Harvard-MIT Division of Health Sciences and Technology
HST.951J: Medical Decision Support, Fall 2005
Instructors: Professor Lucila Ohno-Machado and Professor Staal Vinterbo

6.873/HST.951 Medical Decision Support
Spring 2005

Artificial Neural Networks

Lucila Ohno-Machado
(with many slides borrowed from Stephan Dreiseitl. Courtesy of Stephan Dreiseitl. Used with permission.)

Overview

- Motivation
- Perceptrons
- Multilayer perceptrons
- Improving generalization
- Bayesian perspective

Motivation

Images removed due to copyright reasons.

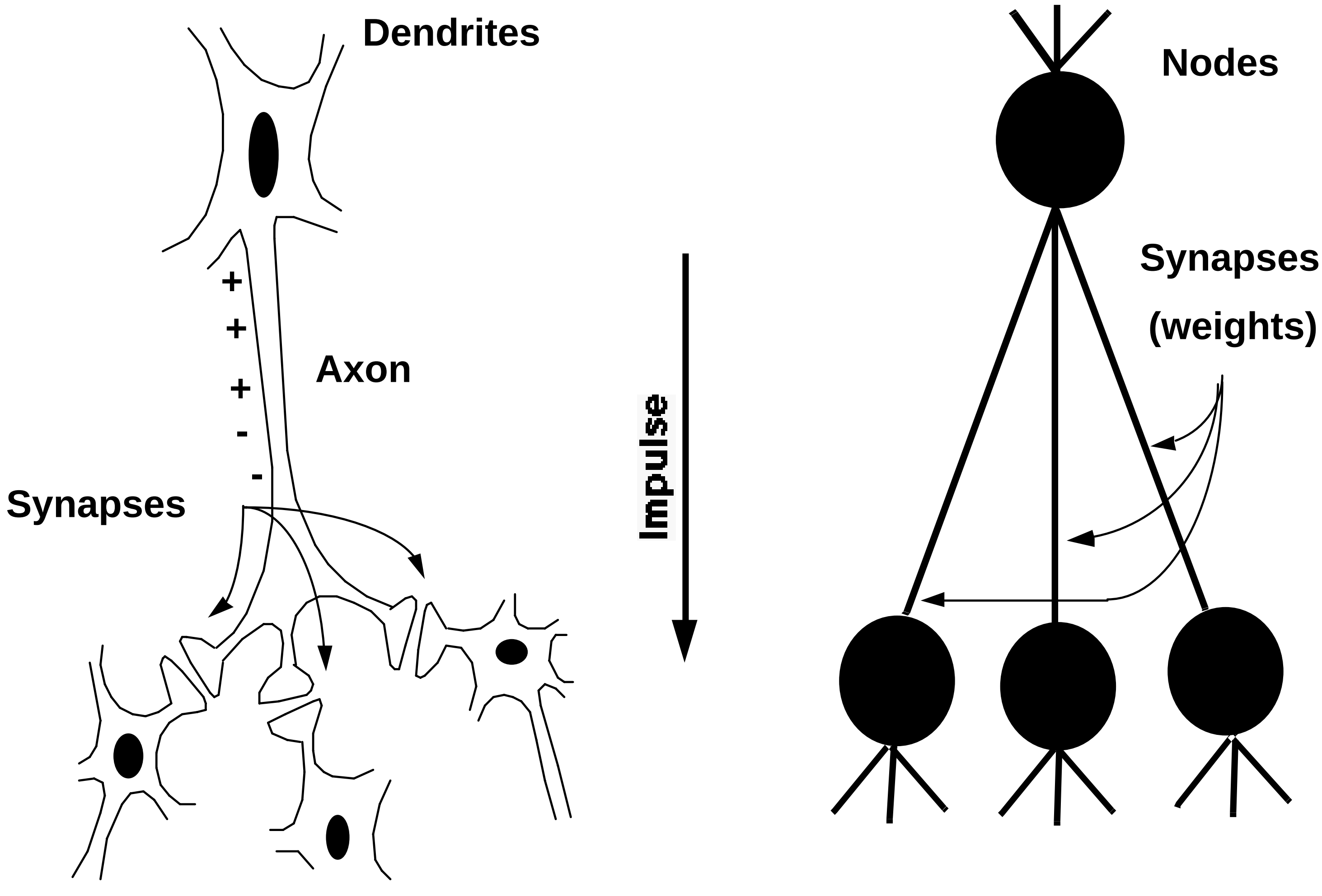
benign lesion

malignant lesion

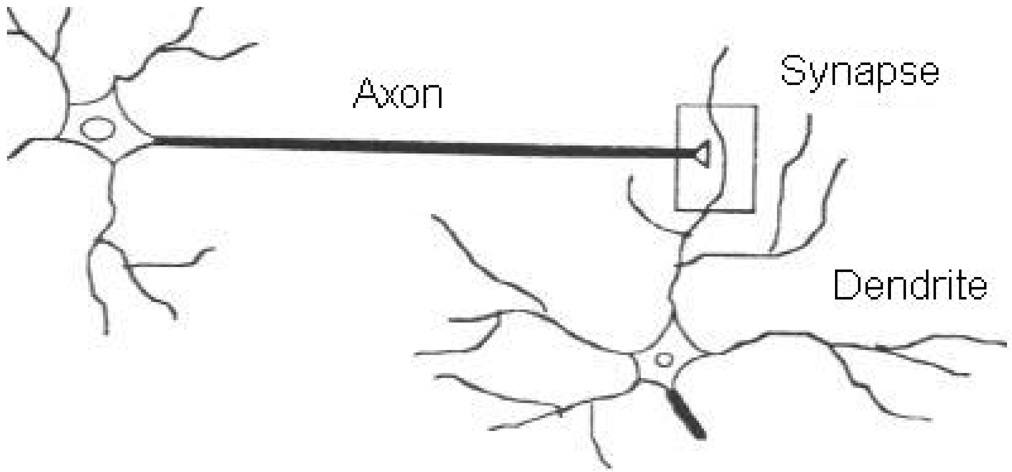
Motivation

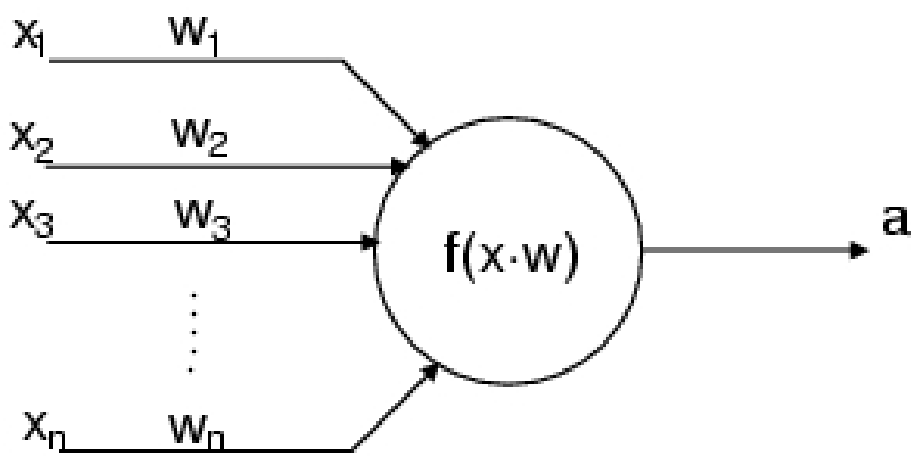
- Human brain
  - Parallel processing
  - Distributed representation
  - Fault tolerant
  - Good generalization capability
- Mimic structure and processing in computational model

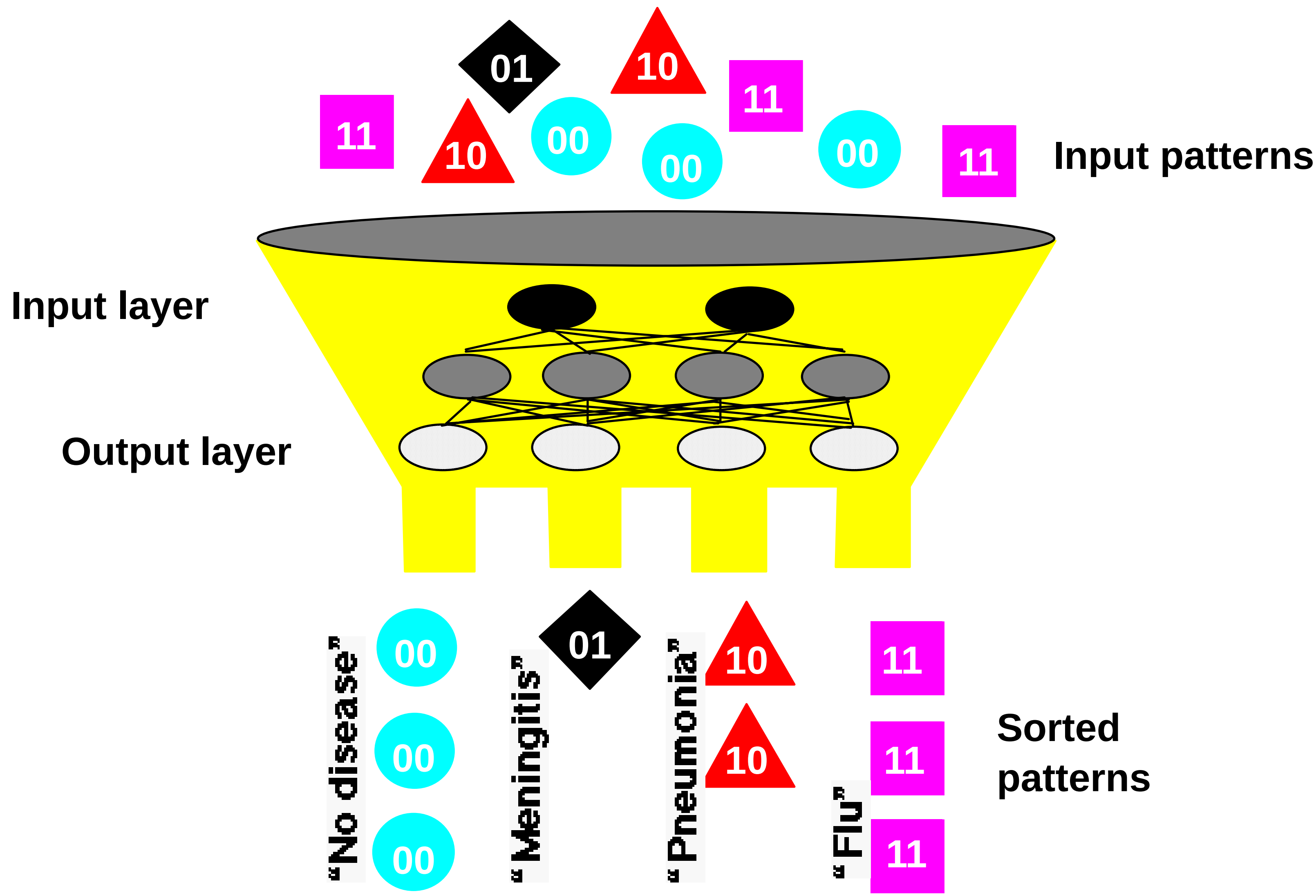
Biological Analogy



Perceptrons

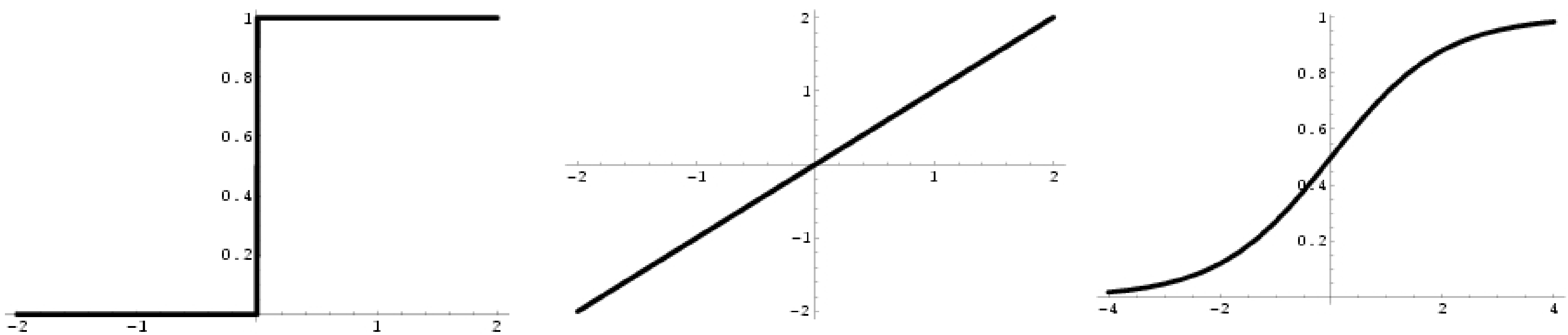


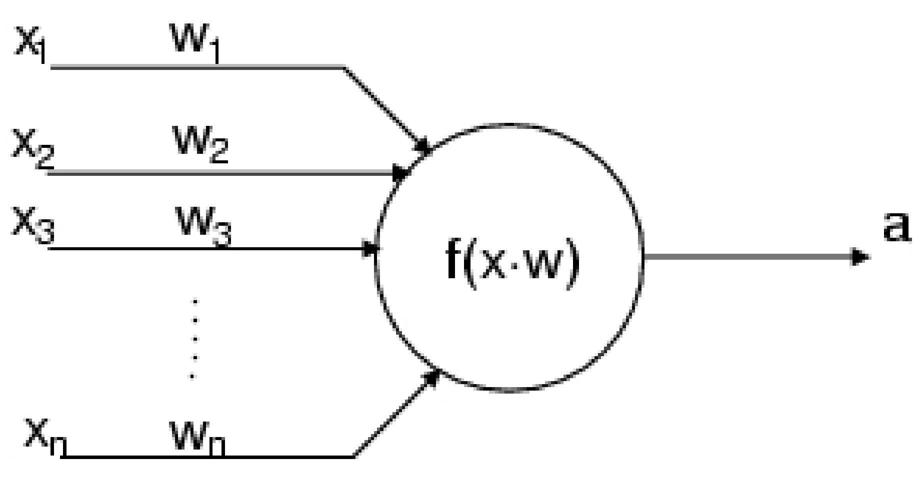




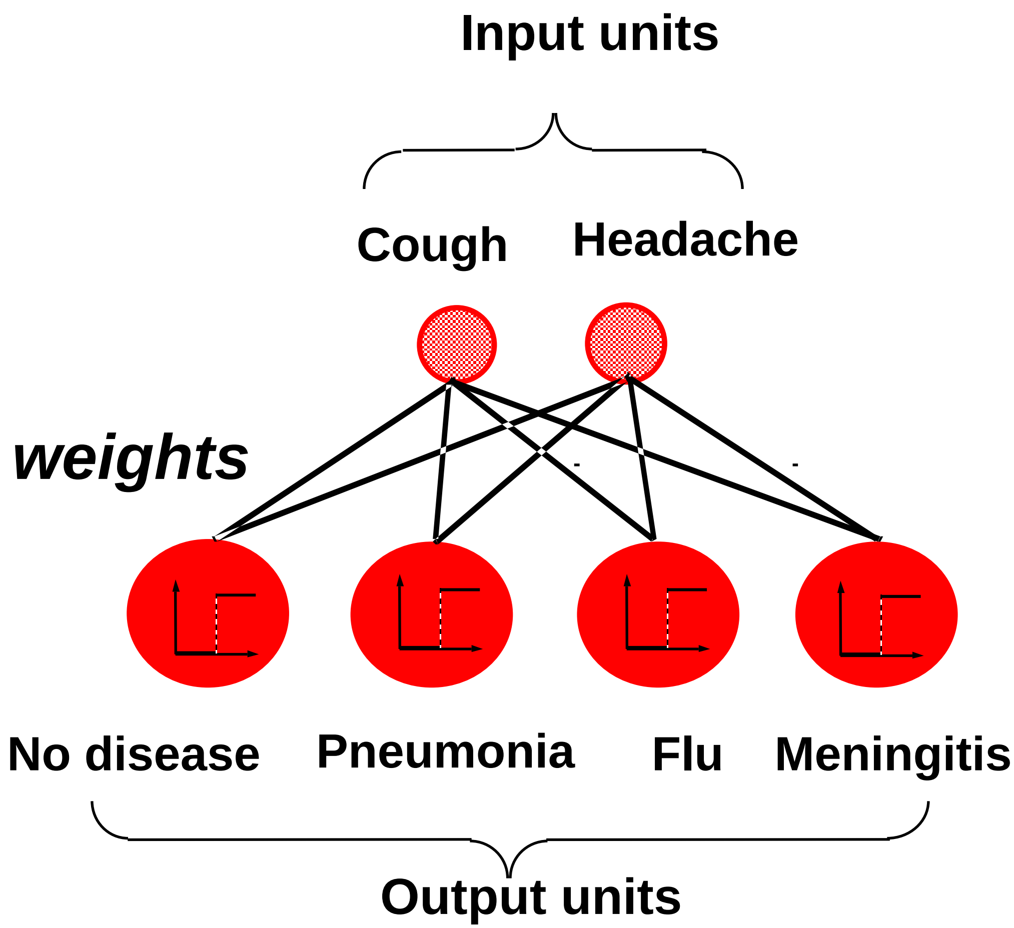
01
10
11
11
10
00
00
00
11
Input patterns
Input layer
Output layer
"No disease"
00
00
00
"Meningitis"
01
"Pneumonia"
10
10
"Flu"
11
11
11
Sorted patterns

Activation functions





Perceptrons (linear machines)

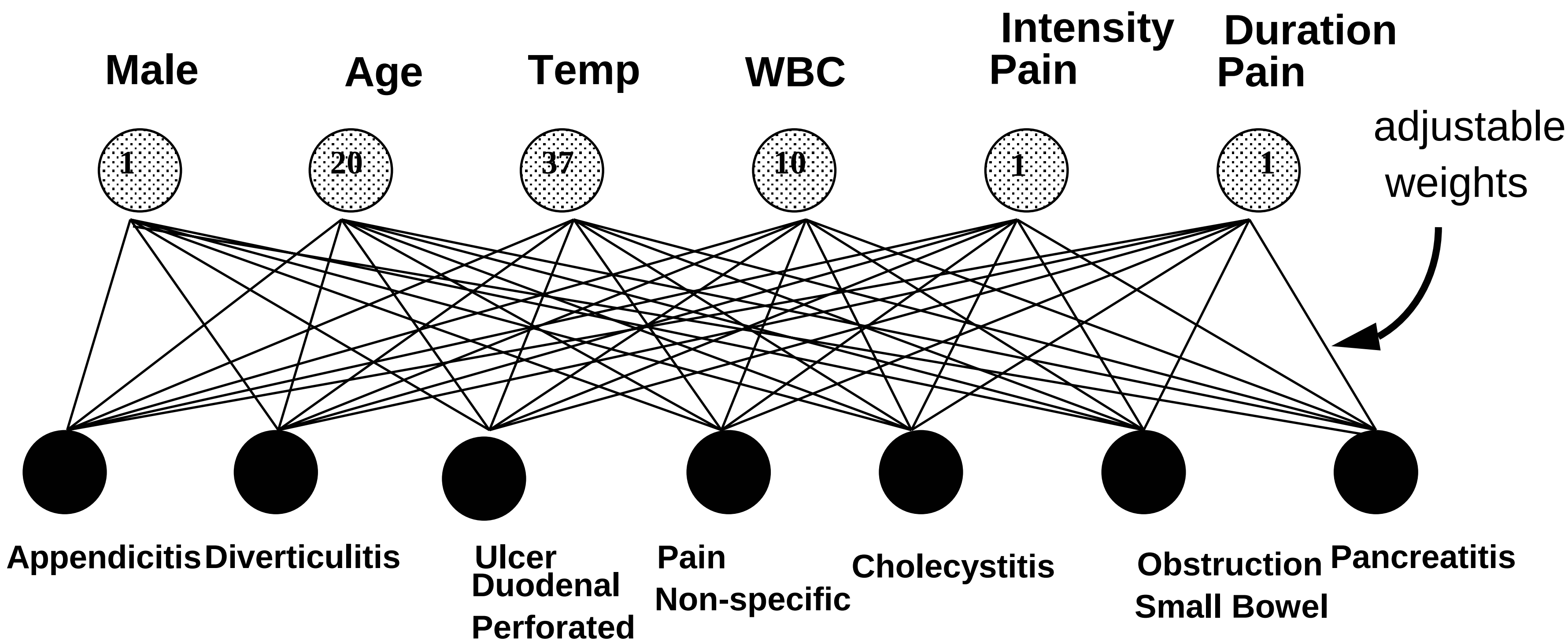


Δ rule
change weights to decrease the error

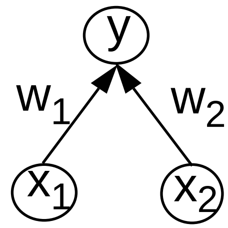
- what we got
what we wanted

error

Abdominal Pain Perceptron



AND



$\theta = 0.5$

input	output
00	0
01	0
10	0
11	1

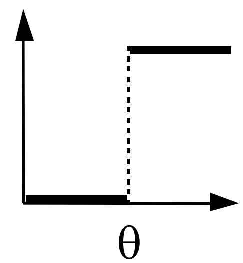
$$f(x_1 w_1 + x_2 w_2) = y$$

$$f(0w_1 + 0w_2) = 0$$

$$f(0w_1 + 1w_2) = 0$$

$$f(1w_1 + 0w_2) = 0$$

$$f(1w_1 + 1w_2) = 1$$

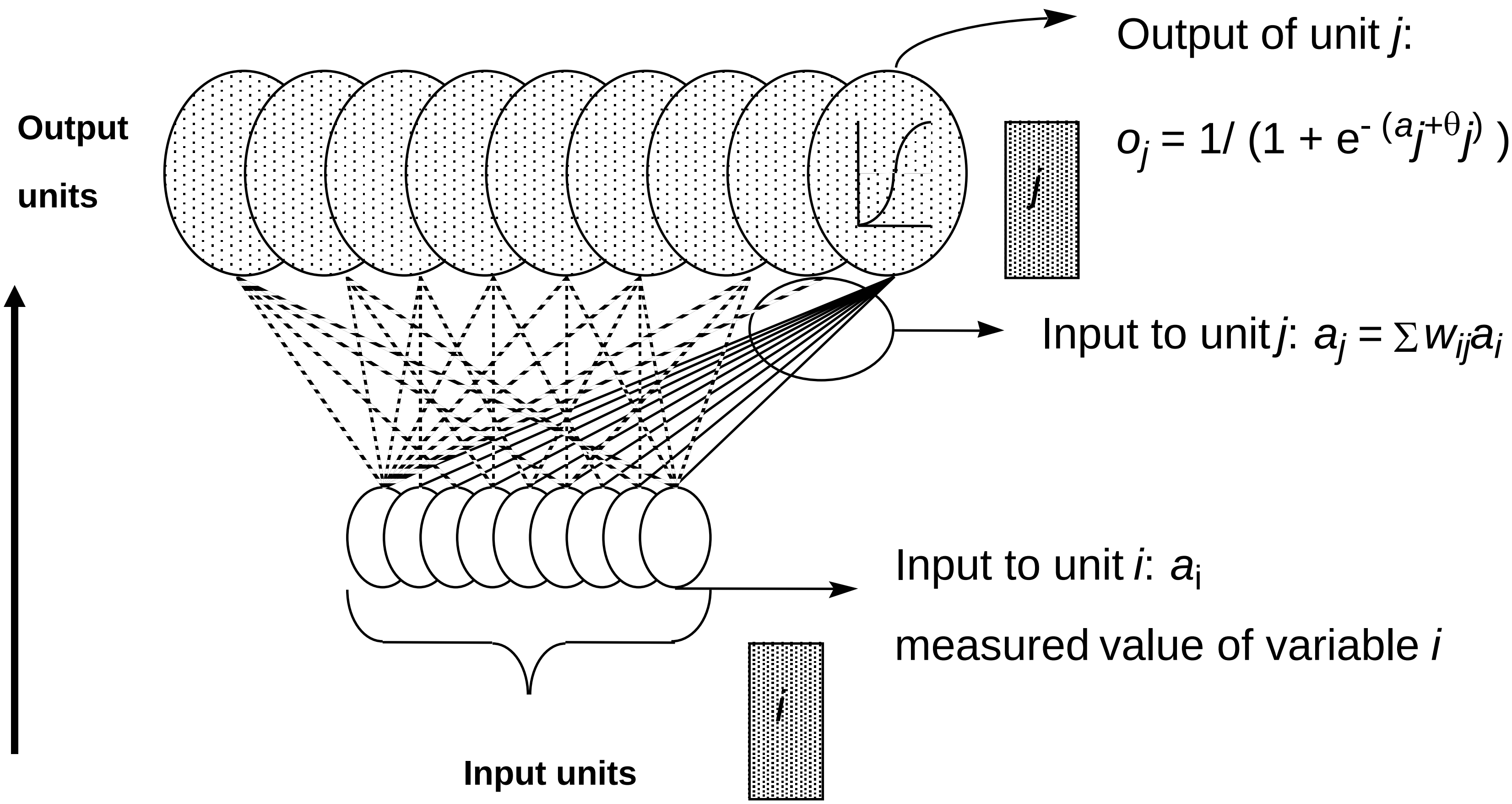


$$f(a) = \begin{cases} 1, \text{ for } a > \theta \\ 0, \text{ for } a \leq \theta \end{cases}$$

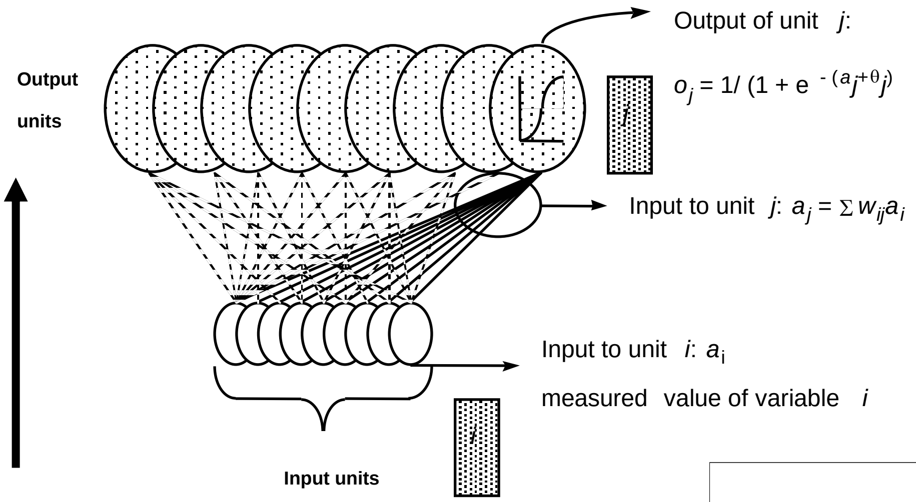
some possible values for w_1 and w_2

w_1	w_2
0.20	0.35
0.20	0.40
0.25	0.30
0.40	0.20

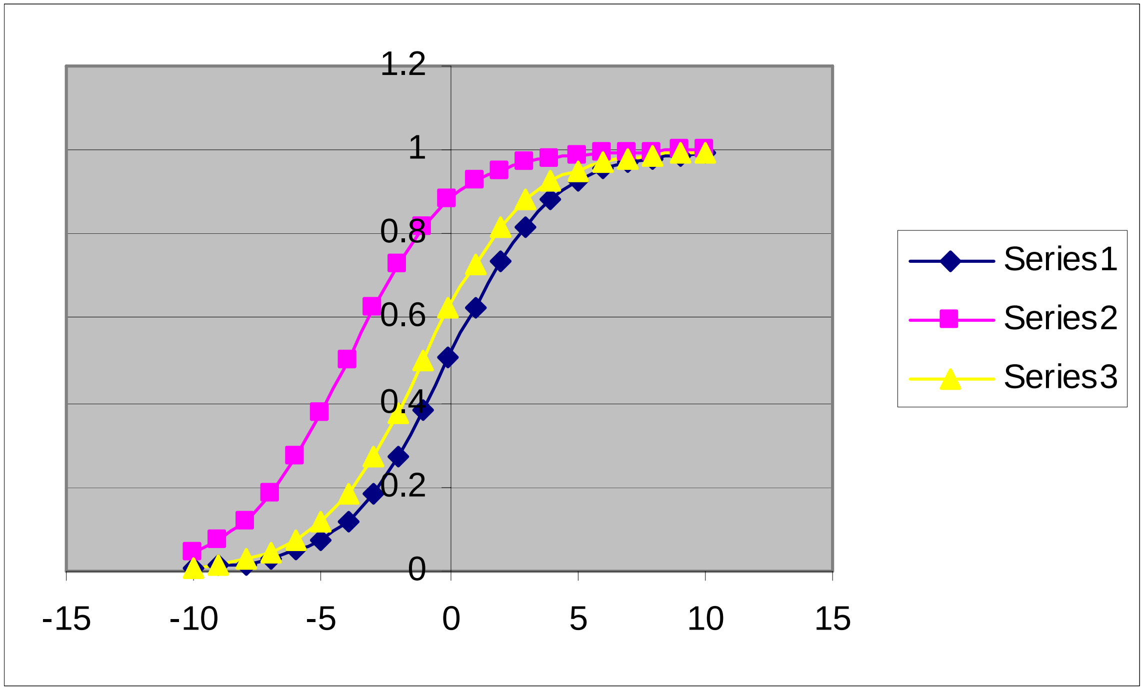
Single layer neural network



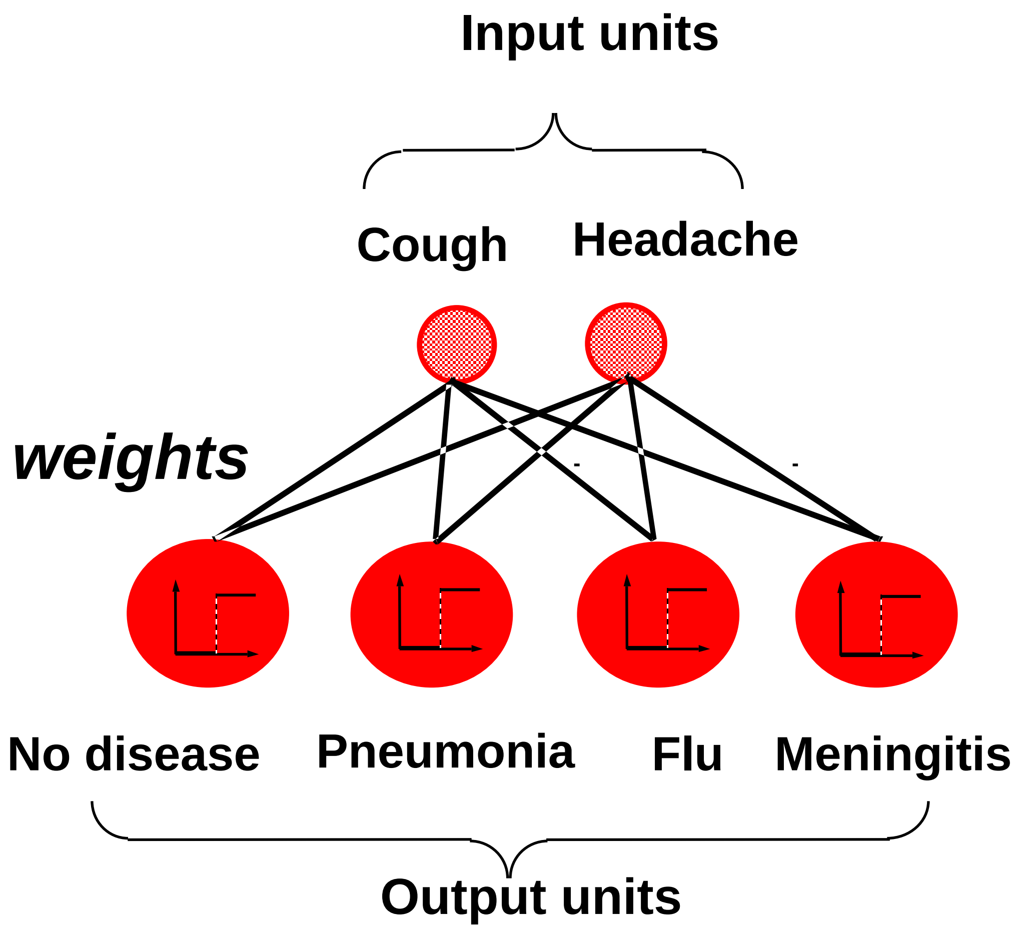
Single layer neural network



Increasing θ



Training: Minimize Error



Δ rule
change weights to decrease the error

  what we got
- what we wanted
error

Error Functions

- Mean Squared Error (for regression problems), where *t* is target, *o* is output

$$\Sigma(t - o)^2/n$$

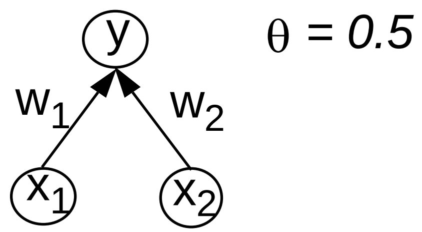
- Cross Entropy Error (for binary classification)

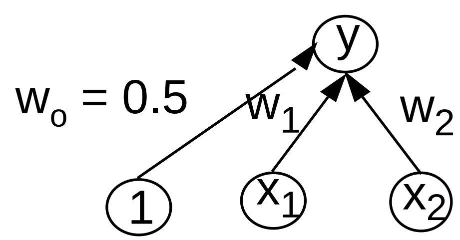
$$-\ \Sigma(t \log o) + (1-t) \log (1-o)$$

$o_j = 1/\ (1 + e^{-(a_j+\theta_j)}\ )$

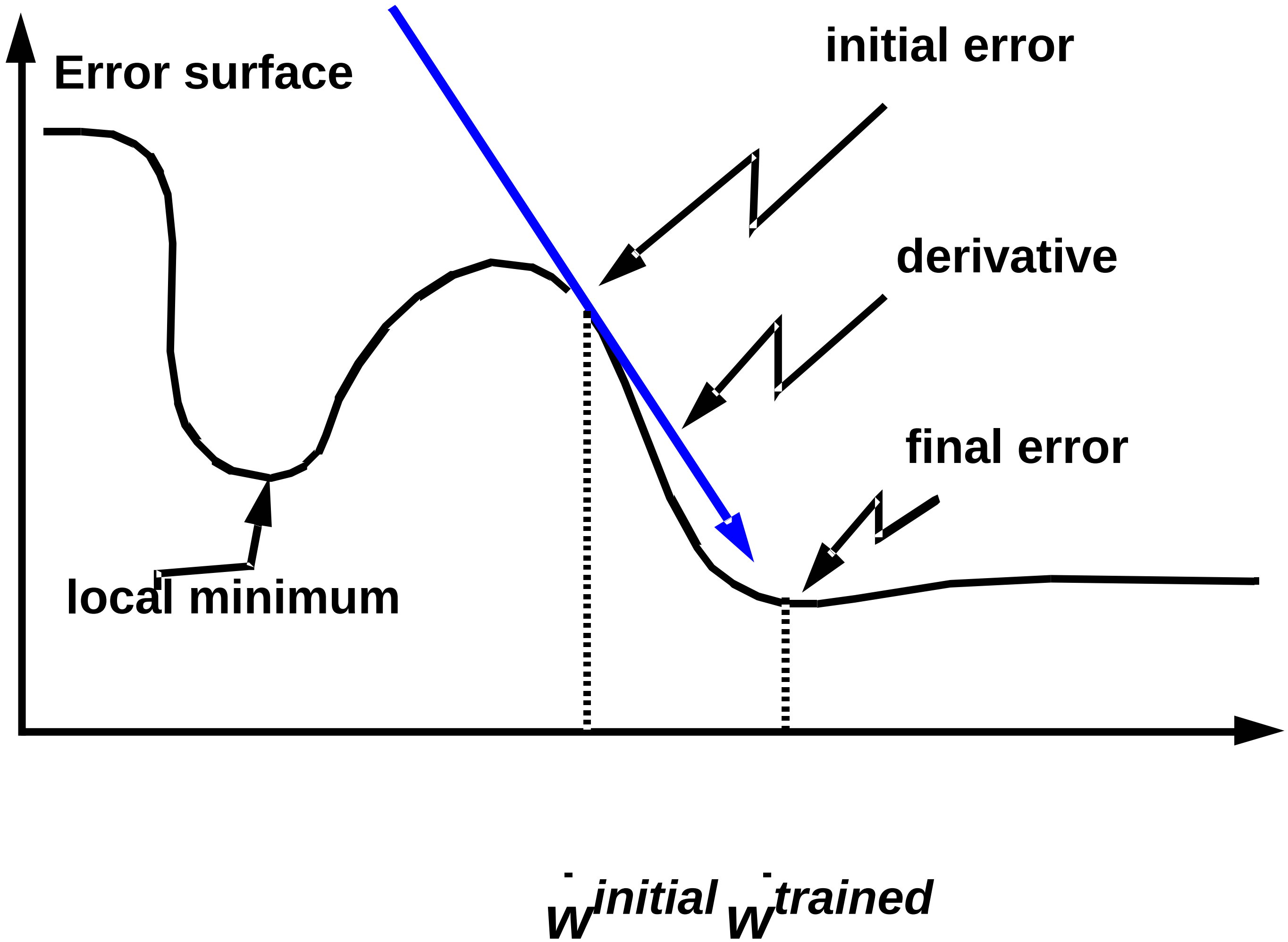
Error function

- Convention: $w := (w_0, w)$, $x := (1, x)$
- w_0 is "bias"
- $o = f(w \bullet x)$
- Class labels $t_i \in \{+1, -1\}$
- Error measure
  - $E = -\sum_{i \text{ miscl.}} t_i (w \bullet x_i)$
- How to minimize E?

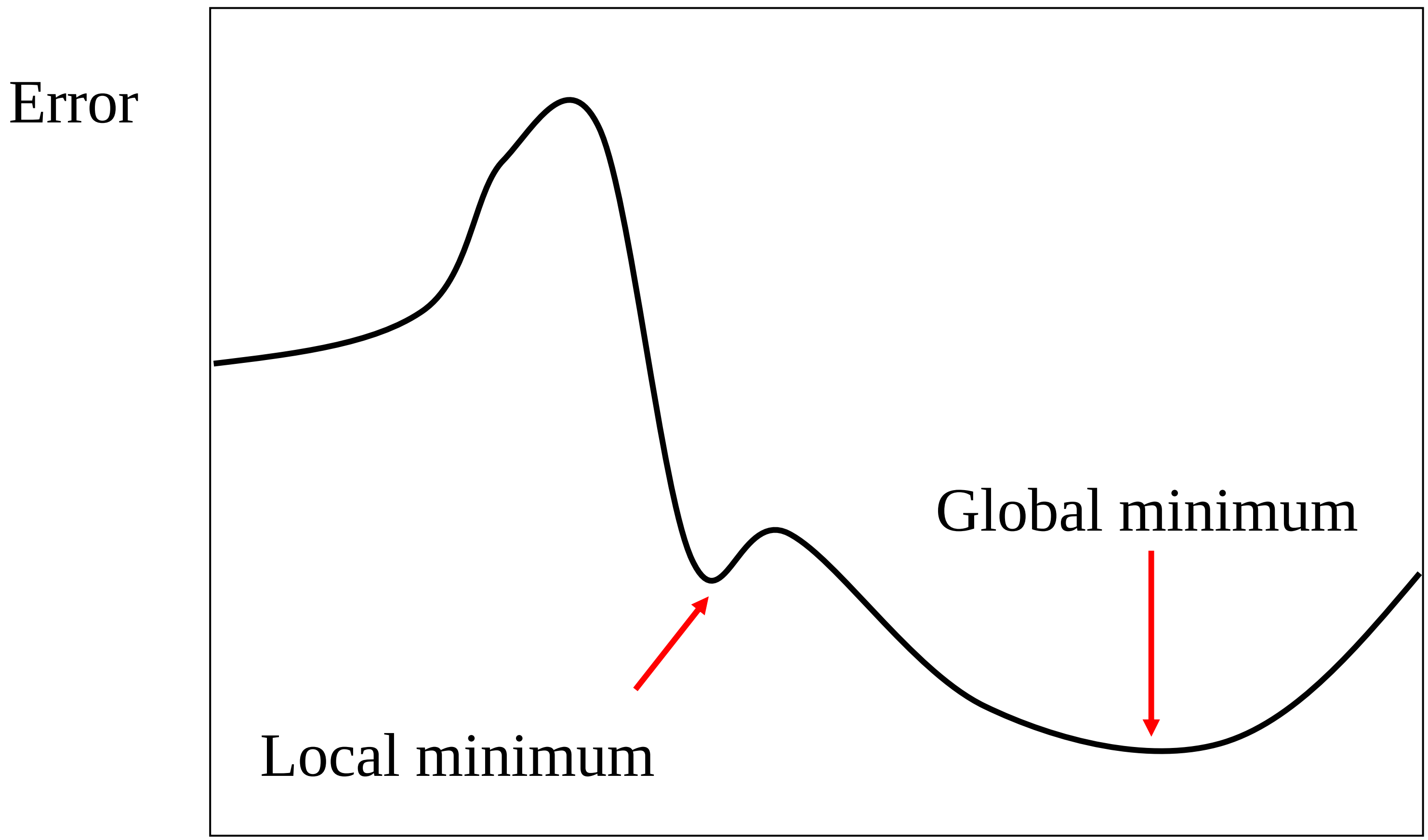




Minimizing the Error



Gradient descent



Perceptron learning

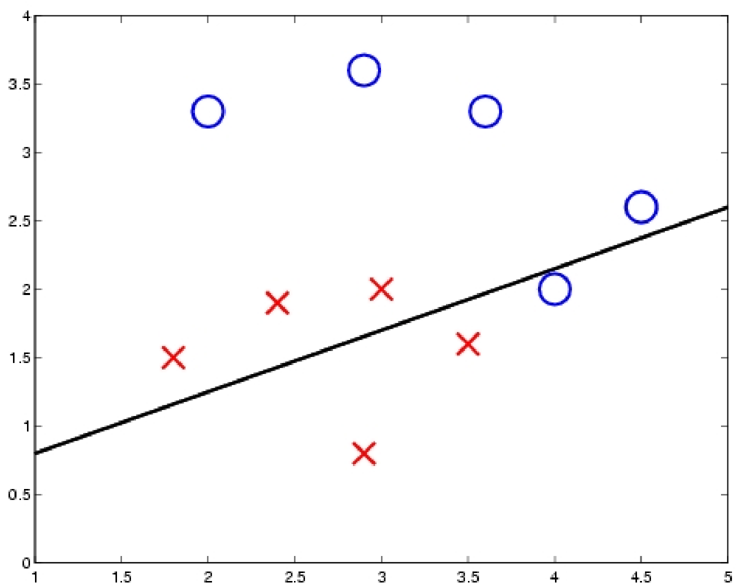
- Find minimum of E by iterating
  $w_{k+1} = w_k - \eta \, \mathrm{grad}_w E$
- $E = -\sum_{i \text{ miscl.}} t_i (w \bullet x_i) \Rightarrow$

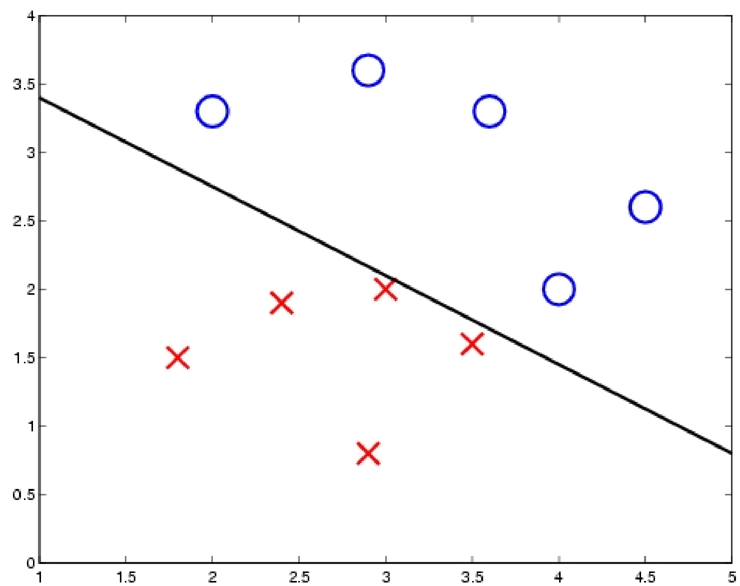
  $$\mathrm{grad}_w E = -\sum_{i \text{ miscl.}} t_i \; x_i$$

- "online" version: pick misclassified x_i
  $w_{k+1} = w_k + \eta \, t_i \, x_i$

Perceptron learning

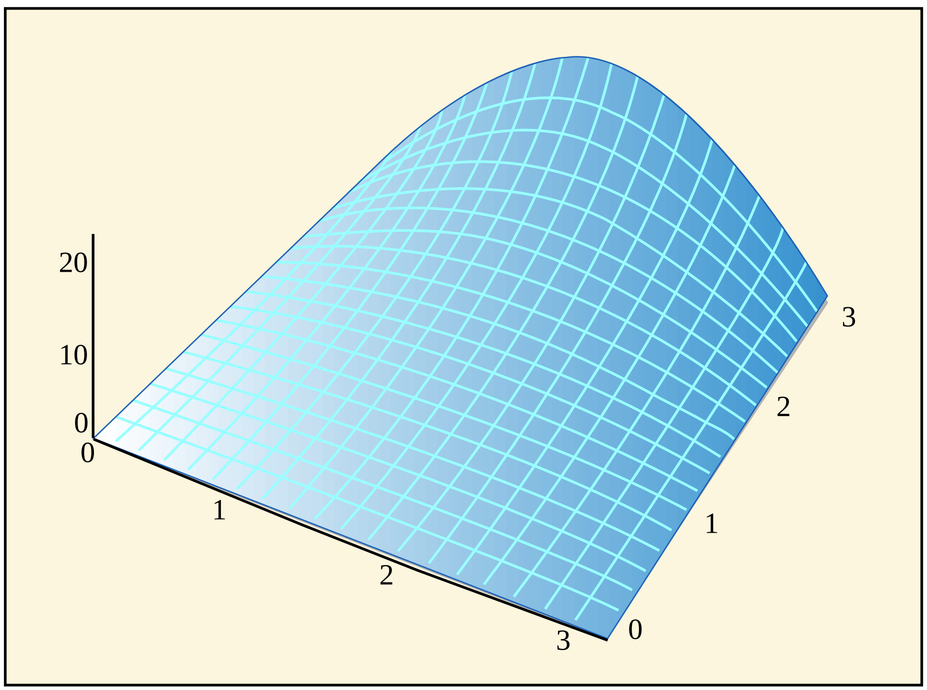
- Update rule $w_{k+1} = w_k + \eta\, t_i\, x_i$
- Theorem: perceptron learning converges for linearly separable sets

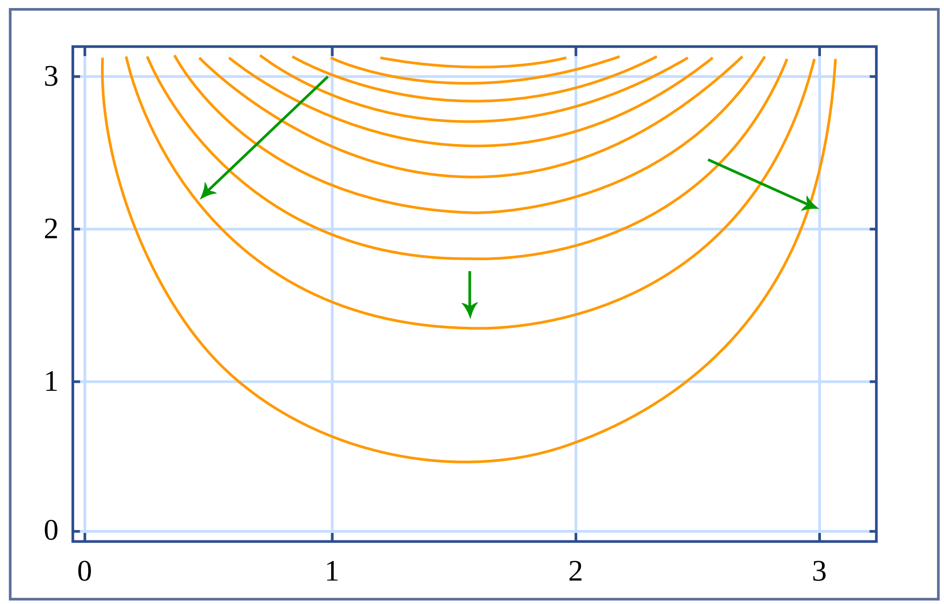




Gradient descent

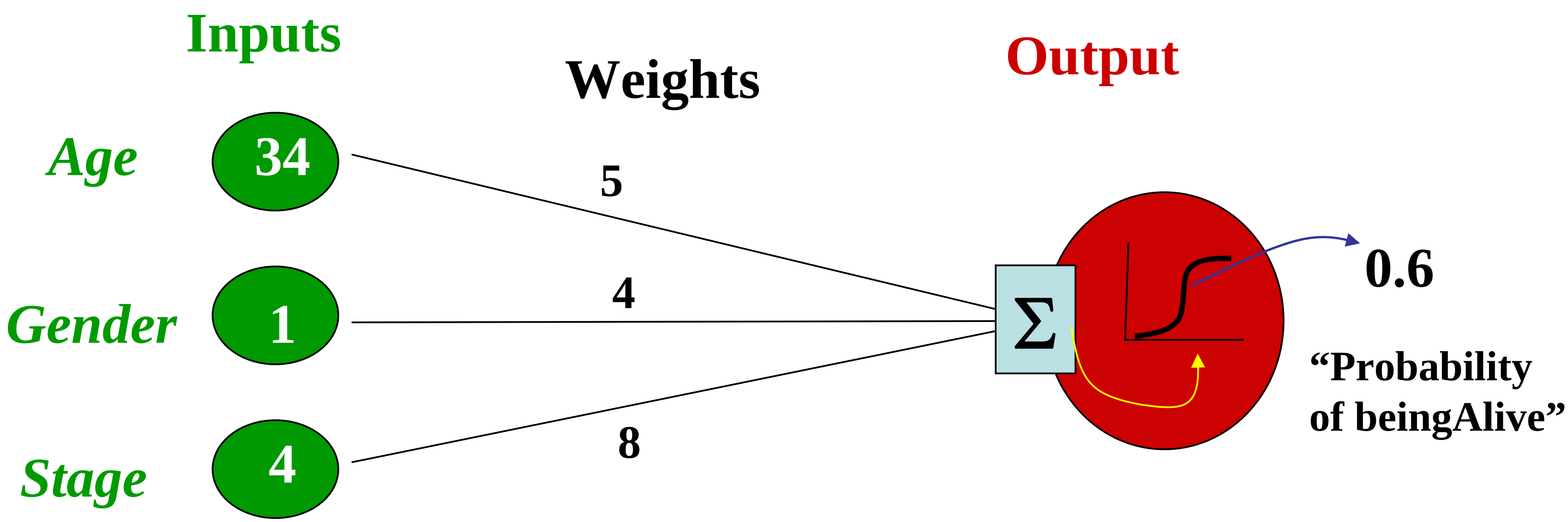
- Simple function minimization algorithm
- Gradient is vector of partial derivatives
- Negative gradient is direction of steepest descent





Figures by MIT OCW.

Classification Model



Independent variables

x1, x2, x3

Coefficients

a, b, c

Dependent variable

p

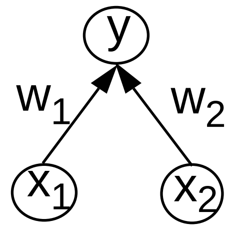
Prediction

Terminology

- Independent variable = input variable
- Dependent variable = output variable
- Coefficients = “weights”
- Estimates = “targets”

- Iterative step = cycle, epoch

XOR

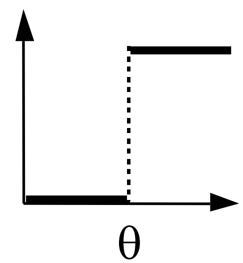


$\theta = 0.5$

input	output
00	0
01	1
10	1
11	0

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

$$f(0w_1 + 0w_2) = 0$$
$$f(0w_1 + 1w_2) = 1$$
$$f(1w_1 + 0w_2) = 1$$
$$f(1w_1 + 1w_2) = 0$$



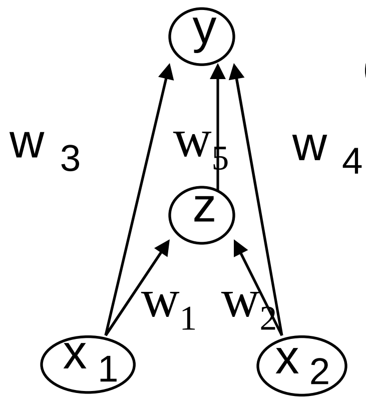
$$f(a) = \begin{cases} 1, \text{ for } a > \theta \\ 0, \text{ for } a \leq \theta \end{cases}$$

some possible values for w_1 and w_2

w_1	w_2

XOR

input	output
00	0
01	1
10	1
11	0



θ *= 0.5*

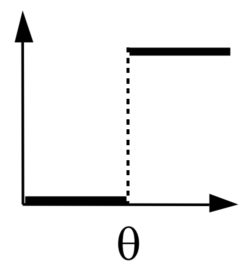
θ *= 0.5*

$$f(w_1, w_2, w_3, w_4, w_5)$$

a possible set of values for w_s

$(w_1, w_2, w_3, w_4, w_5)$

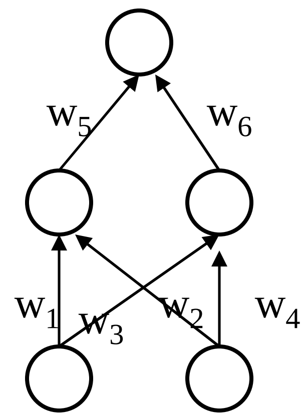
(0.3,0.3,1,1,-2)



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

XOR

input	output
00	0
01	1
10	1
11	0



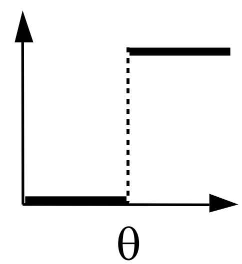
θ = 0.5 for all units

$$f(w_1, w_2, w_3, w_4, w_5, w_6)$$

a possible set of values for w_s

$$(w_1, w_2, w_3, w_4, w_5, w_6)$$

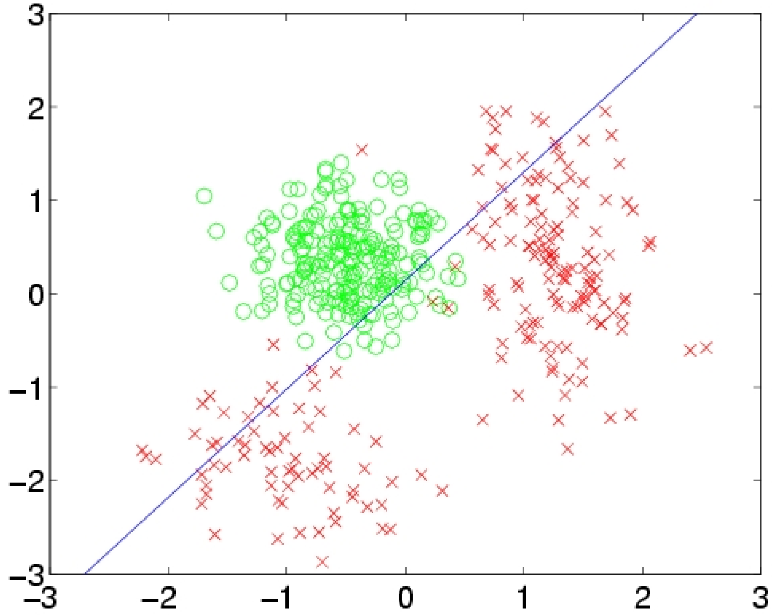
$$(0.6,-0.6,-0.7,0.8,1,1)$$

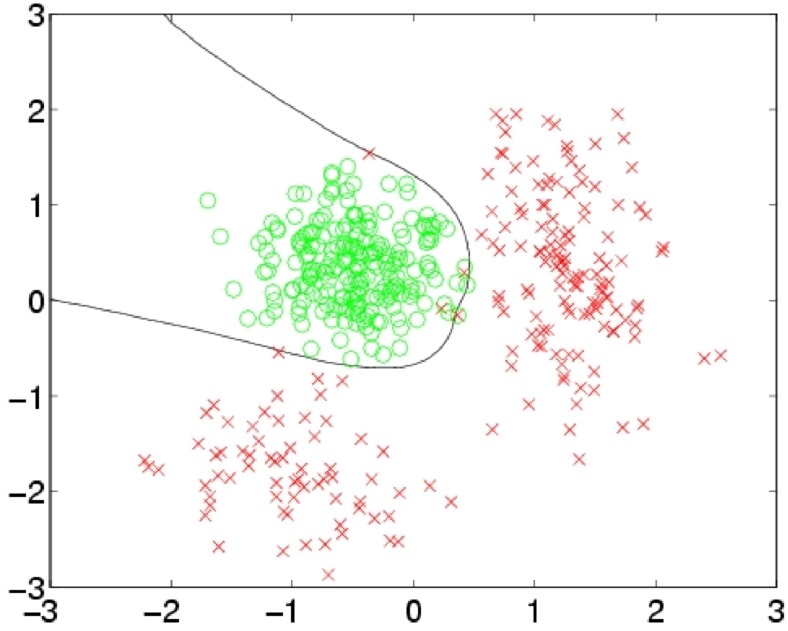


$$f(a) = \begin{cases} 1, \text{ for } a > \theta \\ 0, \text{ for } a \leq \theta \end{cases}$$

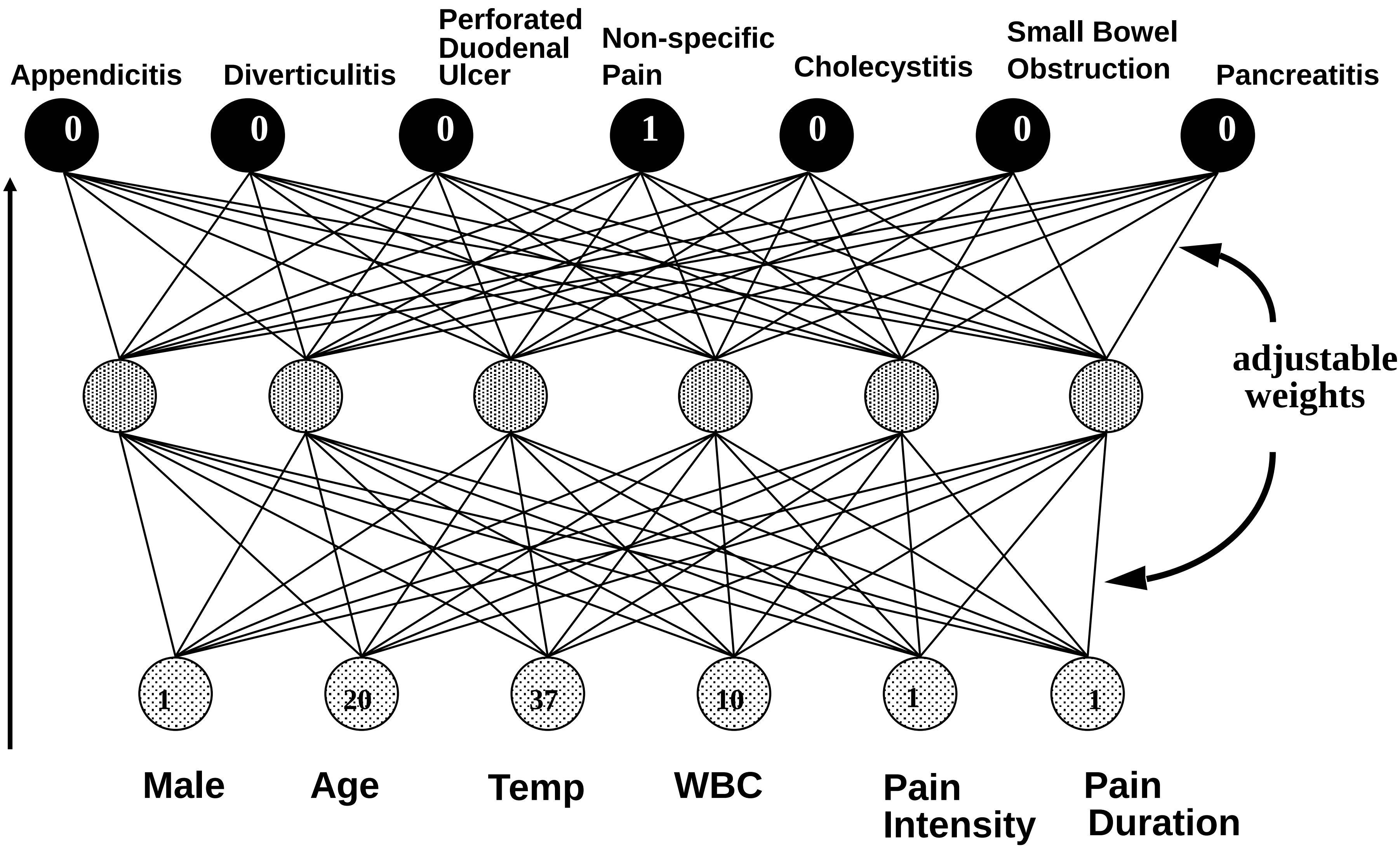
From perceptrons to multilayer perceptrons

Why?

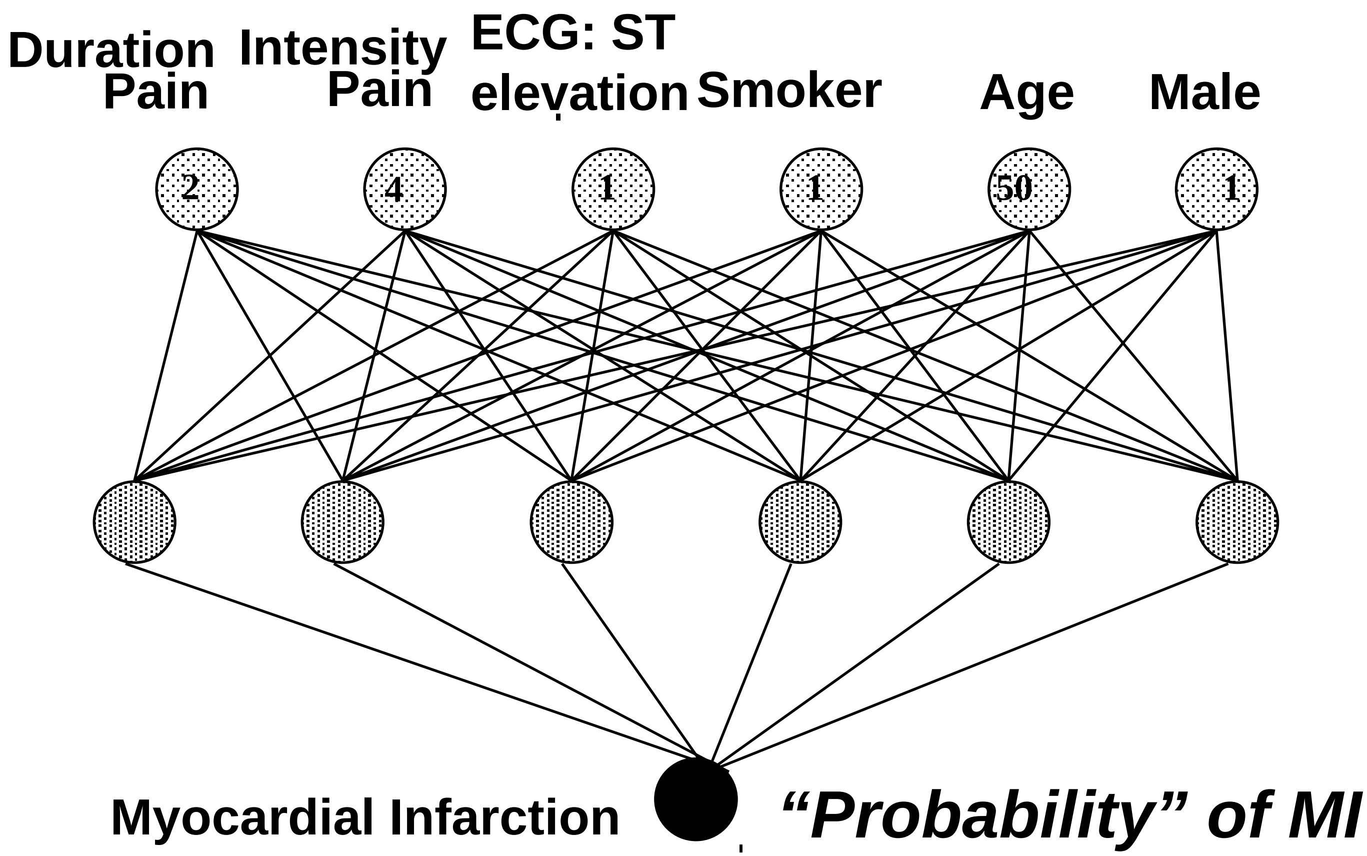




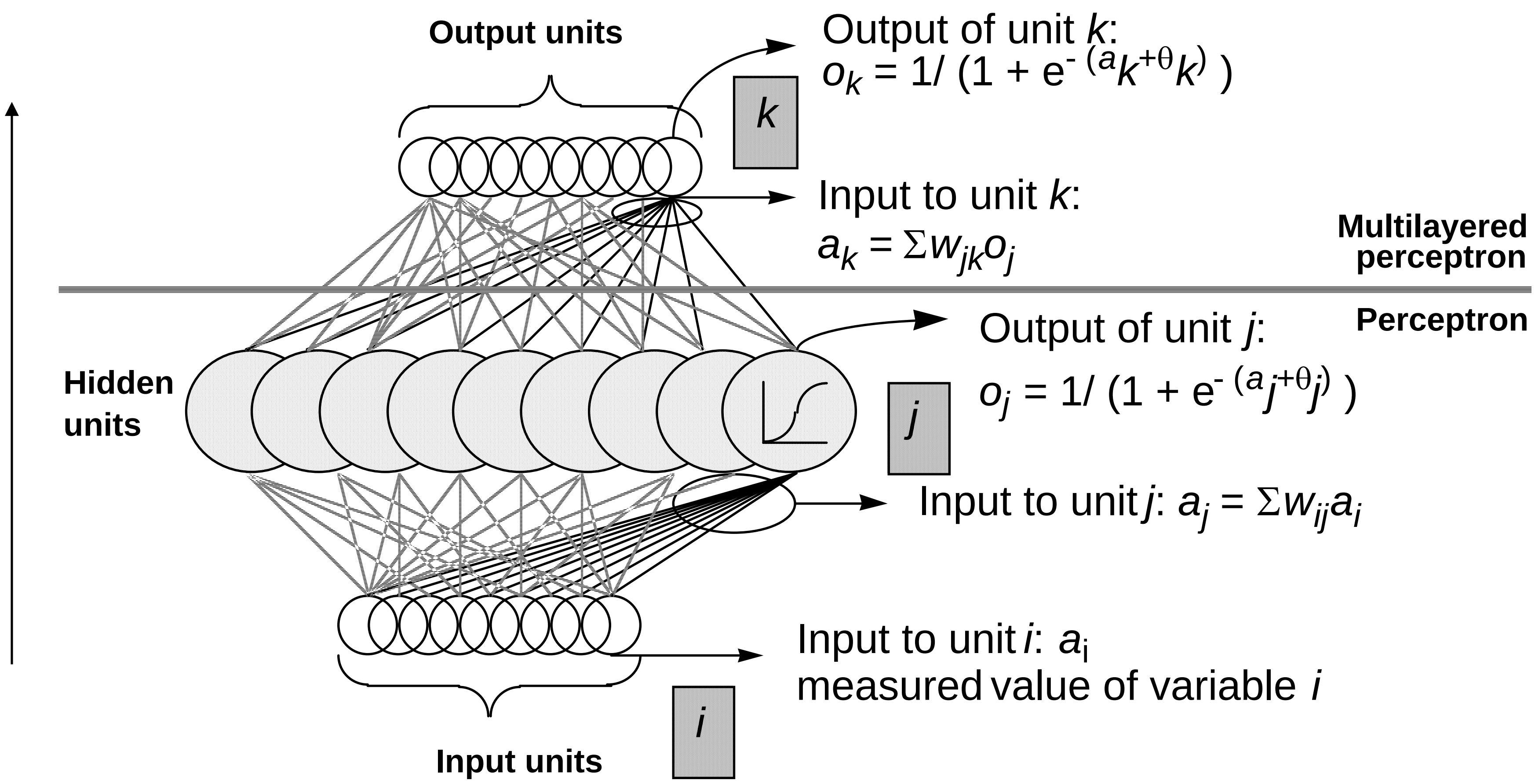
Abdominal Pain



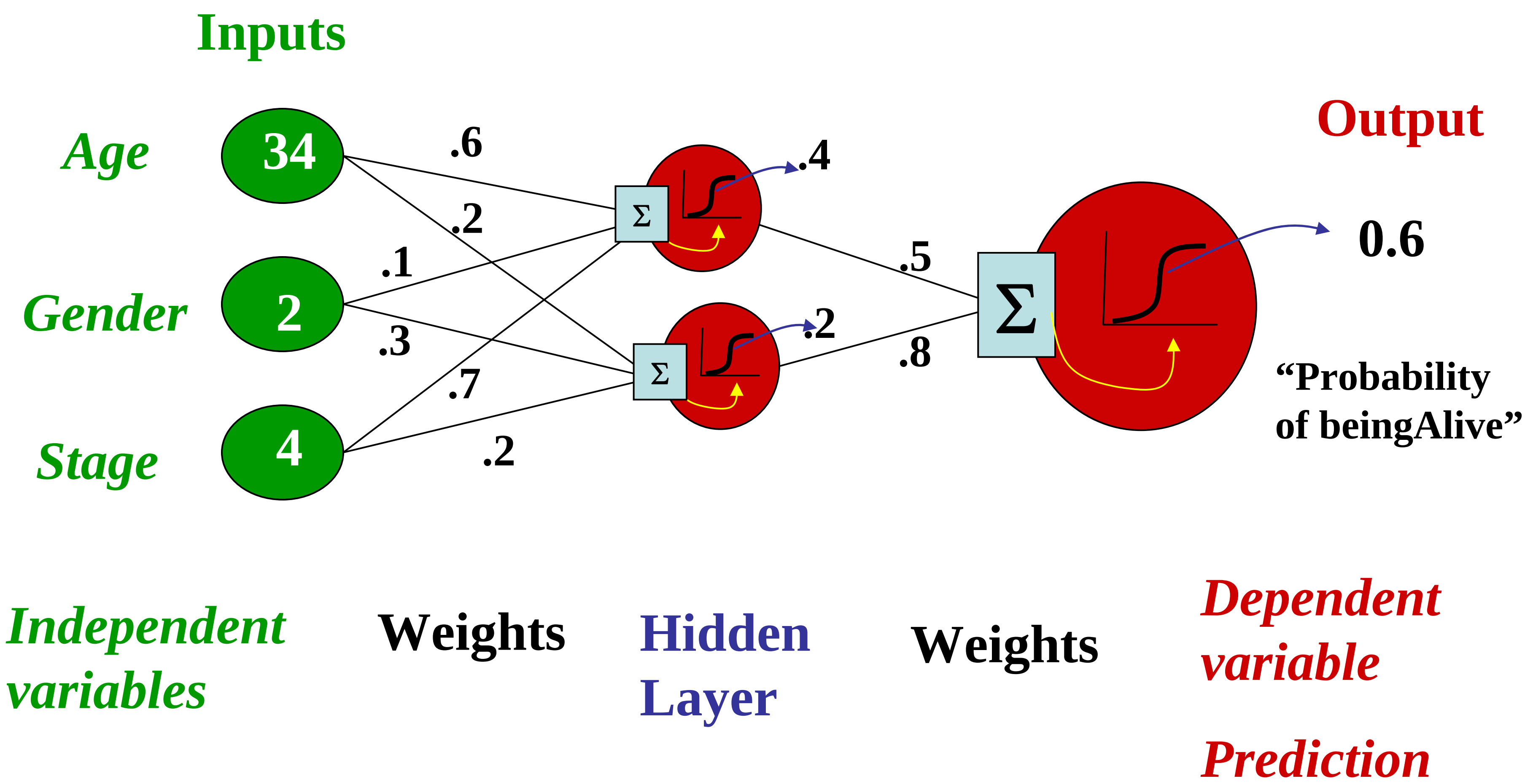
Heart Attack Network



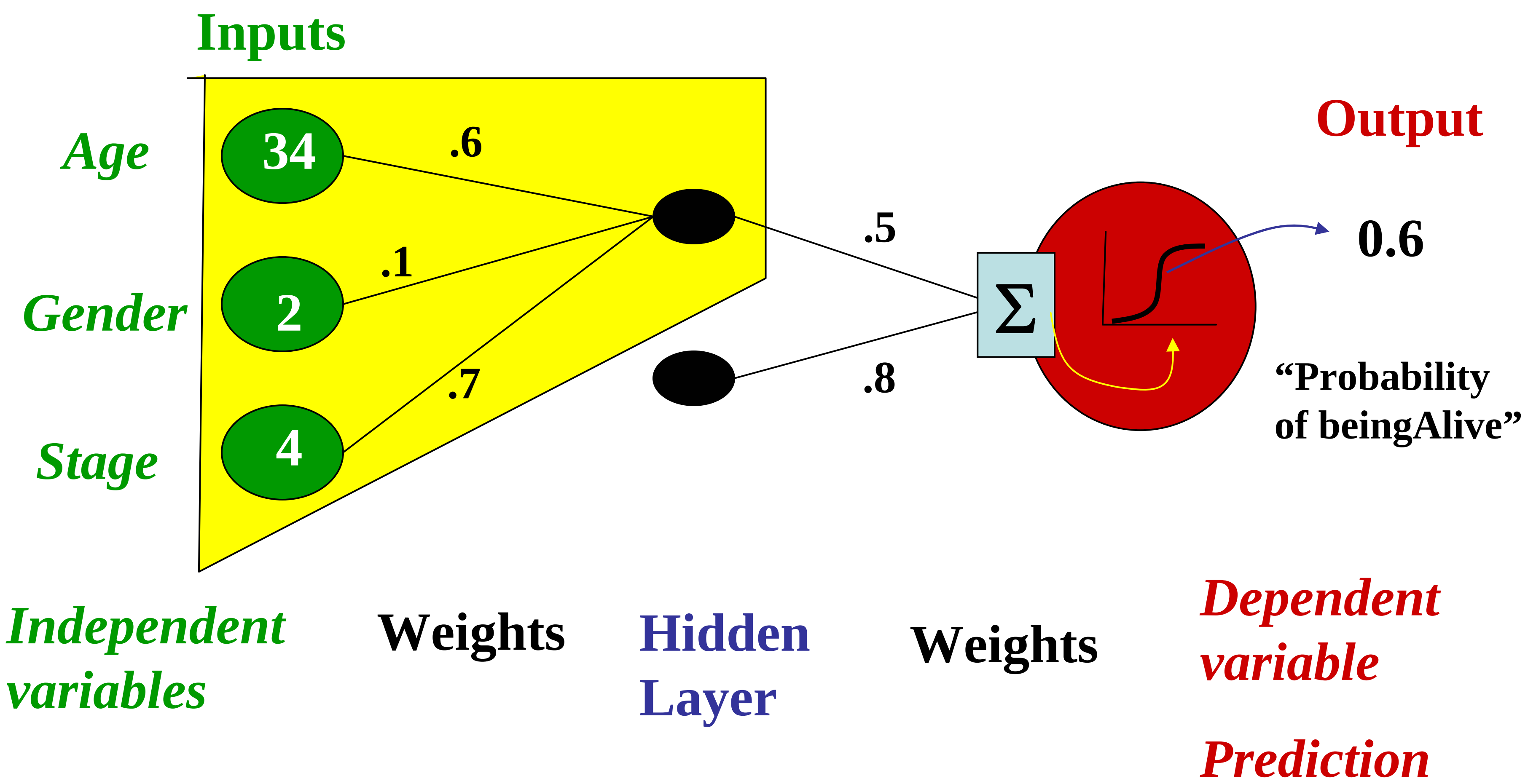
Multilayered Perceptrons

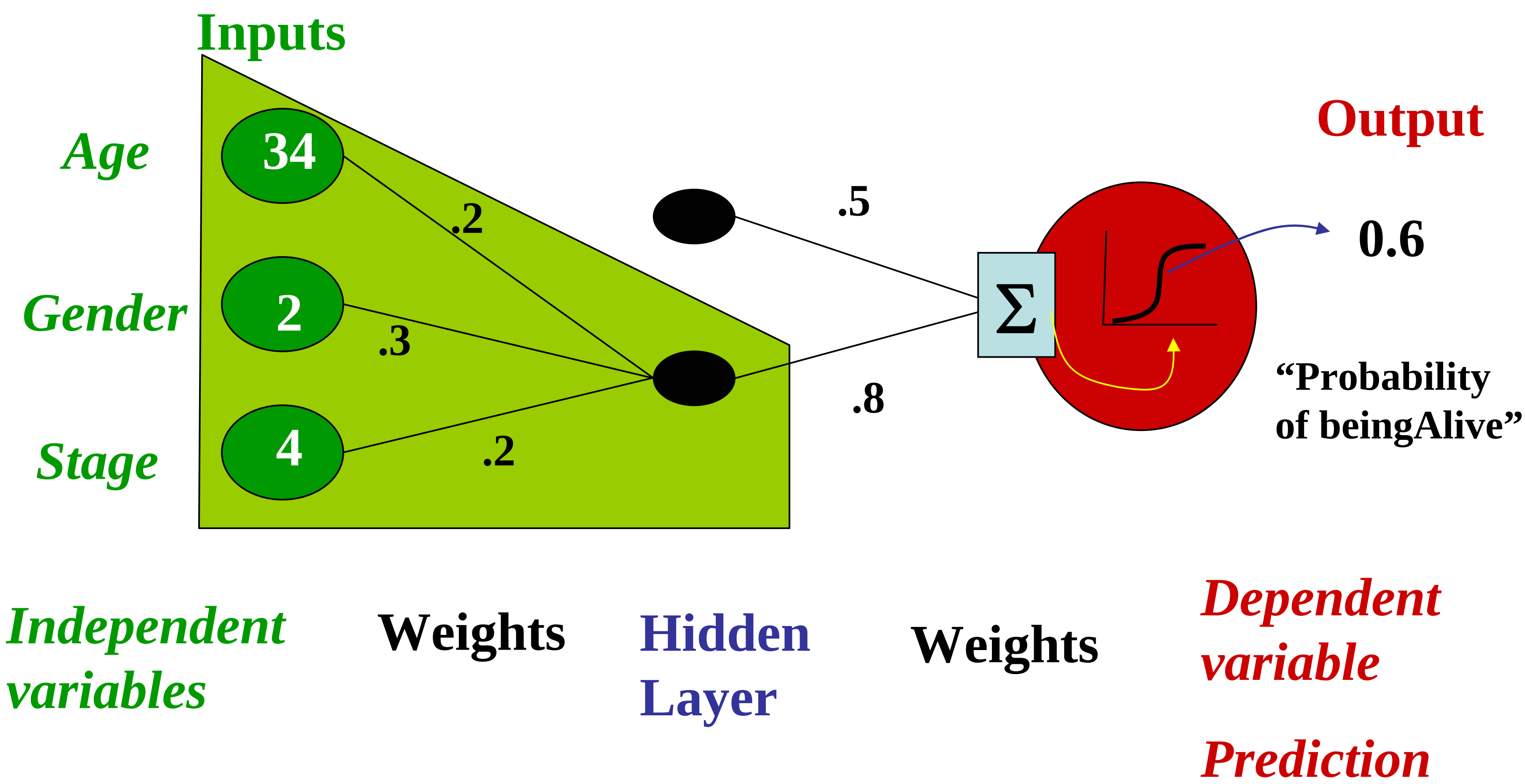


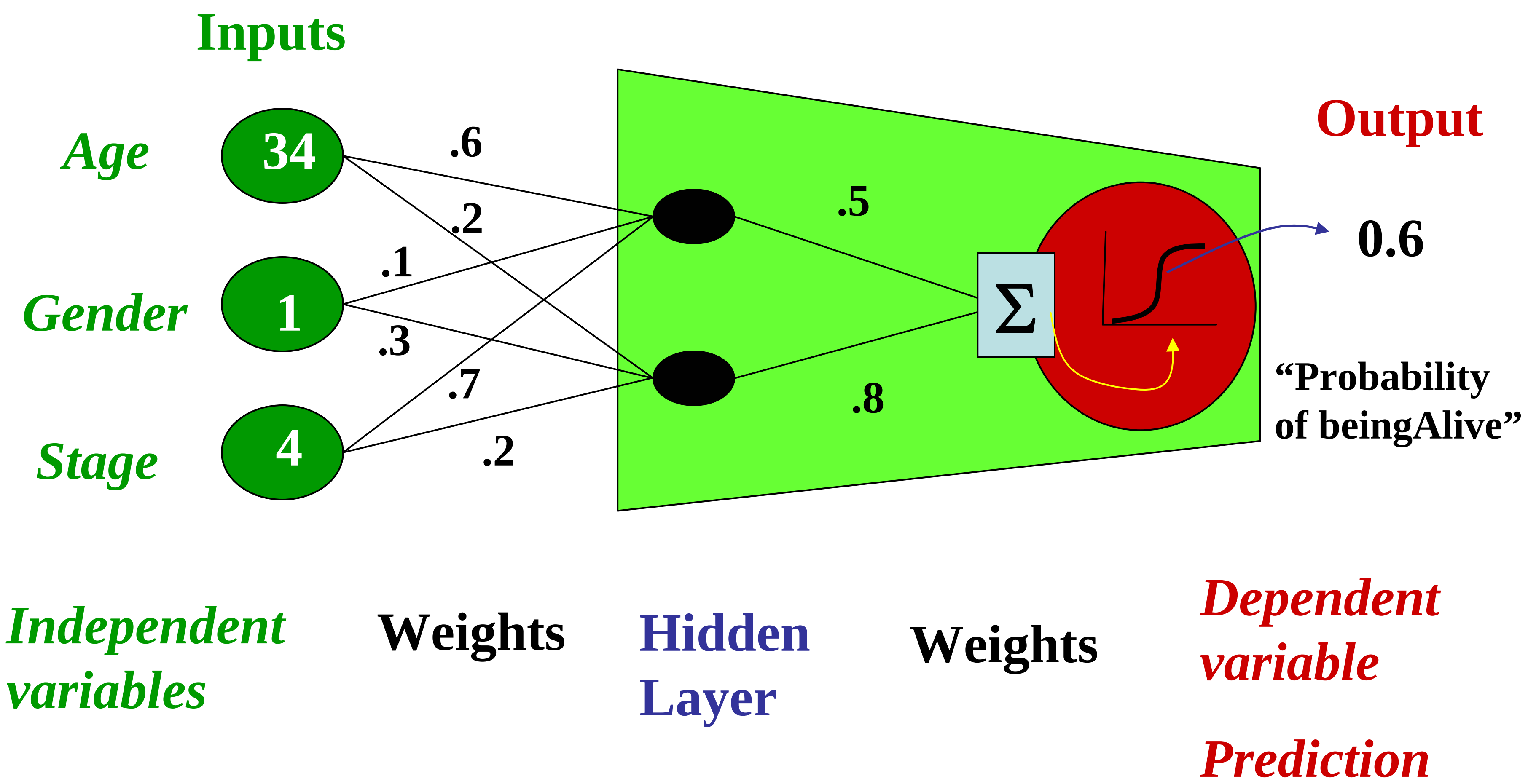
Neural Network Model



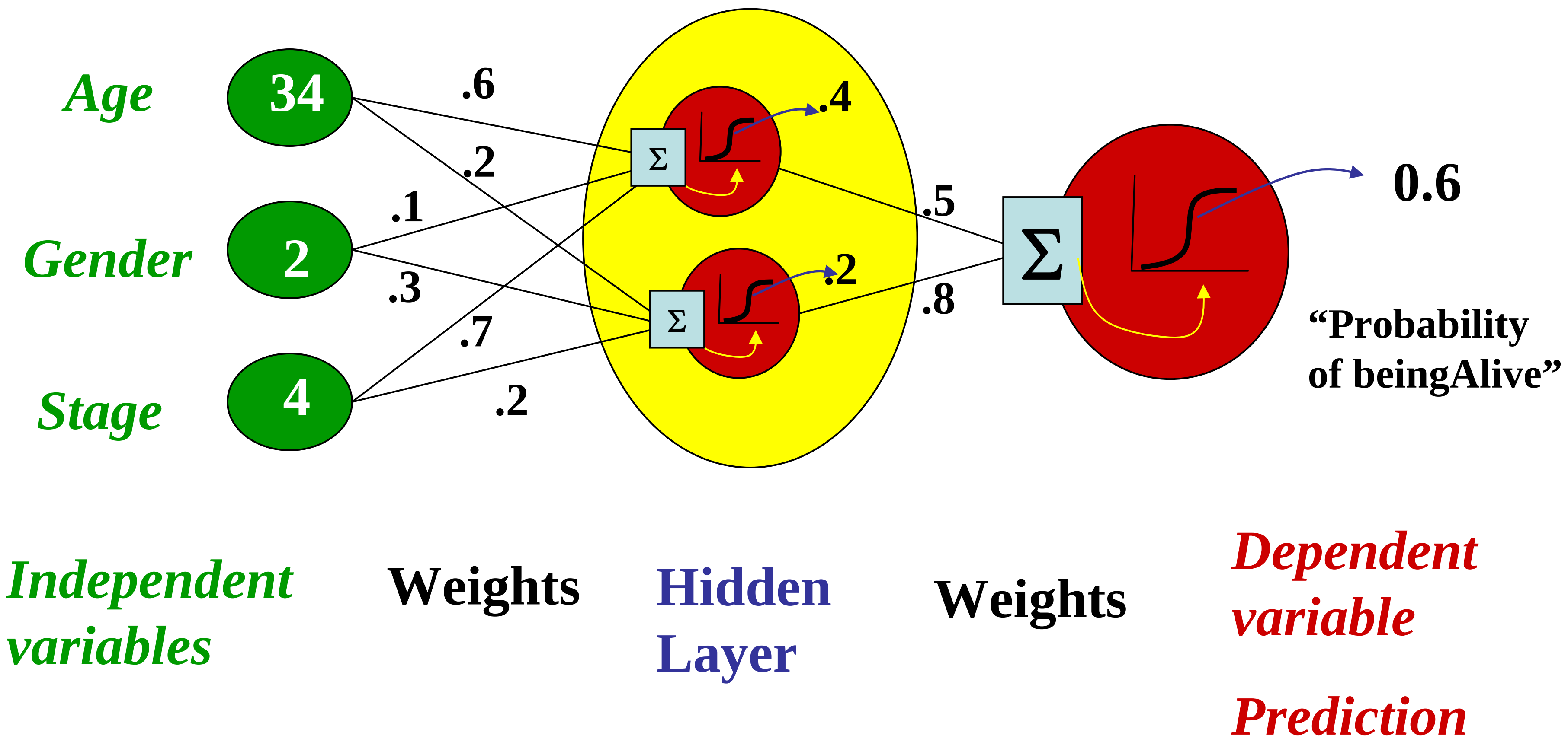
"Combined logistic models"



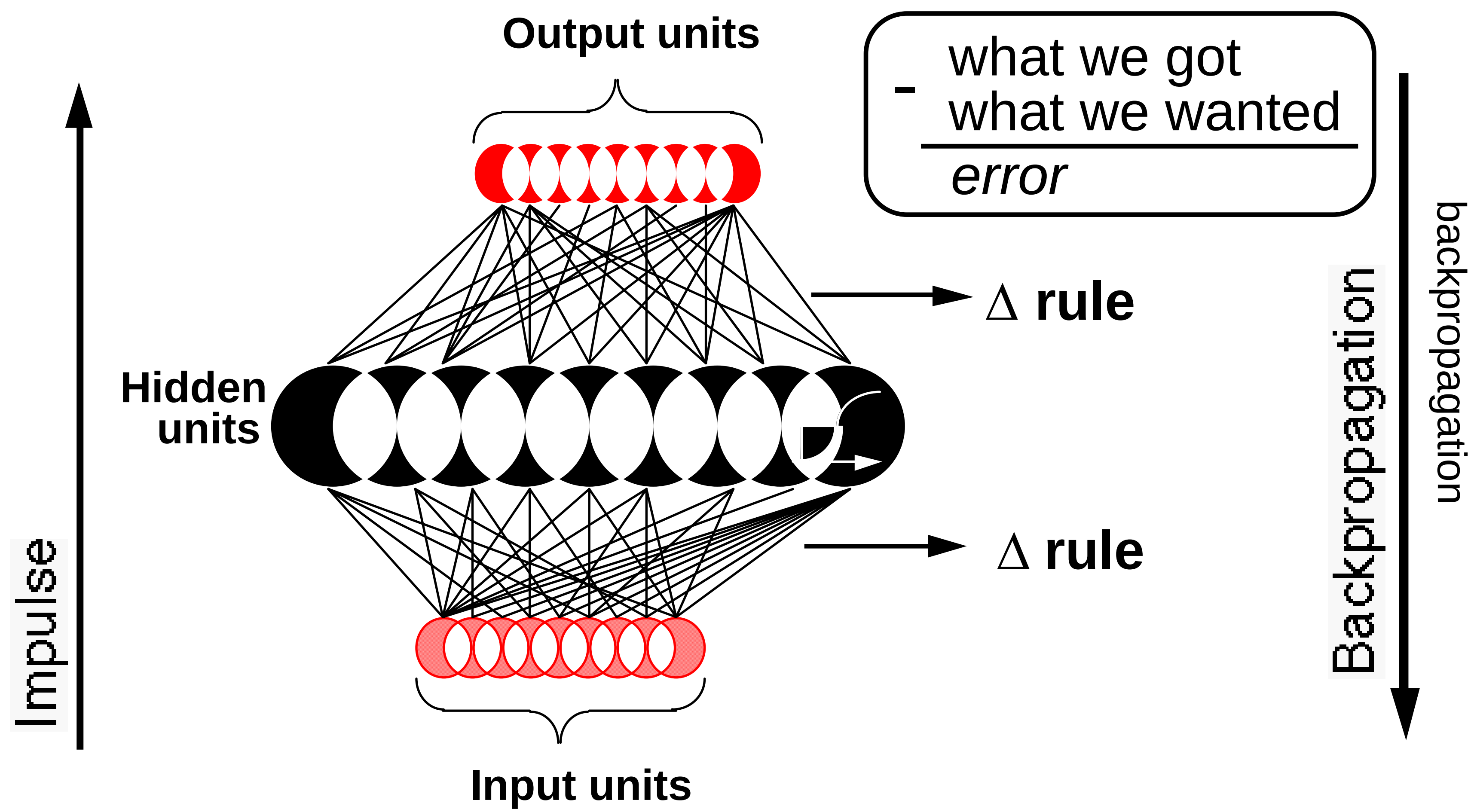

Inputs
Age
34
Gender
2
Stage
4
.2
.3
.2
.5
.8
Σ
Output
0.6
“Probability of beingAlive”
Independent variables
Weights
Hidden Layer
Weights
Dependent variable
Prediction


Inputs
Age
34
Gender
1
Stage
4
.6
.2
.1
.3
.7
.2
.5
.8
Σ
Output
0.6
"Probability of beingAlive"
Independent variables
Weights
Hidden Layer
Weights
Dependent variable
Prediction

Not really,
no target for hidden units...

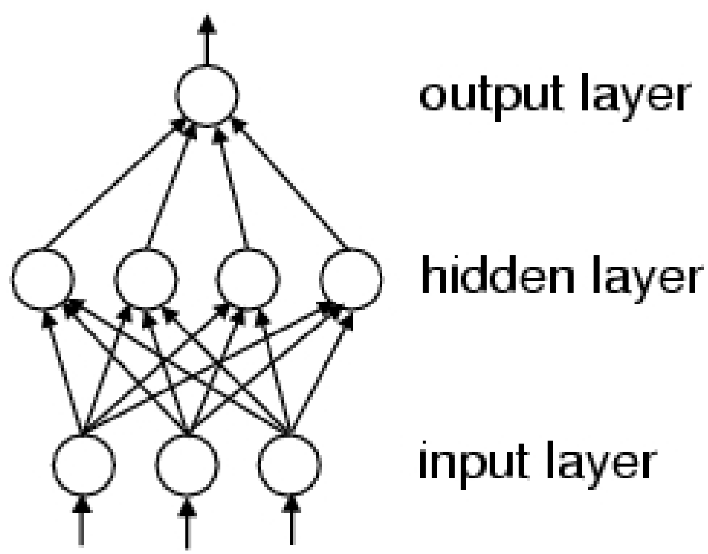


Hidden Units and Backpropagation

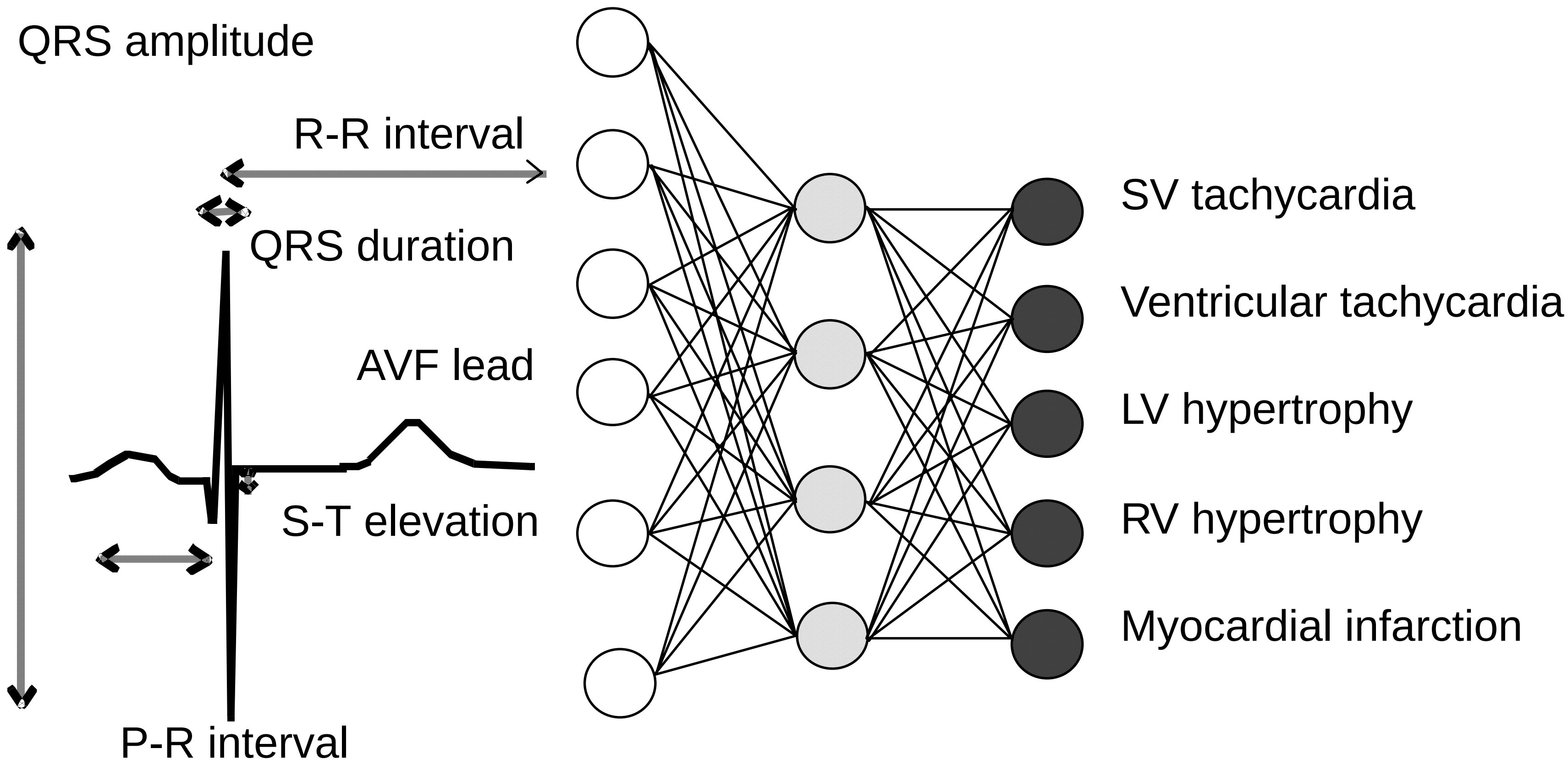


Multilayer perceptrons

- Sigmoidal hidden layer
- Can represent arbitrary decision regions
- Can be trained similar to perceptrons



ECG Interpretation

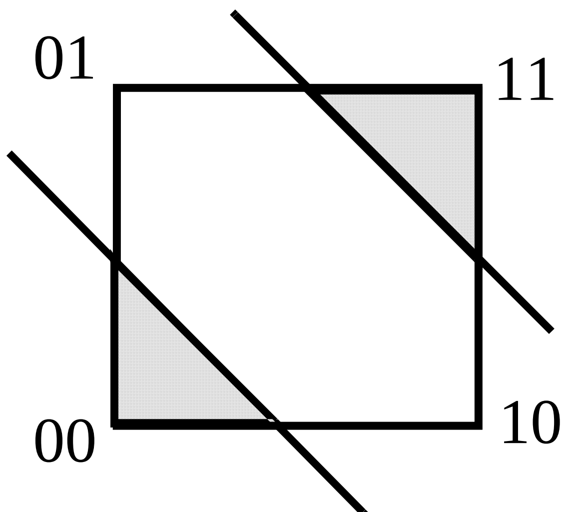


Linear Separation

Separate n-dimensional space using **<u>one</u>** (n - 1)-dimensional space

Meningitis
No cough
Headache

Flu
Cough
Headache

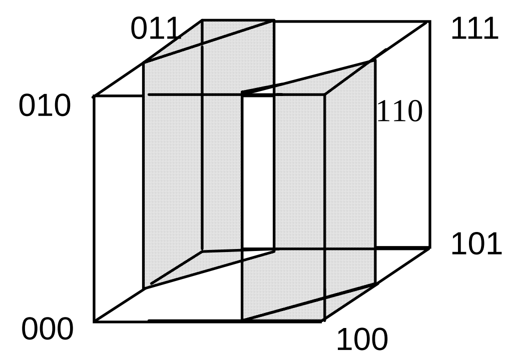


No treatment

Treatment

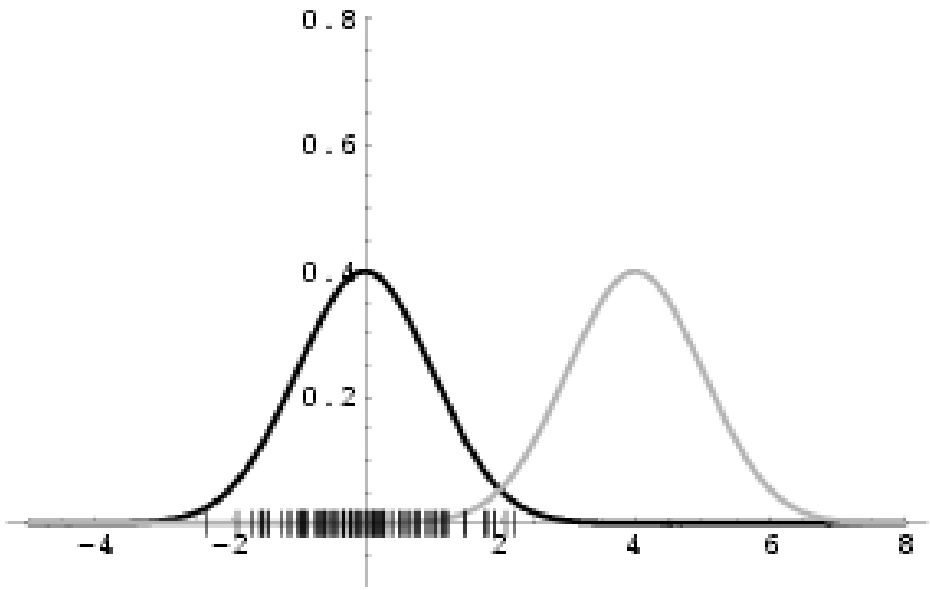
No disease
No cough
No headache

Pneumonia
Cough
No headache



Another way of thinking about this…

- Have data set $D = \{(x_i,t_i)\}$ drawn from probability distribution $P(x,t)$
- Model $P(x,t)$ given samples D by ANN with adjustable parameter w
- Statistics analogy:



Maximum Likelihood Estimation

- Maximize likelihood of data D
- Likelihood $L = \prod p(x_i,t_i) = \prod p(t_i|x_i)p(x_i)$
- Minimize -log $L = -\sum \log p(t_i|x_i) - \sum \log p(x_i)$
- Drop second term: does not depend on w
- Two cases: “regression” and classification

Likelihood for classification (ie categorical target)

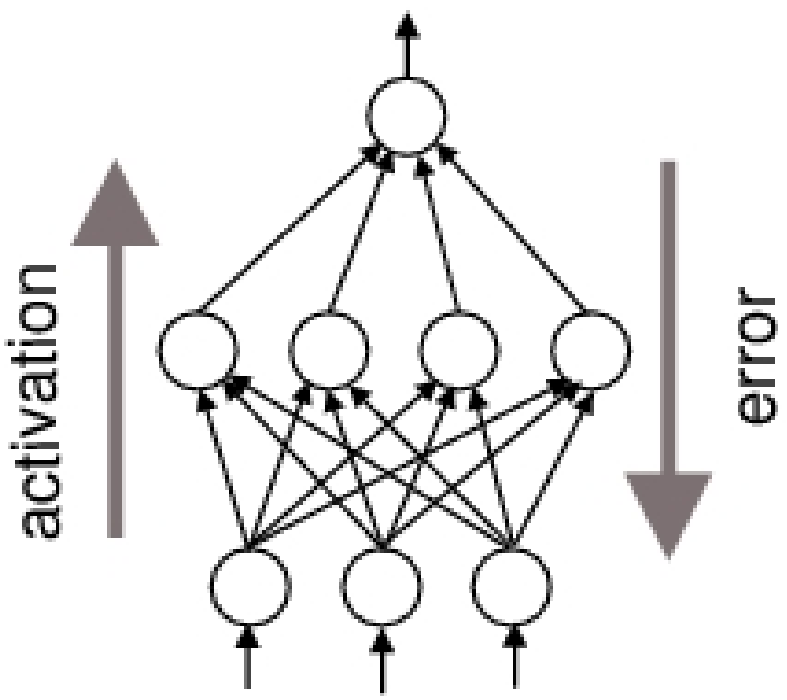
- For classification, targets *t* are class labels
- Minimize $-\Sigma \log p(t_i|x_i)$
- $p(t_i|x_i) = \mathrm{y}(x_i,w)^{t_i}(1-\mathrm{y}(x_i,w))^{1-t_i} \Rightarrow$
  $-\log p(t_i|x_i) = -t_i \log \mathrm{y}(x_i,w) - (1-t_i) * \log(1-y(x_i,w))$
- Minimizing $-\log L$ equivalent to minimizing
  $-[\Sigma\, t_i \log \mathrm{y}(x_i,w) + (1-t_i) * \log(1-y(x_i,w))]$
  (*cross-entropy error*)

Likelihood for "regression" (ie continuous target)

- For regression, targets t are real values
- Minimize $-\Sigma \log p(t_i|x_i)$
- $p(t_i|x_i) = 1/Z \exp(-(y(x_i,w) - t_i)^2/(2\sigma^2)) \Rightarrow$
  $-\log p(t_i|x_i) = 1/(2\sigma^2)\ (y(x_i,w) - t_i)^2 + \log Z$
- $y(x_i,w)$ is network output
- Minimizing $-\log L$ equivalent to minimizing
  $\Sigma\ (y(x_i,w) - t_i)^2$ (*sum-of-squares error*)

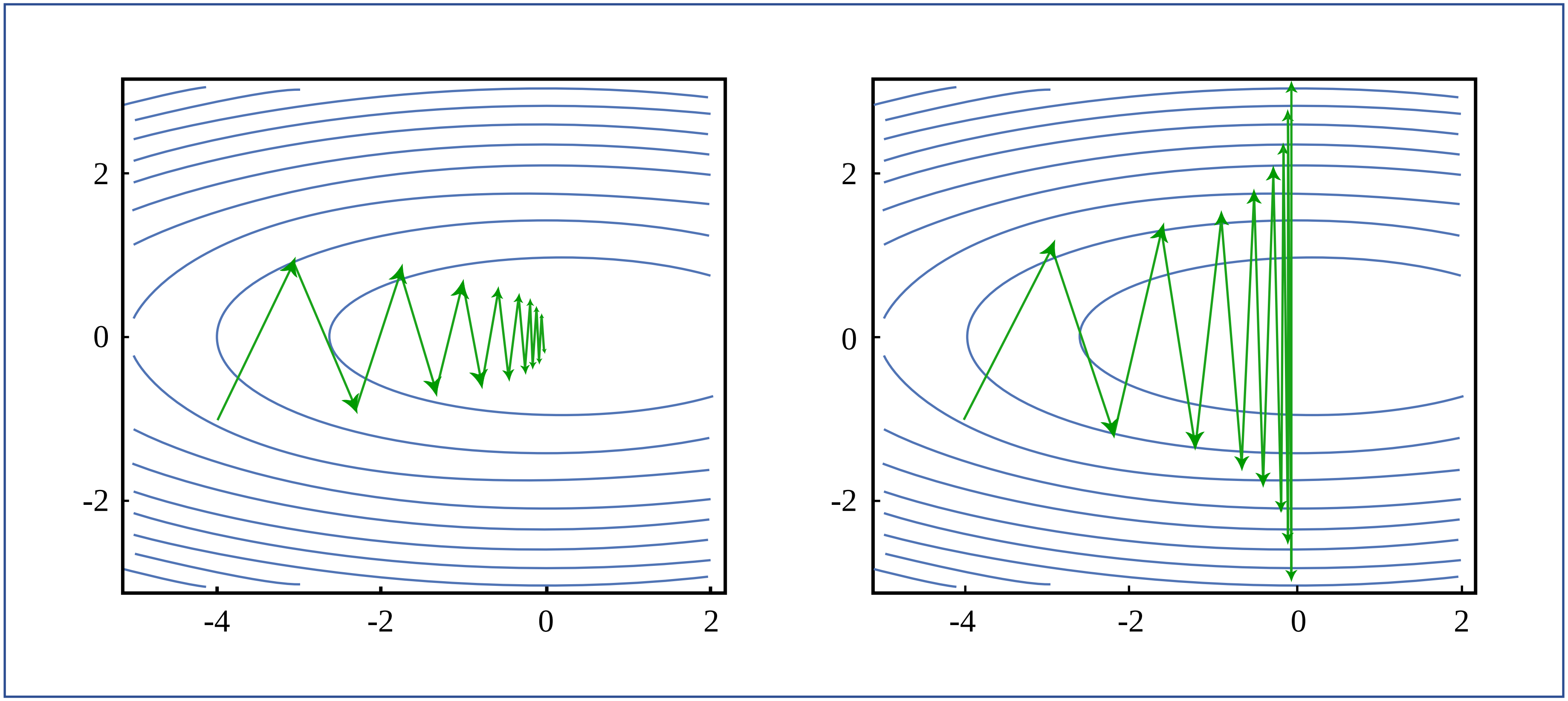
Backpropagation algorithm

- Minimizing error function by gradient descent:
  $w_{k+1} = w_k - \eta\, \mathrm{grad}_w\, E$
- Iterative gradient calculation by propagating error signals



Backpropagation algorithm

Problem: how to set learning rate η ?



Figures by MIT OCW.

Better: use more advanced minimization algorithms (second-order information)

Backpropagation algorithm

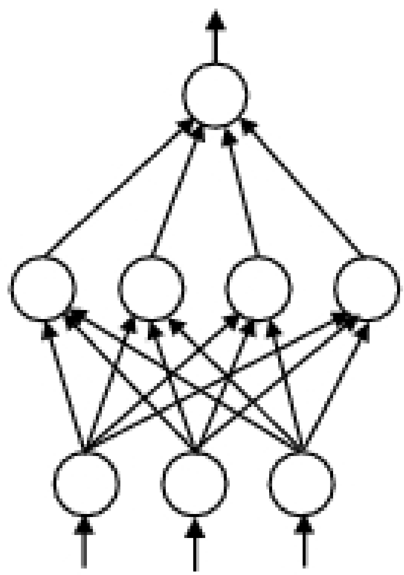
Classification | Regression

cross-entropy | sum-of-squares

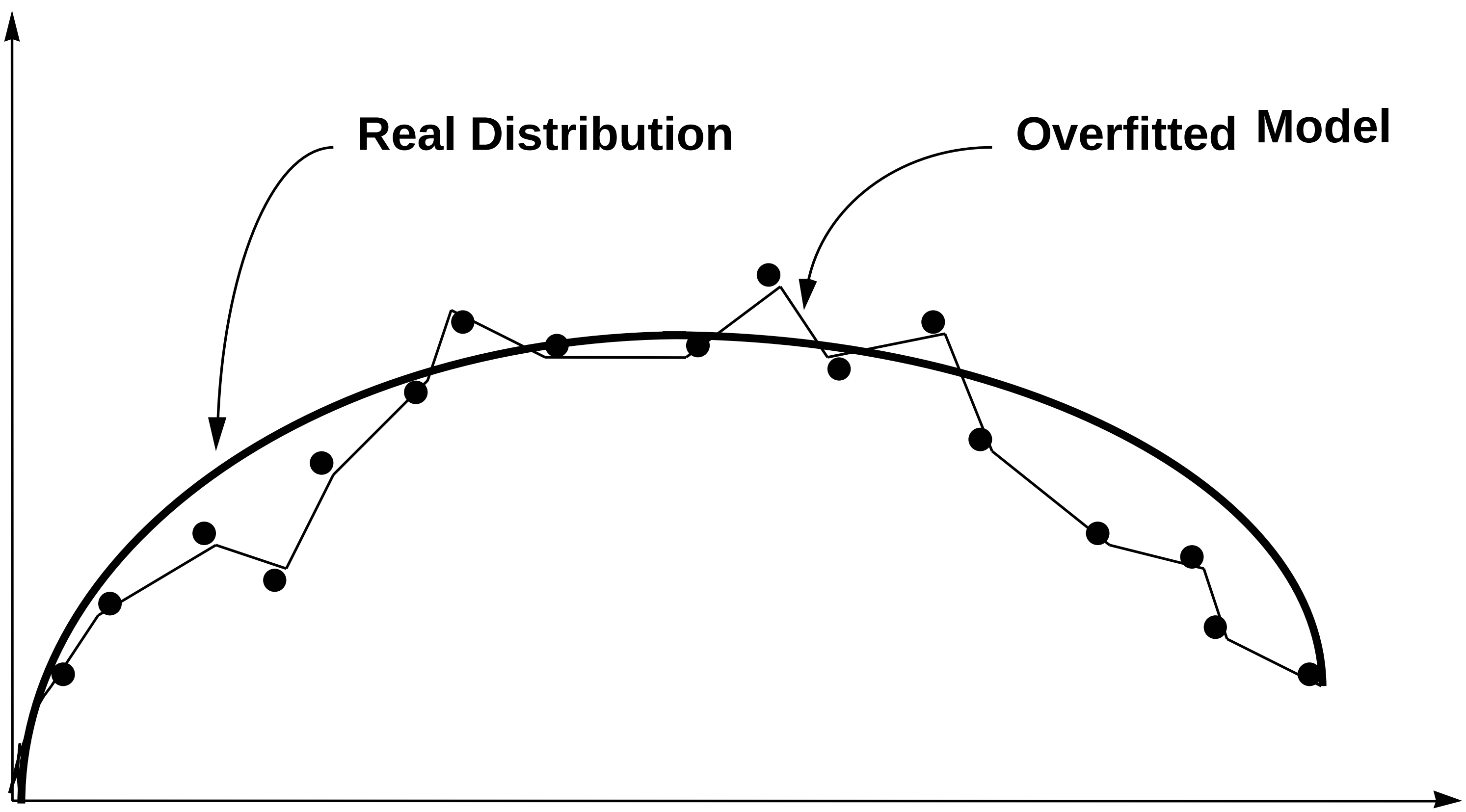
sigmoidal neuron | linear neuron

sigmoidal neurons | sigmoidal neurons

linear neurons | linear neurons



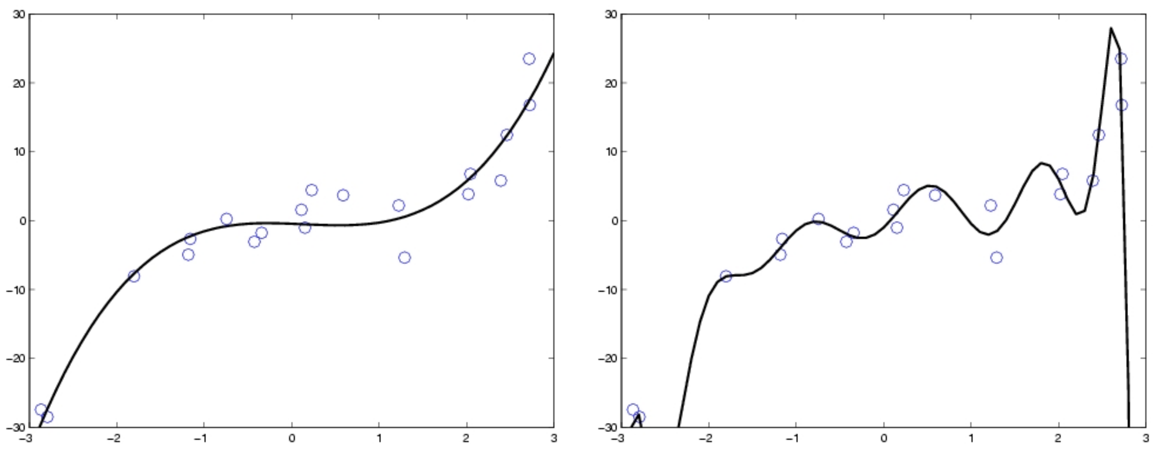
Overfitting



Improving generalization

Problem: memorizing (x,t) combinations ("overtraining")

0.7	0.5	0
-0.5	0.9	1
-0.2	-1.2	1
0.3	0.6	1
-0.2	0.5	?

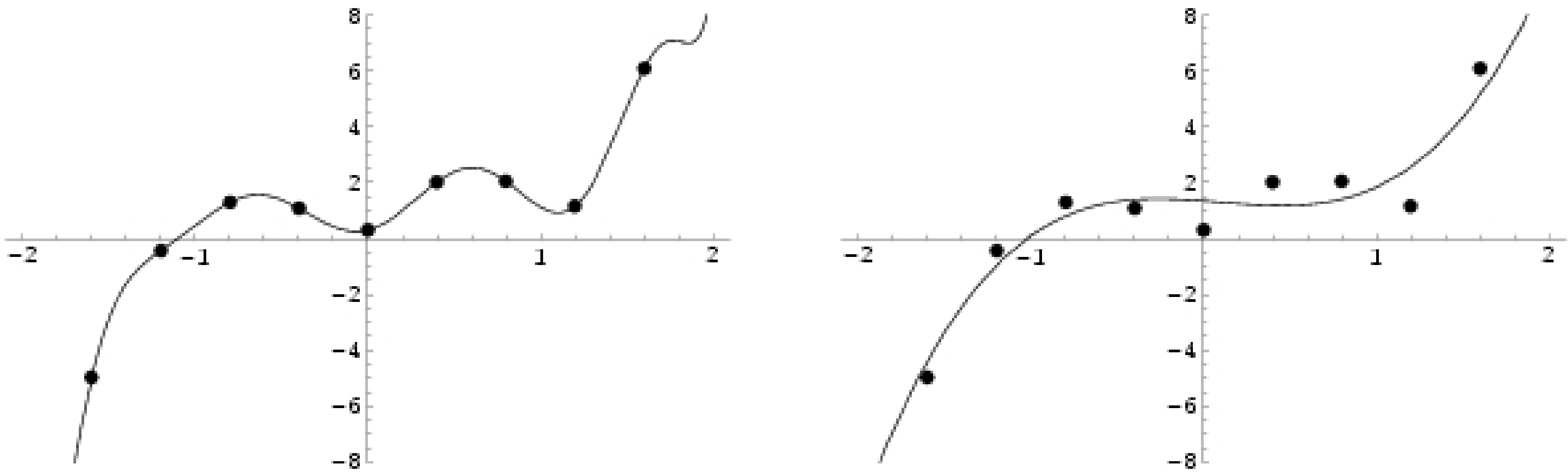


Improving generalization

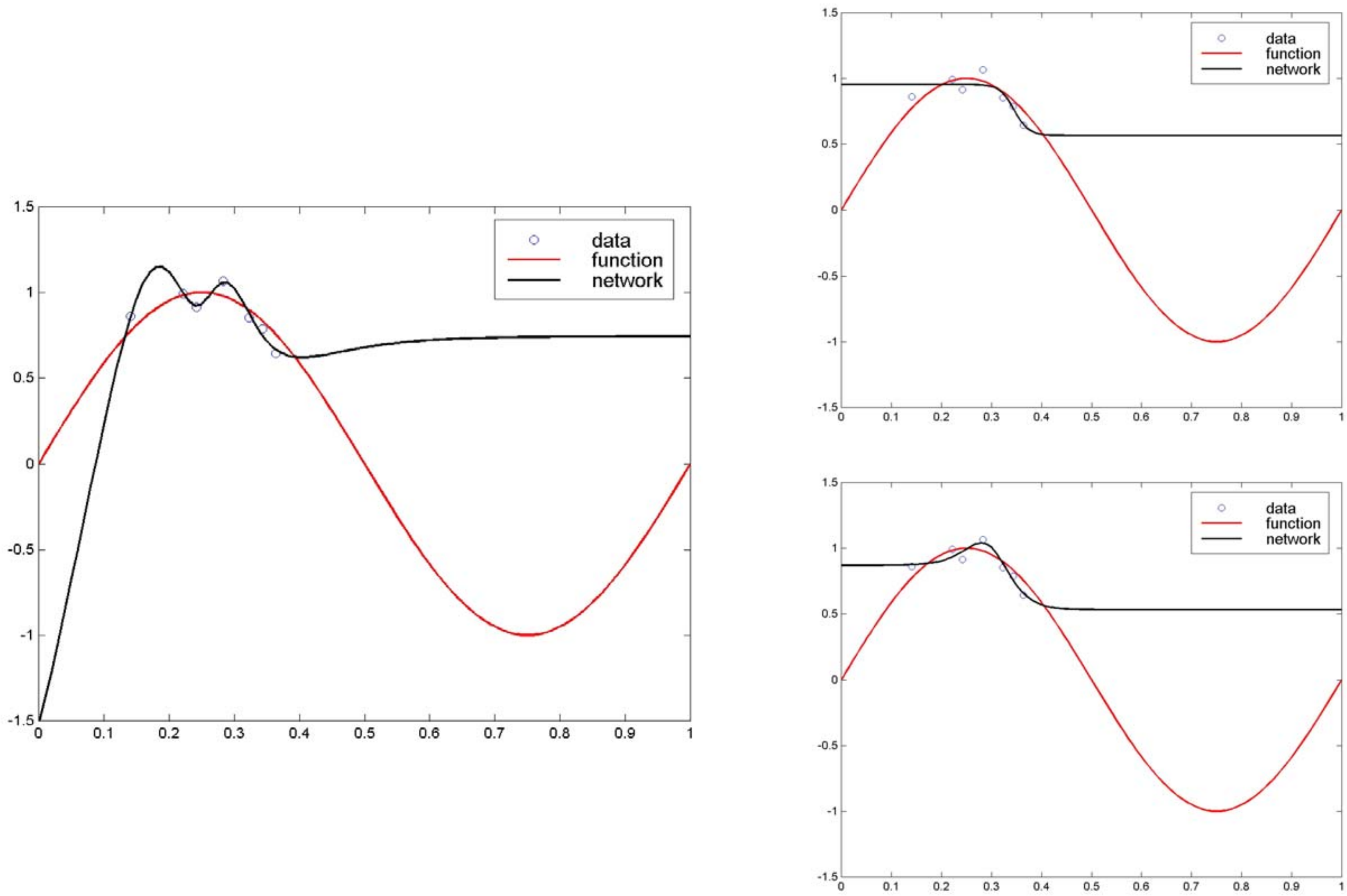
- Need test set to judge performance
- Goal: represent information in data set, not noise
- How to improve generalization?
  - Limit network topology
  - Early stopping
  - Weight decay

Limit network topology

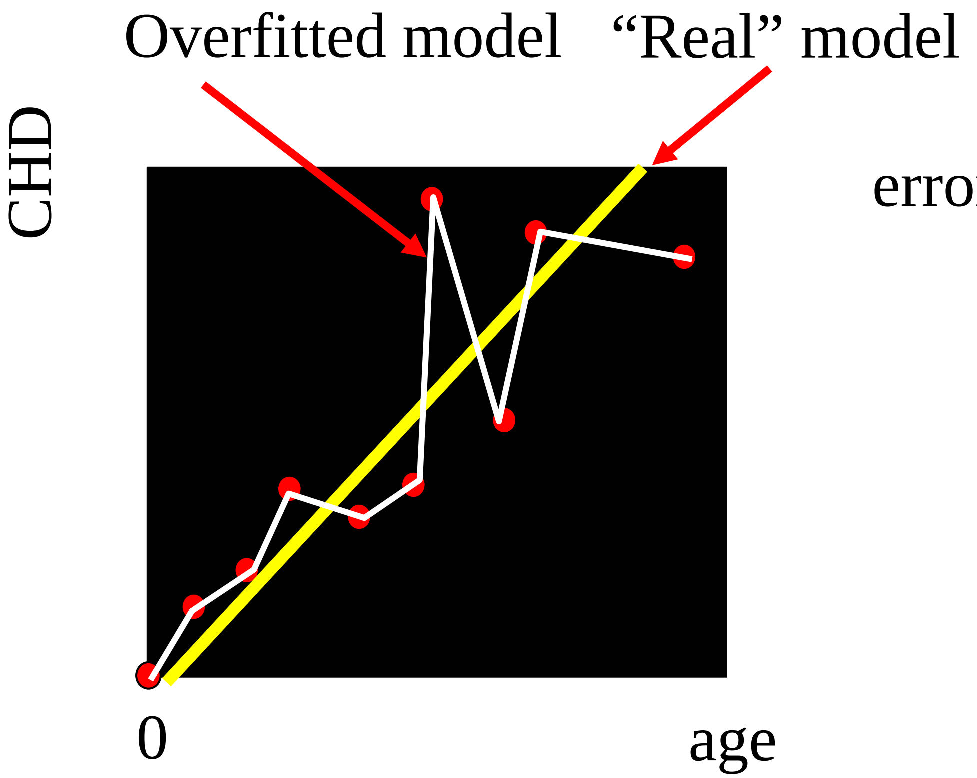
- Idea: fewer weights $\Rightarrow$ less flexibility
- Analogy to polynomial interpolation:

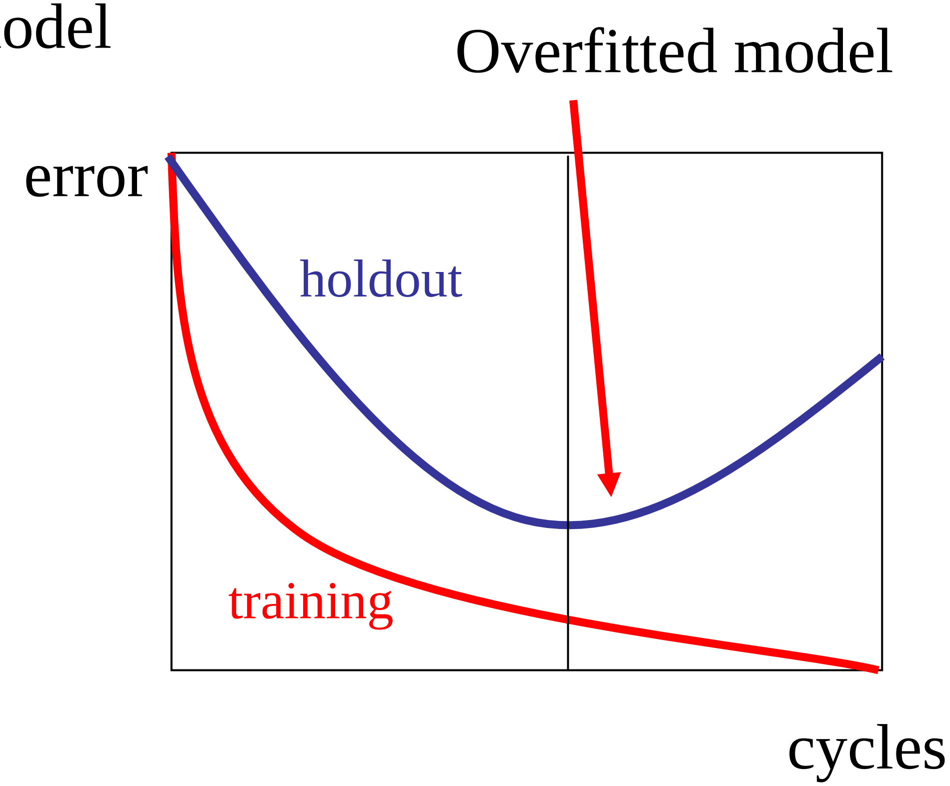


Limit network topology



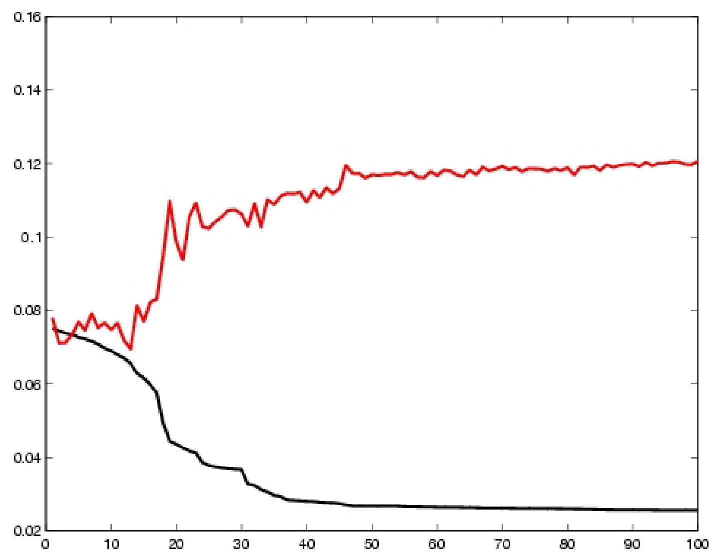
Early Stopping



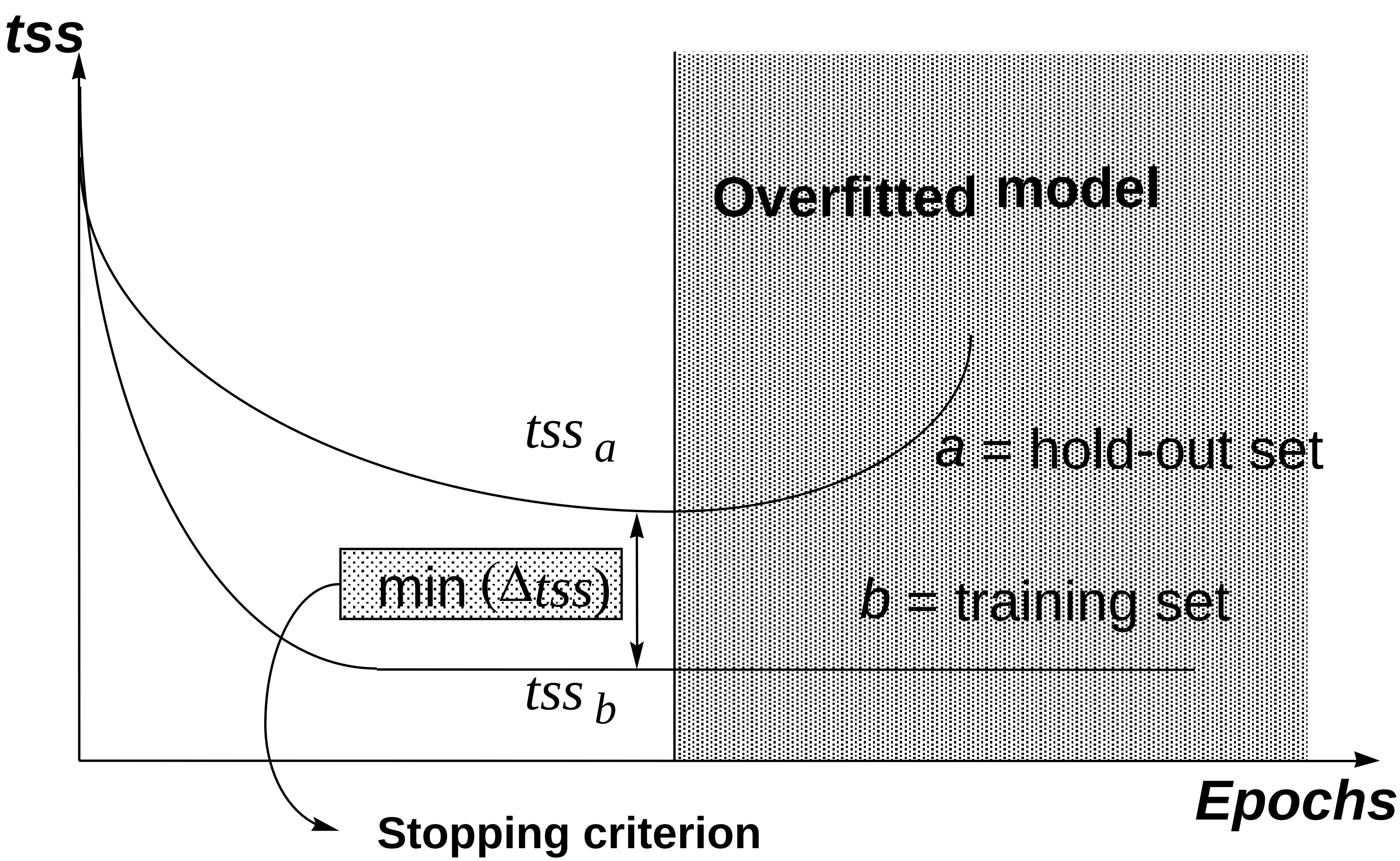


Early stopping

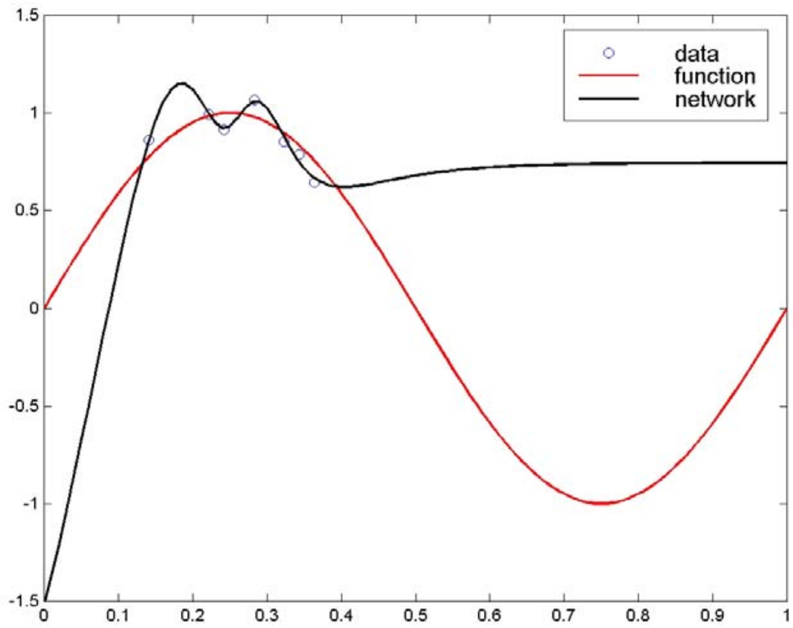
- Idea: stop training when information (but not noise) is modeled
- Need *hold-out set* to determine when to stop training

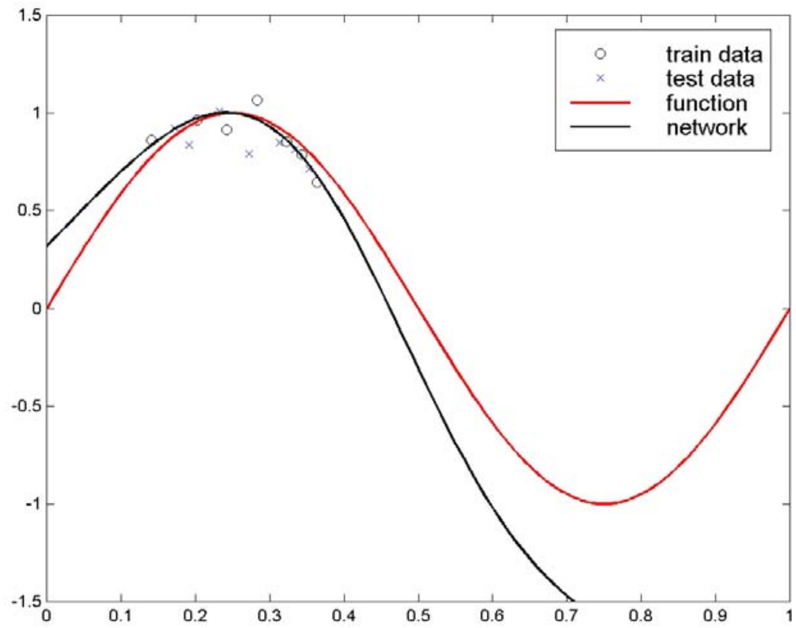


Overfitting



Early stopping

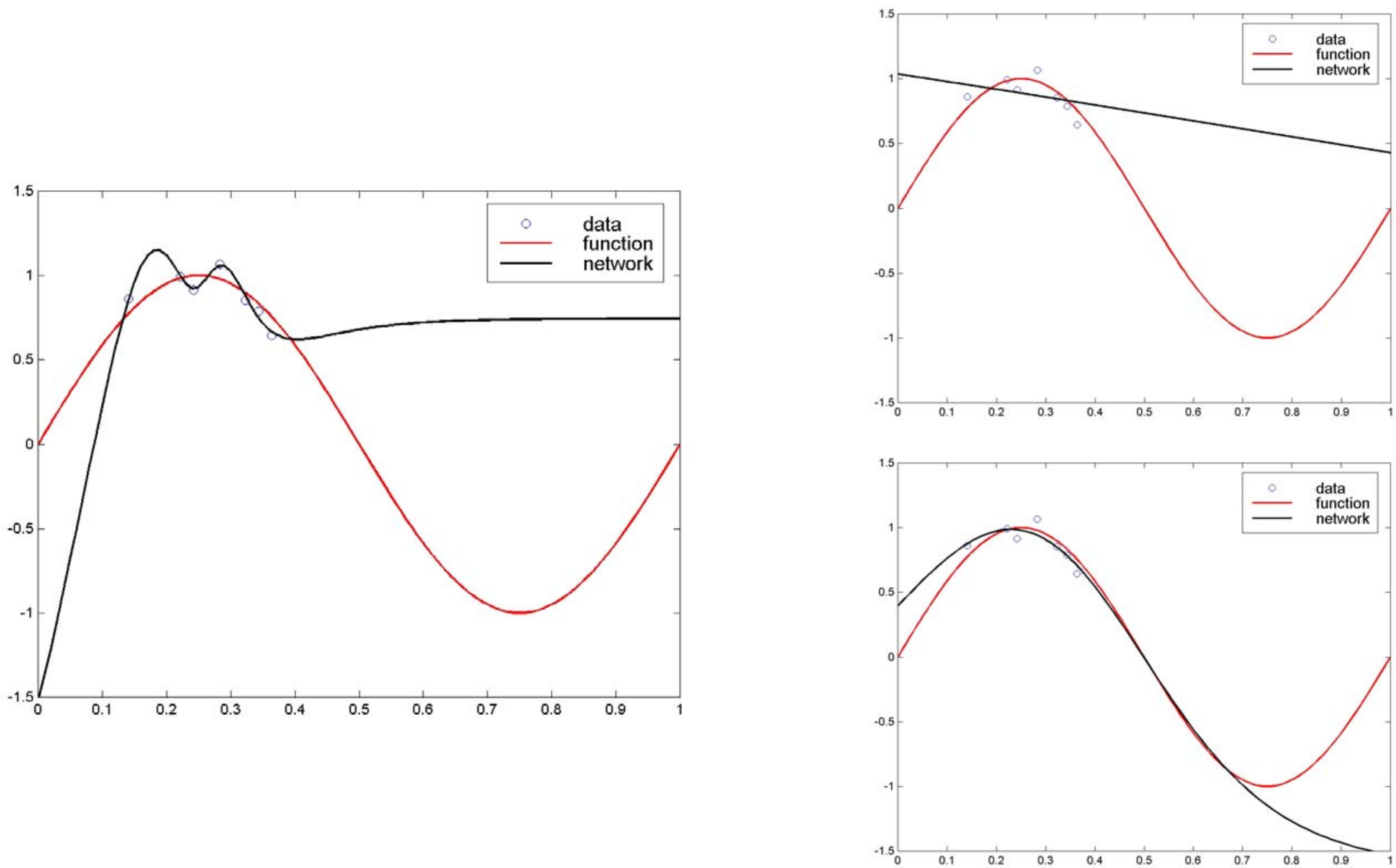




Weight decay

- Idea: control smoothness of network output by controlling size of weights
- Add term $\alpha||w||^2$ to error function

Weight decay



Bayesian perspective

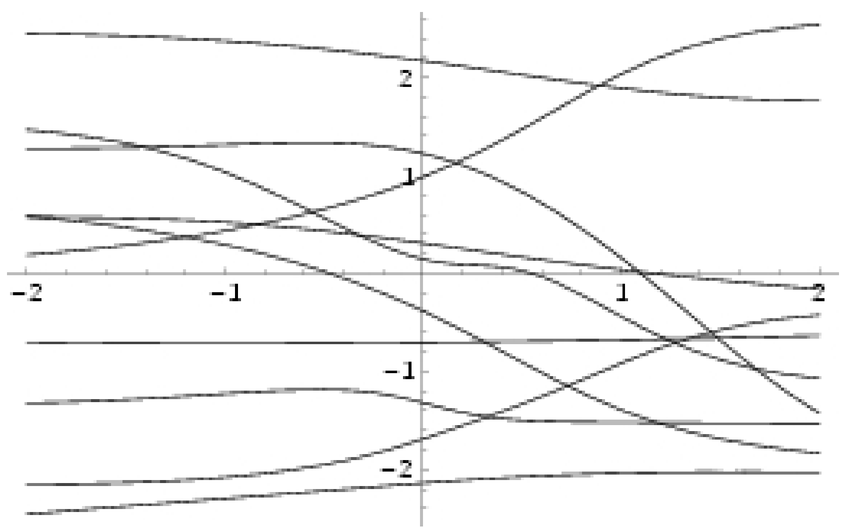
- Error function minimization corresponds to maximum likelihood (ML) estimate: single best solution w_{ML}
- Can lead to overtraining
- Bayesian approach: consider weight posterior distribution $p(w|D)$.

Bayesian perspective

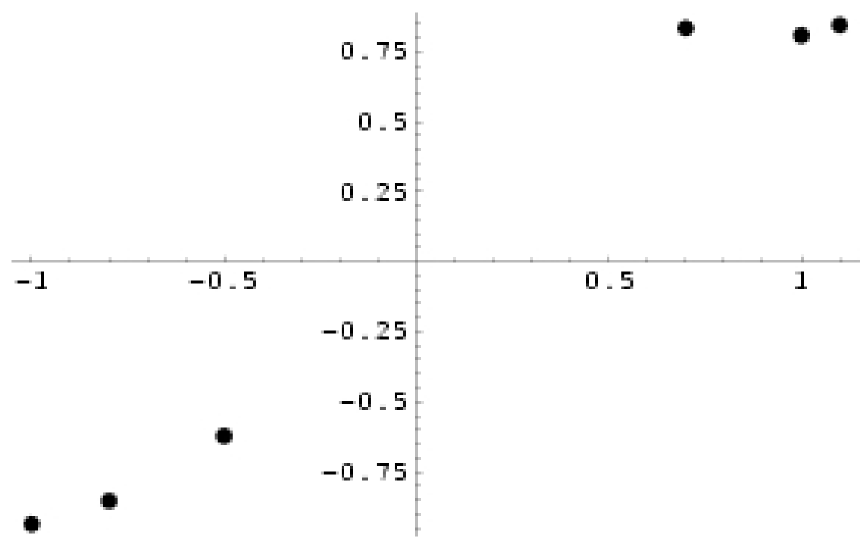
- Posterior = likelihood * prior
- $p(w|D) = p(D|w)\, p(w)/p(D)$
- Two approaches to approximating $p(w|D)$:
  - Sampling
  - Gaussian approximation

Sampling from $p(w|D)$

prior

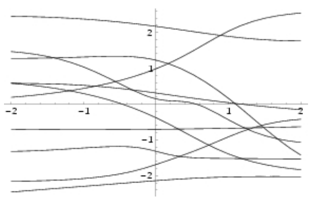


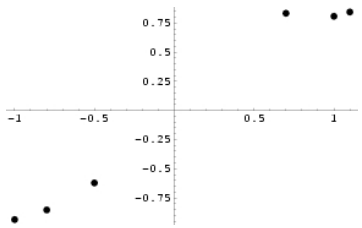
likelihood

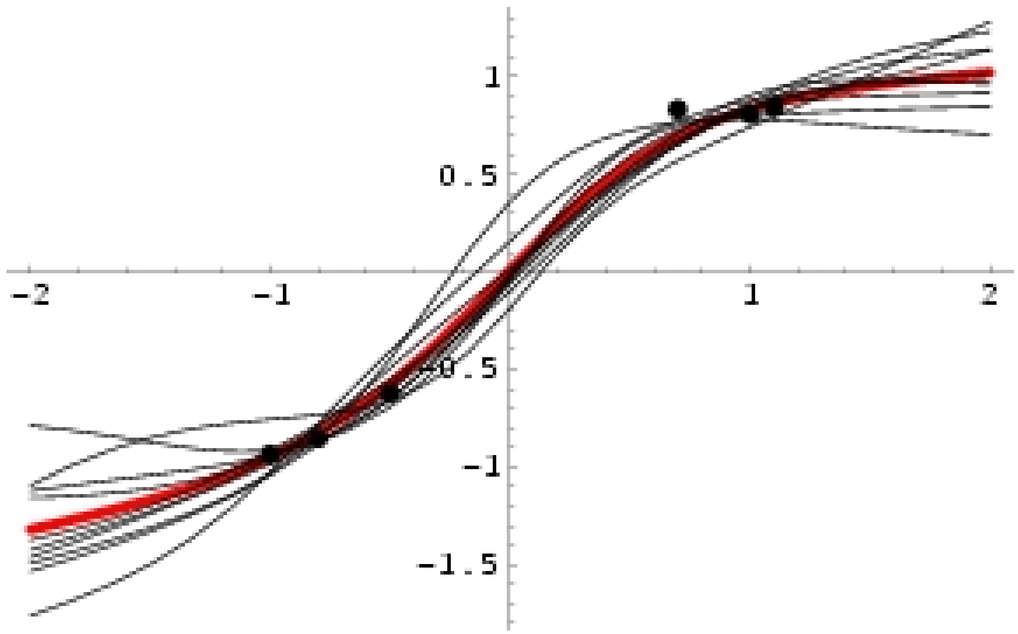


Sampling from $p(w|D)$

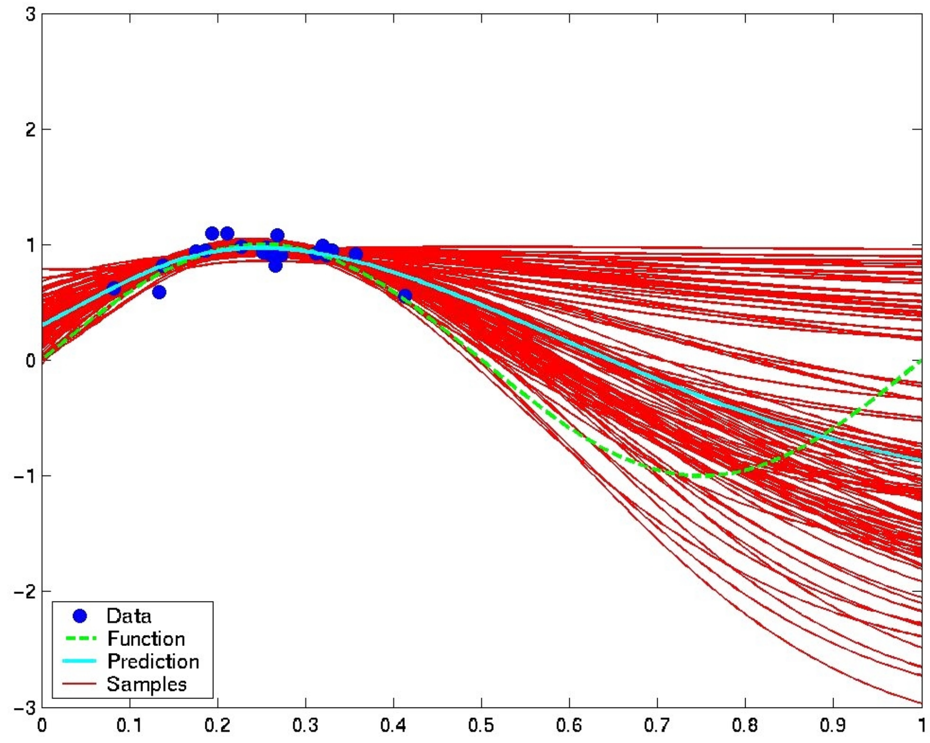
prior * likelihood = posterior



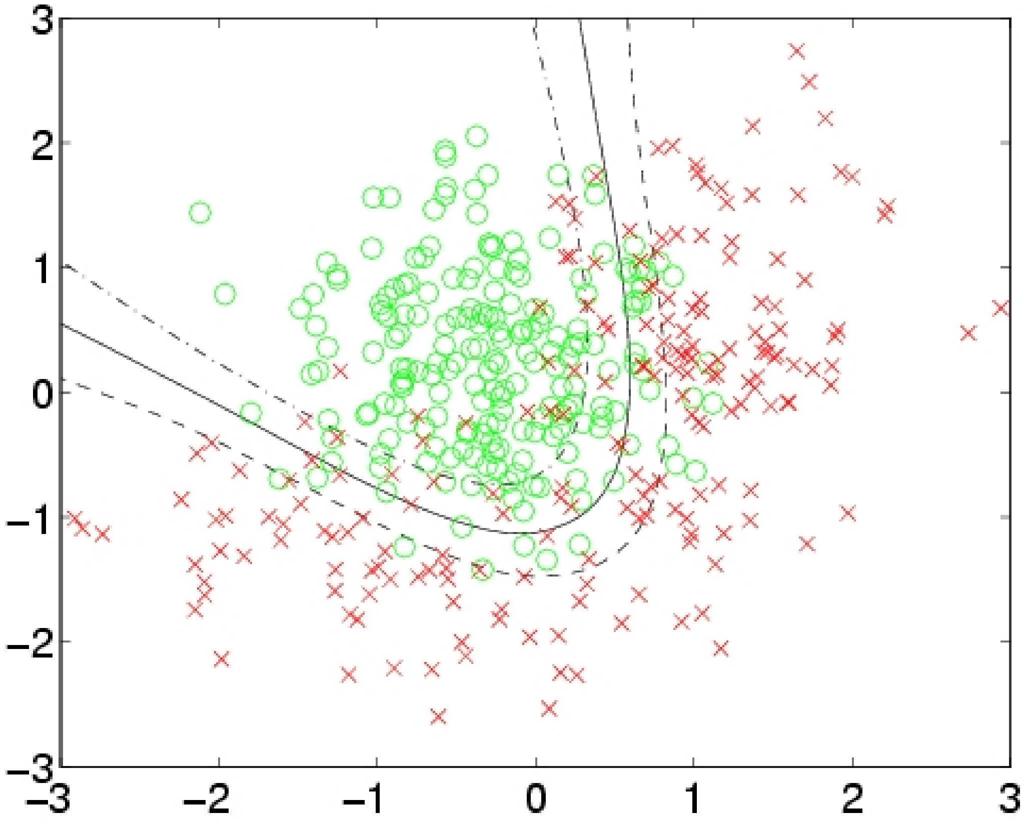




Bayesian example for regression



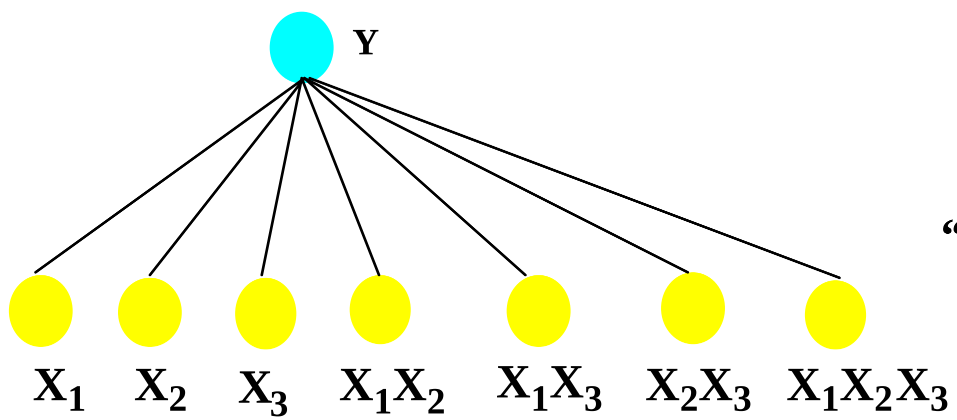
Bayesian example for classification



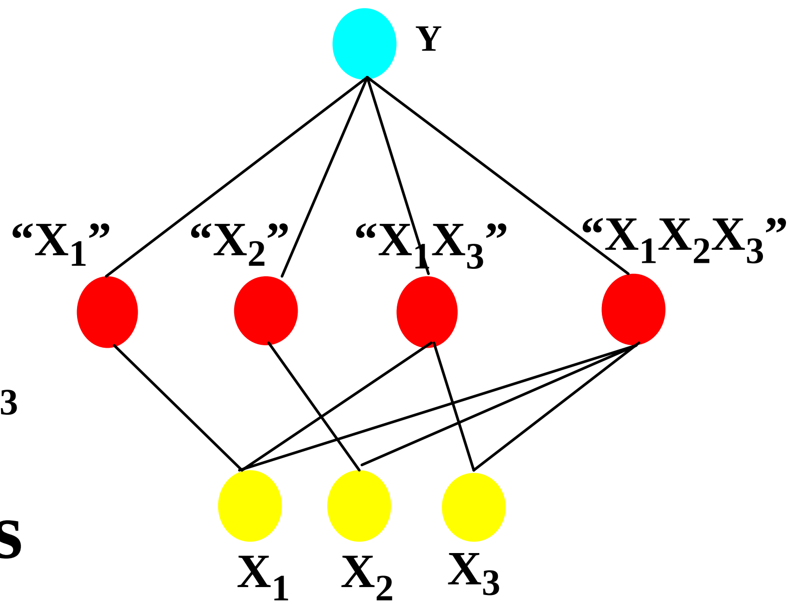
Model Features
(with strong personal biases)

	Modeling Effort	Examples Needed	Explanat.
Rule-based Exp. Syst.	high	low	high?
Classification Trees	low	high+	“high”
Neural Nets, SVM	low	high	low
Regression Models	high	moderate	moderate
Learned Bayesian Nets (beautiful when it works)	low	high+	high

Regression vs. Neural Networks



$(2^3\text{-}1)$ possible combinations



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

Summary

- ANNs inspired by functionality of brain
- Nonlinear data model
- Trained by minimizing error function
- Goal is to generalize well
- Avoid overtraining
- Distinguish ML and MAP solutions

Some References

Introductory and Historical Textbooks

- Rumelhart, D.E., and McClelland, J.L. (eds) Parallel Distributed Processing. MIT Press, Cambridge, 1986. (H)
- Hertz JA; Palmer RG; Krogh, AS. Introduction to the Theory of Neural Computation. Addison-Wesley, Redwood City, 1991.
- Pao, YH. Adaptive Pattern Recognition and Neural Networks. Addison-Wesley, Reading, 1989.
- Bishop CM. Neural Networks for Pattern Recognition. Clarendon Press, Oxford, 1995.