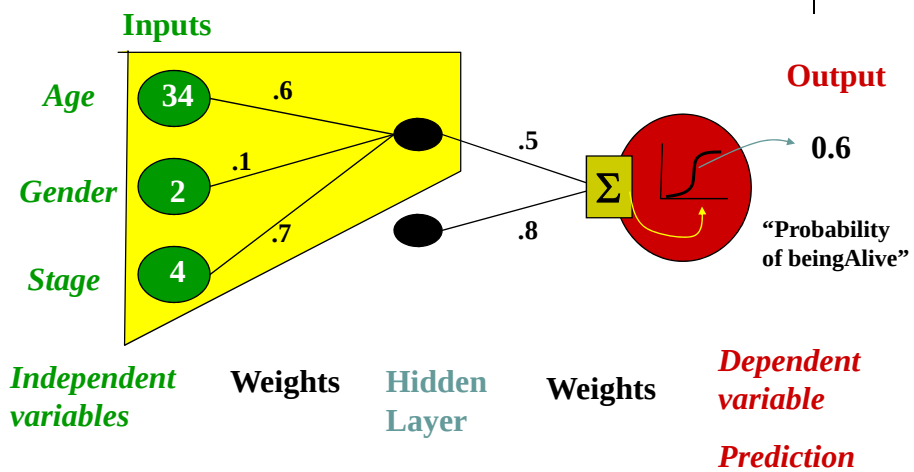
Neural Network Model



© Eric Xing @ CMU, 2006-2011

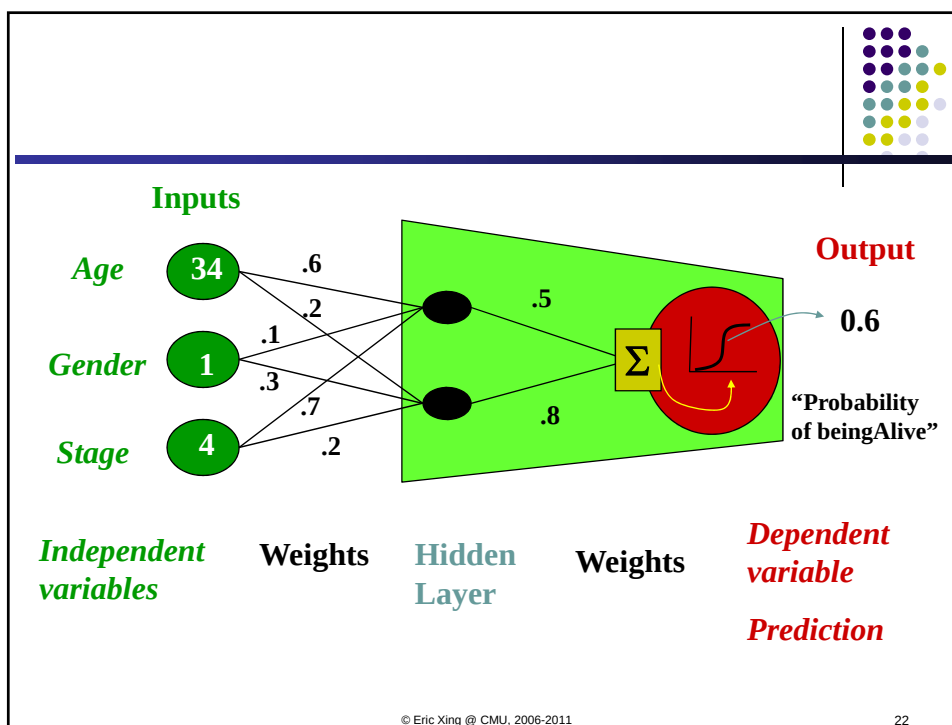
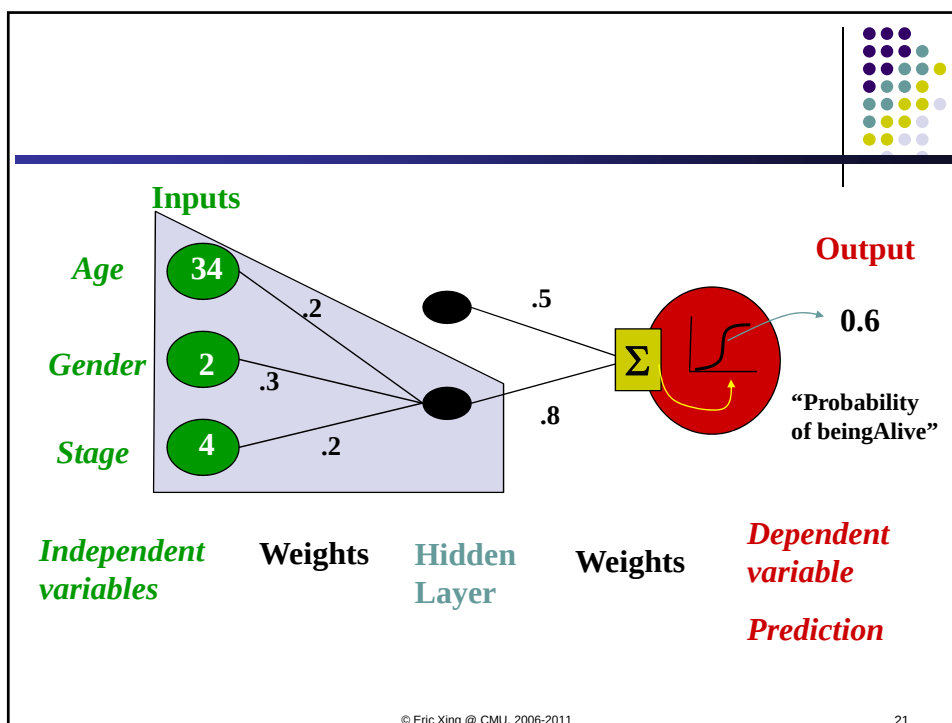
19

"Combined logistic models"

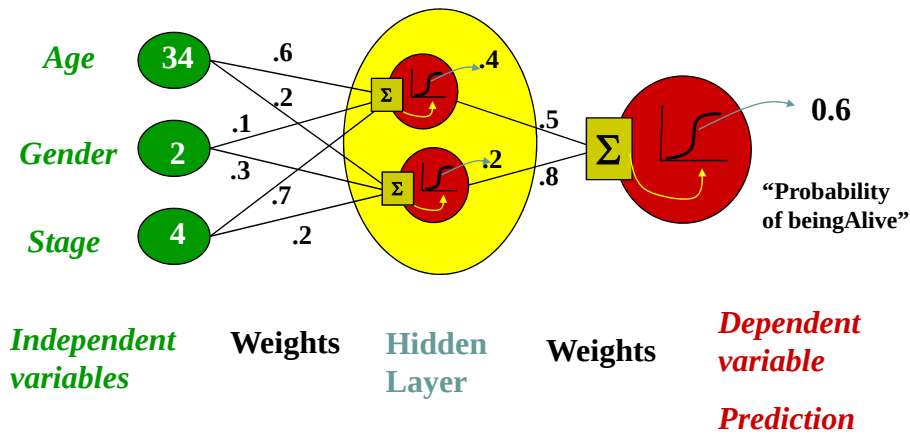


© Eric Xing @ CMU, 2006-2011

20



Not really, no target for hidden units...

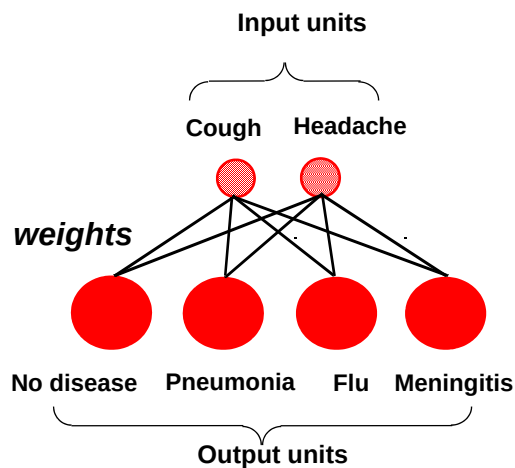


© Eric Xing @ CMU, 2006-2011

23

Recall perceptrons

$$\vec{w} \leftarrow \vec{w} + \eta \sum_d (t_d - o_d) o_d (1 - o_d) \vec{x}_d$$



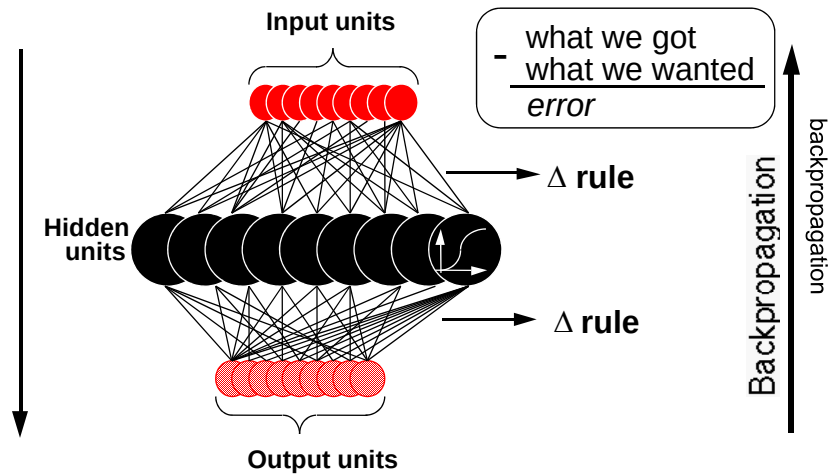
Δ rule
change weights to
decrease the error

what we got
- what we wanted
error

© Eric Xing @ CMU, 2006-2011

24

Hidden Units and Backpropagation



© Eric Xing @ CMU, 2006-2011

25

Backpropagation Algorithm

x_d = input
 t_d = target output
 o_d = observed unit output
 w_i = weight i

- Initialize all weights to small random numbers
- Until convergence, Do

1. Input the training example to the network and compute the network outputs

1. For each output unit k

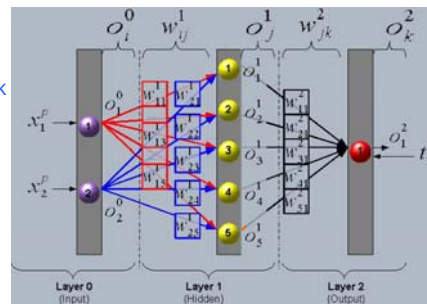
$$\delta_k \leftarrow o_k^2(1 - o_k^2)(t - o_k^2)$$

2. For each hidden unit h

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

3. Update each network weight w_{ij}

$$w_{i,j} \leftarrow w_{i,j} + \Delta w_{i,j} \text{ where } \Delta w_{i,j} = \eta \delta_j x^j$$



$$\min(t - o)^2$$

© Eric Xing @ CMU, 2006-2011

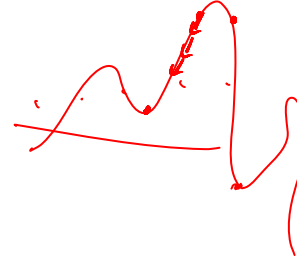
26

More on Backpropatation

- It is doing gradient descent over entire network weight vector
- Easily generalized to arbitrary directed graphs
- Will find a local, not necessarily global error minimum
 - In practice, often works well (can run multiple times)
- Often include weight *momentum* α

$$\Delta w_{i,j}(t) = \eta \delta_j x_{i,j} + \alpha \Delta w_{i,j}(t-1)$$

- Minimizes error over *training* examples
 - Will it generalize well to subsequent testing examples?
- Training can take thousands of iterations, $\rightarrow$ very slow!
- Using network after training is very fast

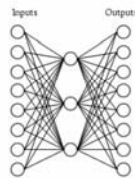


© Eric Xing @ CMU, 2006-2011

27

Learning Hidden Layer Representation

- A network:



- A target function:

Input	Output
10000000	→ 10000000
01000000	→ 01000000
00100000	→ 00100000
00010000	→ 00010000
00001000	→ 00001000
00000100	→ 00000100
00000010	→ 00000010
00000001	→ 00000001

- Can this be learned?

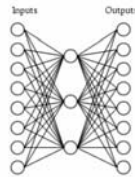
© Eric Xing @ CMU, 2006-2011

28

Learning Hidden Layer Representation



- A network:



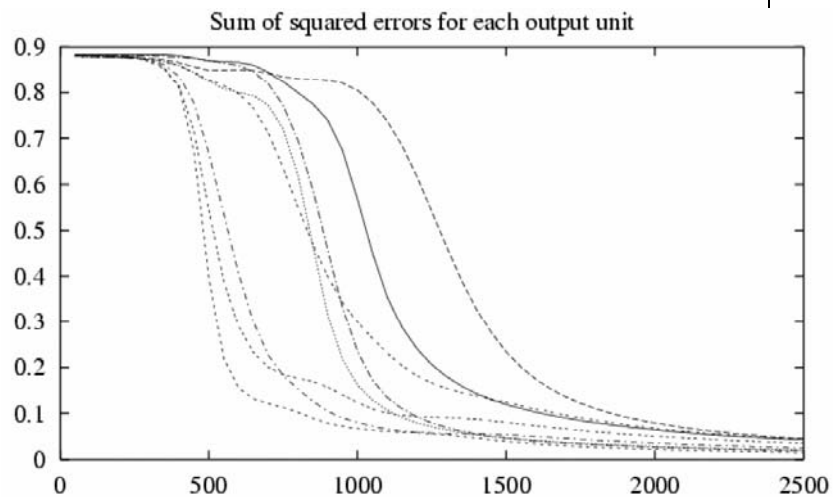
- Learned hidden layer representation:

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

© Eric Xing @ CMU, 2006-2011

29

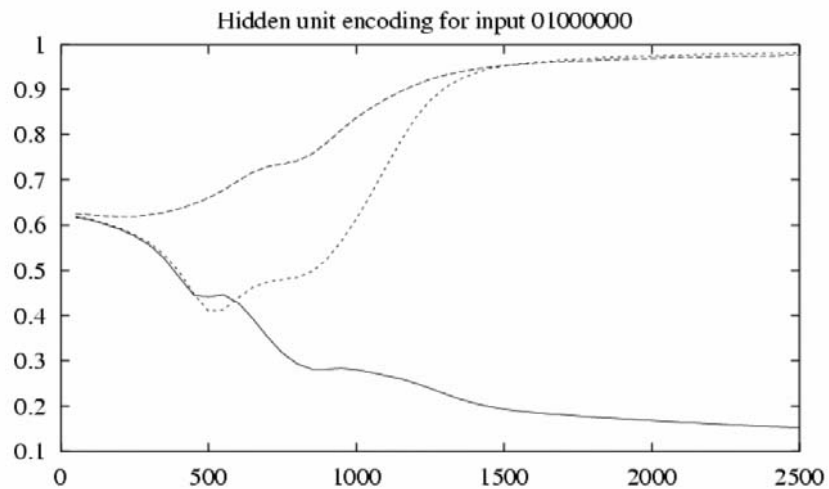
Training



© Eric Xing @ CMU, 2006-2011

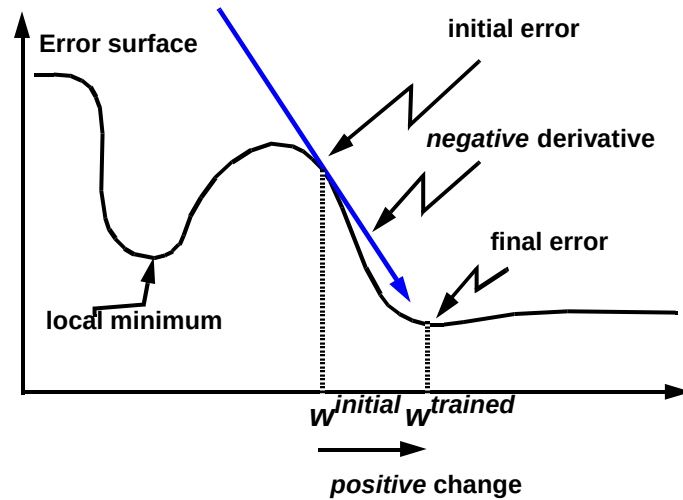
30

Training



31

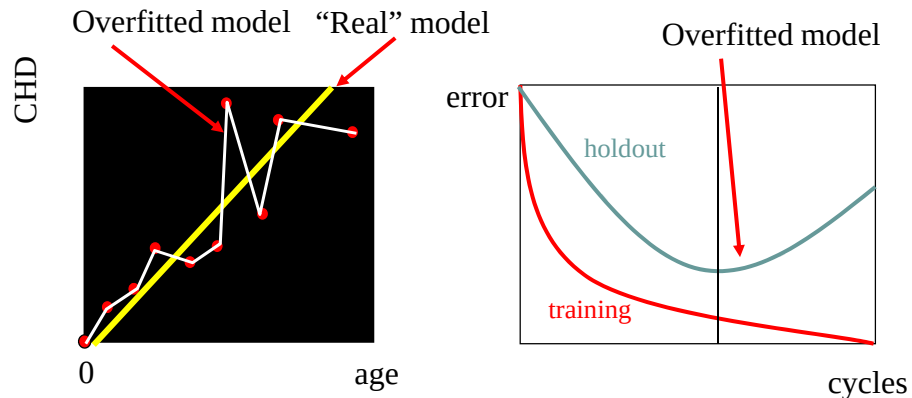
Minimizing the Error



© Eric Xing @ CMU, 2006-2011

32

Overfitting in Neural Nets



© Eric Xing @ CMU, 2006-2011

33

Alternative Error Functions

- Penalize large weights:

$$E[\vec{w}] \equiv \frac{1}{2} \sum_{d \in D} \sum_{k \in \text{outputs}} (t_{k,d} - o_{k,d})^2 + \gamma \sum_{i,j} w_{j,i}^2$$

- Training on target slopes as well as values

$$E[\vec{w}] \equiv \frac{1}{2} \sum_{d \in D} \sum_{k \in \text{outputs}} (t_{k,d} - o_{k,d})^2 + \mu \sum_{j \in \text{inputs}} \left(\frac{\partial t_{k,d}}{\partial x_d^j} - \frac{\partial o_{k,d}}{\partial x_d^j} \right)$$

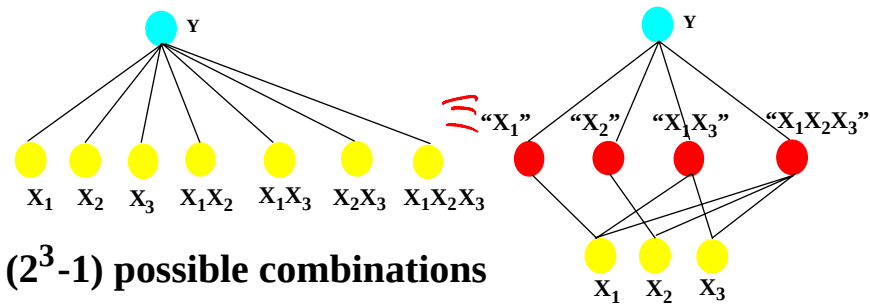
- Tie together weights
 - E.g., in phoneme recognition

$$\frac{t_{k,d}(x_d) - t_{k,d'}(x_{d'})}{x_d - x_{d'}}$$

© Eric Xing @ CMU, 2006-2011

34

Regression vs. Neural Networks



$$Y = a(X_1) + b(X_2) + c(X_3) + d(X_1X_2) + \dots$$

© Eric Xing @ CMU, 2006-2011

35

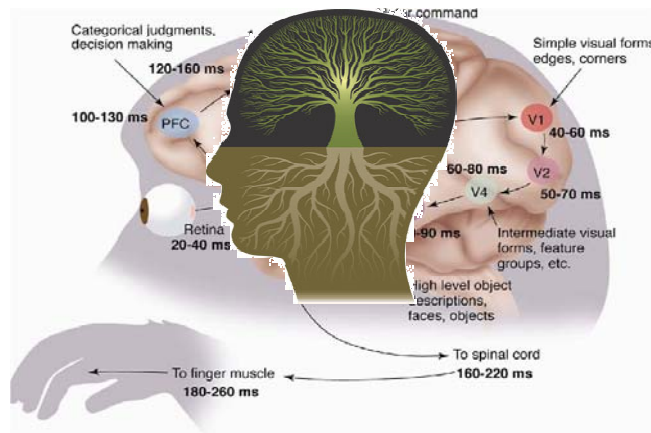
Artificial neural networks – what you should know

- Highly expressive non-linear functions
- Highly parallel network of logistic function units
- Minimizing sum of squared training errors
 - Gives MLE estimates of network weights if we assume zero mean Gaussian noise on output values
- Minimizing sum of sq errors plus weight squared (regularization)
 - MAP estimates assuming weight priors are zero mean Gaussian
- Gradient descent as training procedure
 - How to derive your own gradient descent procedure
- Discover useful representations at hidden units
- Local minima is greatest problem
- Overfitting, regularization, early stopping

© Eric Xing @ CMU, 2006-2011

36

Modern ANN topics: “Deep” Learning



© Eric Xing @ CMU, 2006-2011

37

Expressive Capabilities of ANNs

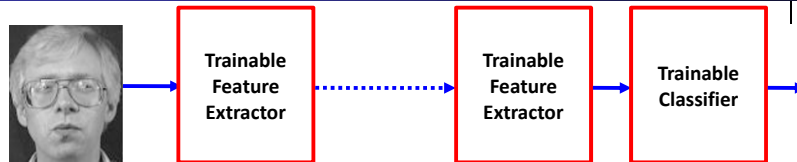


- **Boolean functions:**
 - Every Boolean function can be represented by network with single hidden layer
 - But might require exponential (in number of inputs) hidden units
- **Continuous functions:**
 - Every bounded continuous function can be approximated with arbitrary small error, by network with 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].

© Eric Xing @ CMU, 2006-2011

38

Using ANN to hierarchical representation



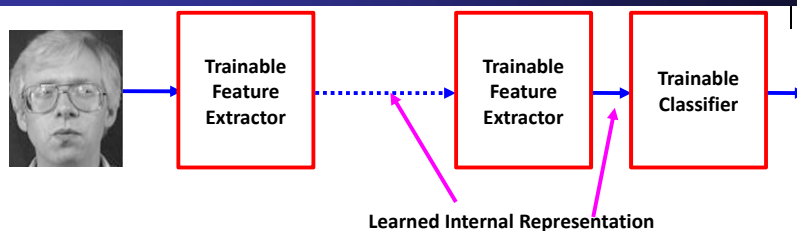
Good Representations are hierarchical

- In Language: hierarchy in syntax and semantics
 - Words->Parts of Speech->Sentences->Text
 - Objects,Actions,Attributes...-> Phrases -> Statements -> Stories
- In Vision: part-whole hierarchy
 - Pixels->Edges->Textons->Parts->Objects->Scenes

© Eric Xing @ CMU, 2006-2011

39

“Deep” learning: learning hierarchical representations



- Deep Learning: learning a hierarchy of internal representations
- From low-level features to mid-level invariant representations, to object identities
- Representations are increasingly invariant as we go up the layers
- using multiple stages gets around the specificity/invariance dilemma

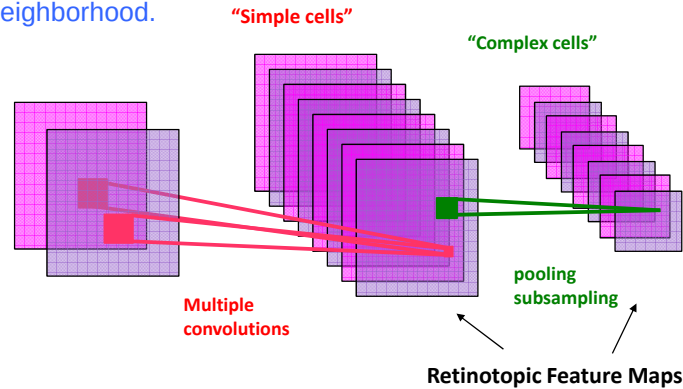
© Eric Xing @ CMU, 2006-2011

40

Filtering+NonLinearity+Pooling = 1 stage of a Convolutional Net



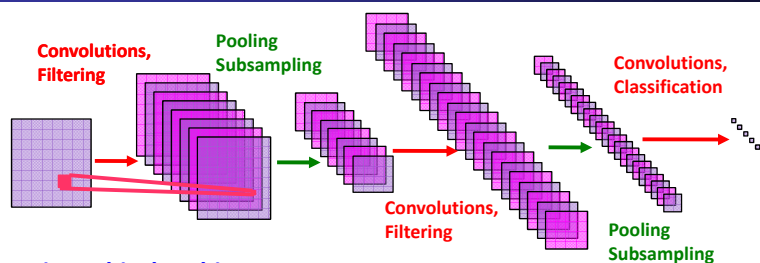
- [Hubel & Wiesel 1962]:
 - simple cells detect local features
 - complex cells “pool” the outputs of simple cells within a retinotopic neighborhood.



© Eric Xing @ CMU, 2006-2011

41

Convolutional Network: Multi-Stage Trainable Architecture



■ Hierarchical Architecture

- ▶ Representations are more global, more invariant, and more abstract as we go up the layers

■ Alternated Layers of Filtering and Spatial Pooling

- ▶ Filtering detects conjunctions of features
- ▶ Pooling computes local disjunctions of features

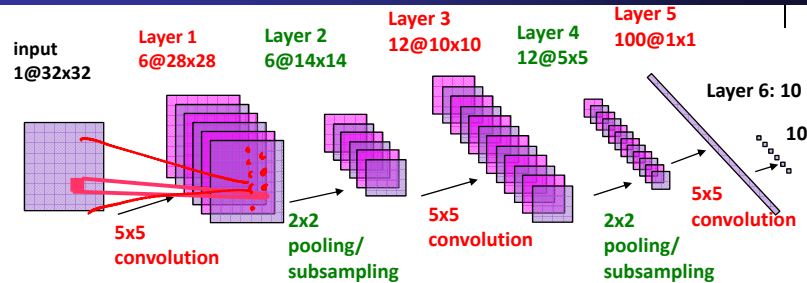
■ Fully Trainable

- ▶ All the layers are trainable

© Eric Xing @ CMU, 2006-2011

42

Convolutional Net Architecture for Hand-writing recognition



- Convolutional net for handwriting recognition (400,000 synapses)
 - Convolutional layers (simple cells): all units in a feature plane share the same weights
 - Pooling/subsampling layers (complex cells): for invariance to small distortions.
 - Supervised gradient-descent learning using back-propagation
 - The entire network is trained end-to-end. All the layers are trained simultaneously.
 - [LeCun et al. Proc IEEE, 1998]

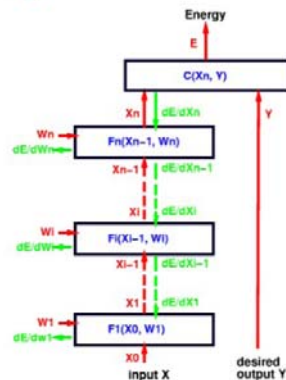
© Eric Xing @ CMU, 2006-2011

43

How to train?



To compute all the derivatives, we use a backward sweep called the **back-propagation algorithm** that uses the recurrence equation for $\frac{\partial E}{\partial X_i}$



$$\begin{aligned} \frac{\partial E}{\partial X_n} &= \frac{\partial C(X_n, Y)}{\partial X_n} \\ \frac{\partial E}{\partial X_{n-1}} &= \frac{\partial E}{\partial X_n} \frac{\partial F_n(X_{n-1}, W_n)}{\partial X_{n-1}} \\ \frac{\partial E}{\partial W_n} &= \frac{\partial E}{\partial X_n} \frac{\partial F_n(X_{n-1}, W_n)}{\partial W_n} \\ \frac{\partial E}{\partial X_{n-2}} &= \frac{\partial E}{\partial X_{n-1}} \frac{\partial F_{n-1}(X_{n-2}, W_{n-1})}{\partial X_{n-2}} \\ \frac{\partial E}{\partial W_{n-1}} &= \frac{\partial E}{\partial X_{n-1}} \frac{\partial F_{n-1}(X_{n-2}, W_{n-1})}{\partial W_{n-1}} \\ &\dots \text{etc, until we reach the first module.} \\ &\text{we now have all the } \frac{\partial E}{\partial W_i} \text{ for } i \in [1, n]. \end{aligned}$$

© Eric Xing @ CMU, 2006-2011

44

Application: MNIST Handwritten Digit Dataset



3 6 8 1 7 9 6 6 4 1
6 7 5 7 8 6 3 4 8 5
2 1 7 9 7 1 2 8 4 5
4 8 1 9 0 1 8 8 9 4
7 6 1 8 6 4 1 5 6 0
7 5 9 2 6 5 8 1 9 7
2 2 2 2 2 3 4 4 8 0
0 2 3 8 0 7 3 8 5 7
0 1 4 6 4 6 0 2 4 3
7 1 2 8 7 6 9 8 6 1

0 0 0 0 0 0 0 0 0 0
1 1 1 1 1 1 1 1 1 1
2 2 2 2 2 2 2 2 2 2
3 3 3 3 3 3 3 3 3 3
4 4 4 4 4 4 4 4 4 4
5 5 5 5 5 5 5 5 5 5
6 6 6 6 6 6 6 6 6 6
7 7 7 7 7 7 7 7 7 7
8 8 8 8 8 8 8 8 8 8
9 9 9 9 9 9 9 9 9 9

Handwritten Digit Dataset MNIST: 60,000 training samples, 10,000 test samples

© Eric Xing @ CMU, 2006-2011

45

Results on MNIST Handwritten Digits



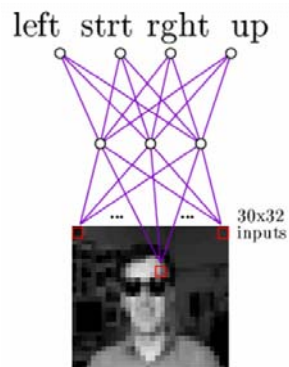
CLASSIFIER	DEFORMATION	PREPROCESSING	ERROR (%)	Reference
linear classifier (1-layer NN)		none	12.00	LeCun et al. 1998
linear classifier (1-layer NN)		deskewing	8.40	LeCun et al. 1998
pairwise linear classifier		deskewing	7.60	LeCun et al. 1998
K-nearest-neighbors, (L2)		none	3.09	Kenneth Wilder, U. Chicago
K-nearest-neighbors, (L2)		deskewing	2.40	LeCun et al. 1998
K-nearest-neighbors, (L2)		deskew, clean, blur	1.80	Kenneth Wilder, U. Chicago
K-NN L3, 2 pixel jitter		deskew, clean, blur	1.22	Kenneth Wilder, U. Chicago
K-NN, shape context matching		shape context feature	0.63	Belongie et al. IEEE PAMI 2002
40 PCA + quadratic classifier		none	3.30	LeCun et al. 1998
1000 RBF + linear classifier		none	3.60	LeCun et al. 1998
K-NN, Tangent Distance		subsamp 16x16 pixels	1.10	LeCun et al. 1998
SVM, Gaussian Kernel		none	1.40	
SVM deg 4 polynomial		deskewing	1.10	LeCun et al. 1998
Reduced Set SVM deg 5 poly		deskewing	1.00	LeCun et al. 1998
Virtual SVM deg-9 poly		none	0.80	LeCun et al. 1998
V-SVM, 2-pixel jittered	Affine	none	0.68	DeCoste and Scholkopf, MLJ2002
V-SVM, 2-pixel jittered		deskewing	0.56	DeCoste and Scholkopf, MLJ2002
2-layer NN, 300 HU, MSE		none	4.70	LeCun et al. 1998
2-layer NN, 300 HU, MSE	Affine	none	3.60	LeCun et al. 1998
2-layer NN, 300 HU		deskewing	1.60	LeCun et al. 1998
3-layer NN, 500+ 150 HU		none	2.95	LeCun et al. 1998
3-layer NN, 500+ 150 HU	Affine	none	2.45	LeCun et al. 1998
3-layer NN, 500+ 300 HU, CE, reg		none	1.53	Hinton, unpublished, 2005
2-layer NN, 800 HU, CE		none	1.60	Simard et al., ICDAR 2003
2-layer NN, 800 HU, CE	Affine	none	1.10	Simard et al., ICDAR 2003
2-layer NN, 800 HU, MSE	Elastic	none	0.90	Simard et al., ICDAR 2003
2-layer NN, 800 HU, CE	Elastic	none	0.70	Simard et al., ICDAR 2003
Convolutional net LeNet-1		subsamp 16x16 pixels	1.70	LeCun et al. 1998
Convolutional net LeNet-4		none	1.10	LeCun et al. 1998
Convolutional net LeNet-5,		none	0.95	LeCun et al. 1998
Conv. net LeNet-5,	Affine	none	0.80	LeCun et al. 1998
Boosted LeNet-4	Affine	none	0.70	LeCun et al. 1998
Conv. net, CE	Affine	none	0.60	Simard et al., ICDAR 2003
Conv net, CE	Elastic	none	0.40	Simard et al., ICDAR 2003

© Eric Xing @ CMU, 2006-2011

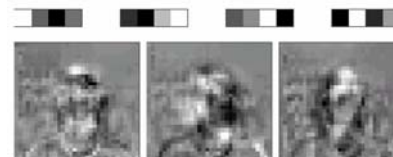
46

Application: ANN for Face Reco.

- The model



- The learned hidden unit weights



Typical input images

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

© Eric Xing @ CMU, 2006-2011

47

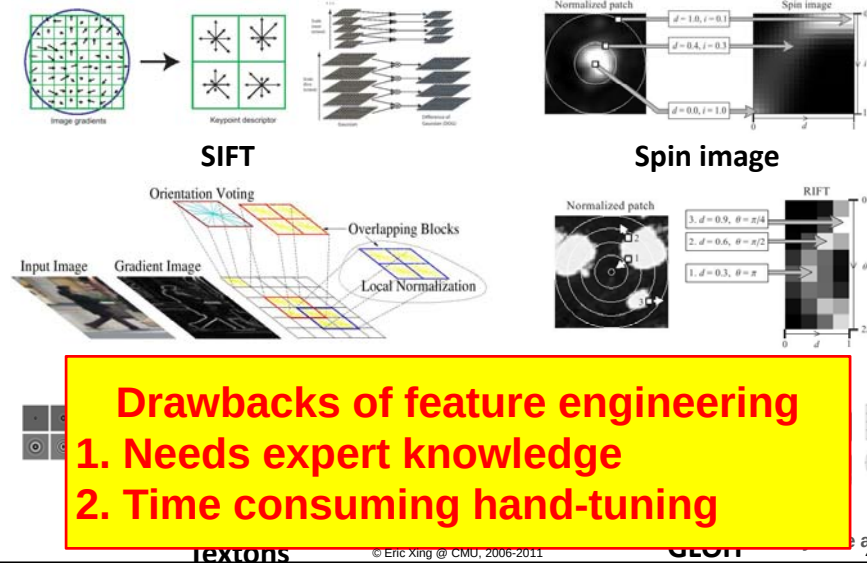
Face Detection with a Convolutional Net



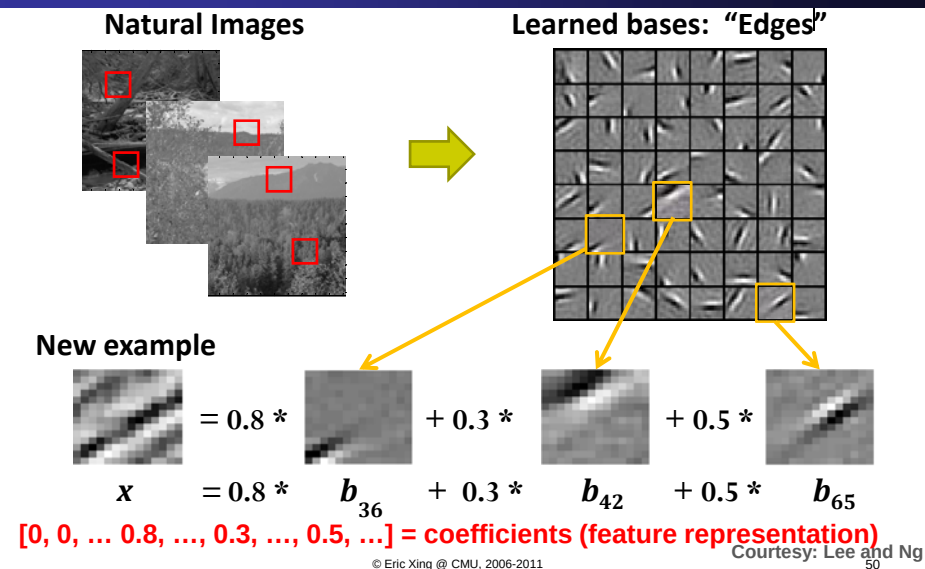
© Eric Xing @ CMU, 2006-2011

48

Computer vision features



Sparse coding on images



Basis (or features) can be learned by Optimization



Given input data $\{x^{(1)}, \dots, x^{(m)}\}$, we want to find good bases $\{b_1, \dots, b_n\}$:

$$\min_{b,a} \sum_i \underbrace{\|x^{(i)} - \sum_j a_j^{(i)} b_j\|_2^2}_{\text{Reconstruction error}} + \underbrace{\beta \sum_i \|a^{(i)}\|_1}_{\text{Sparsity penalty}}$$

$\forall j: \|b_j\| \leq 1$ **Normalization constraint**

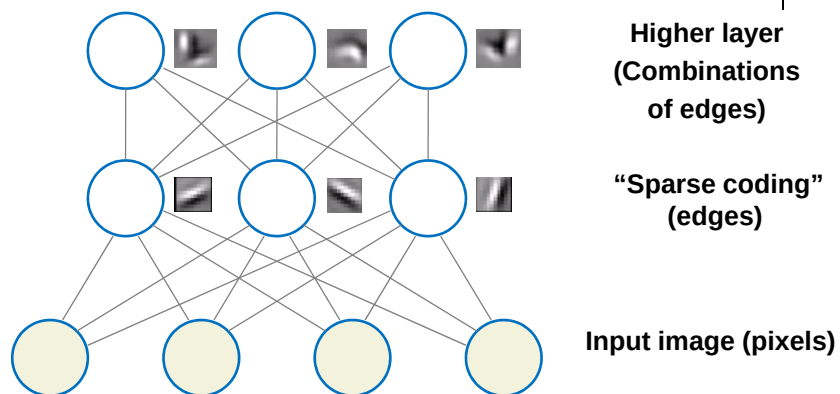
Solve by alternating minimization:

- Keep b fixed, find optimal a .
- Keep a fixed, find optimal b .

© Eric Xing @ CMU, 2006-2011

Courtesy: Lee and Ng
51

Learning Feature Hierarchy



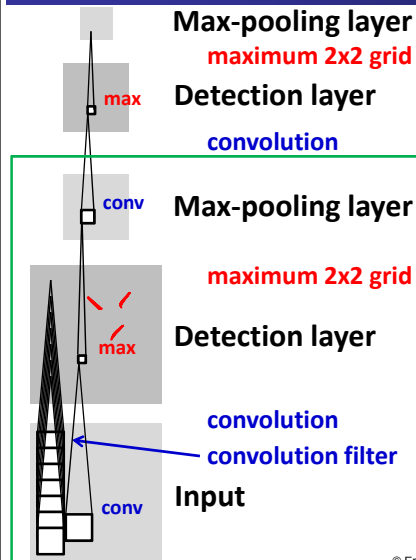
DBN (Hinton et al., 2006) with additional sparseness constraint.

[Related work: Hinton, Bengio, LeCun, and others.]

© Eric Xing @ CMU, 2006-2011

Courtesy: Lee and Ng
52

Convolutional architectures



- Weight sharing by convolution (e.g., [Lecun et al., 1989])
- “Max-pooling”
 - Invariance
 - Computational efficiency
 - Deterministic and feed-forward
- One can develop convolutional Restricted Boltzmann machine (CRBM).
- One can define *probabilistic max-pooling* that combine bottom-up and top-down information.

© Eric Xing @ CMU, 2006-2011

Courtesy: Lee and Ng
53

Convolutional Deep Belief Networks

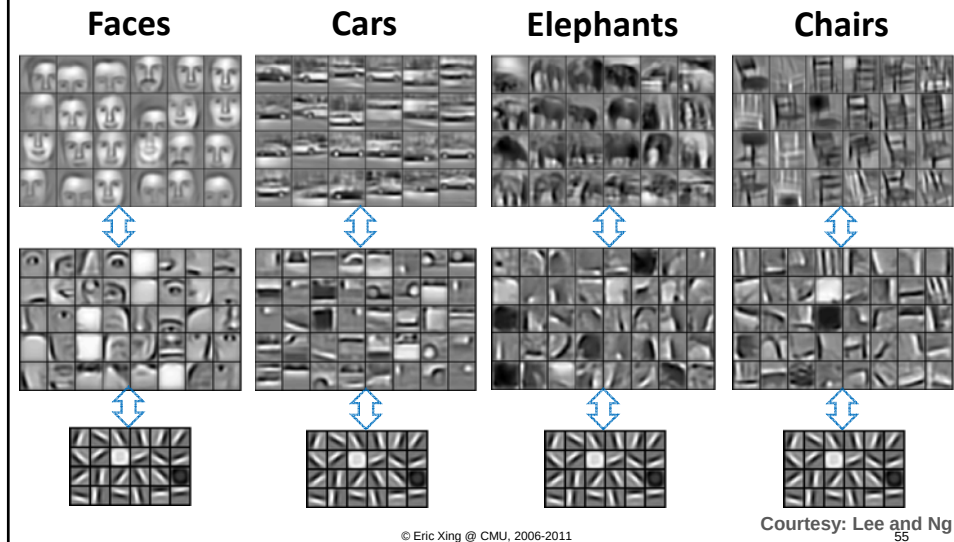


- Bottom-up (greedy), layer-wise training
 - Train one layer (convolutional RBM) at a time.
- Inference (approximate)
 - Undirected connections for all layers (Markov net) [Related work: Salakhutdinov and Hinton, 2009]
 - Block Gibbs sampling or mean-field
 - Hierarchical probabilistic inference

© Eric Xing @ CMU, 2006-2011

Courtesy: Lee and Ng
54

Unsupervised learning of object-parts



Weaknesses & Criticisms



- Learning everything. Better to encode prior knowledge about structure of images.

A: Compare with machine learning vs. linguists debate in NLP.

- Results not yet competitive with best engineered systems.

A: Agreed. True for some domains.