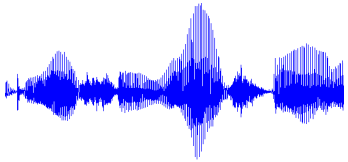
Neural Network

Outline

- Typical goal of machine Learning
- Single Perceptron
- Linear Regression
- Logistic Regression
- Multilayer Perceptron

*Many of slides adapted from Andrew Ng

Typical goal of machine learning

input			output
images/video		ML	Label: “Motorcycle” Suggest tags Image search …
audio		ML	Speech recognition Music classification Speaker identification …
text		ML	Web search Anti-spam Machine translation …

Typical goal of machine learning

Feature engineering:
most time consuming!

input

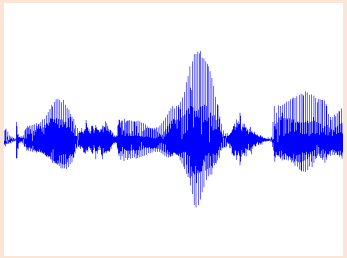
tput

images/video

ML

Label: “Motorcycle”
Suggest tags
Image search
…

audio



ML

Speech recognition
Music classification
Speaker identification
…

text

ML

Web search
Anti-spam
Machine translation
…

Our goal in object classification

“motorcycle”

Face Recognition

Face Expression

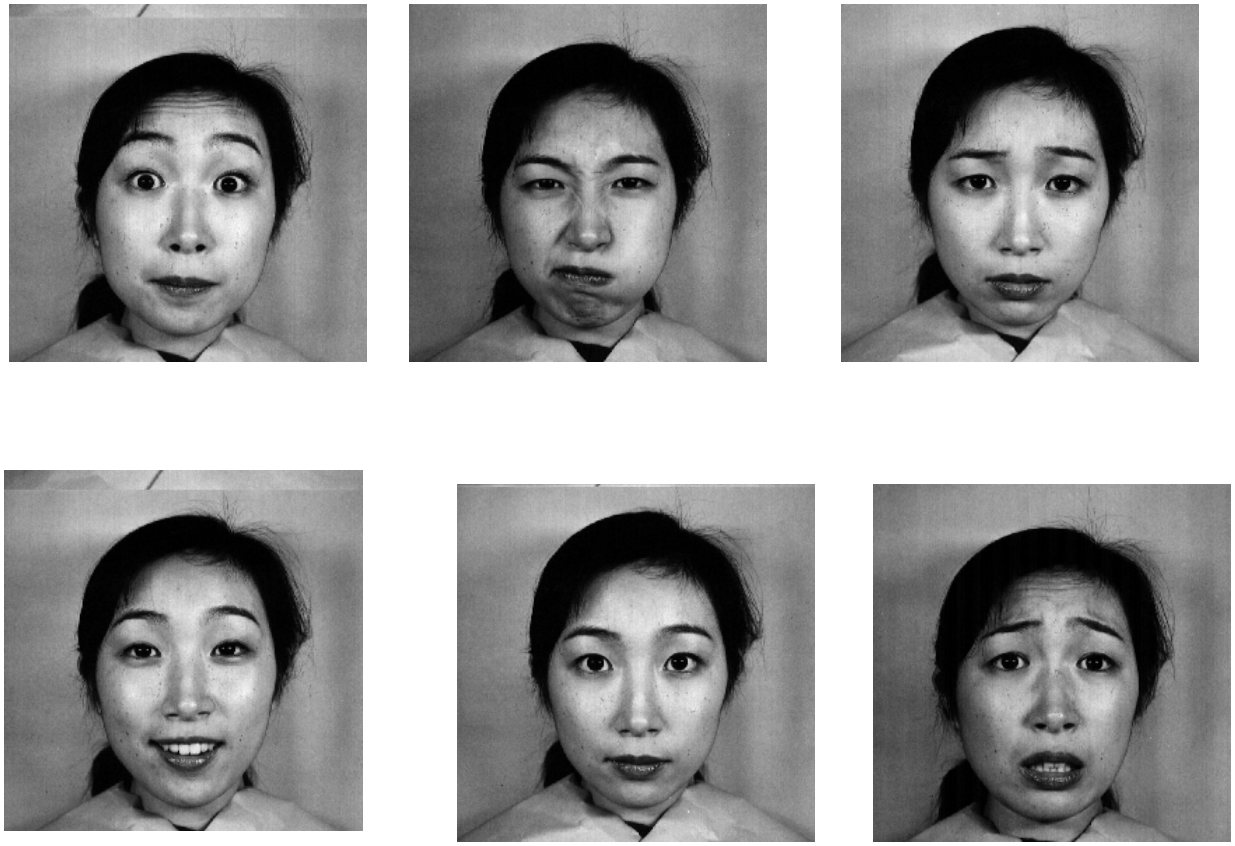
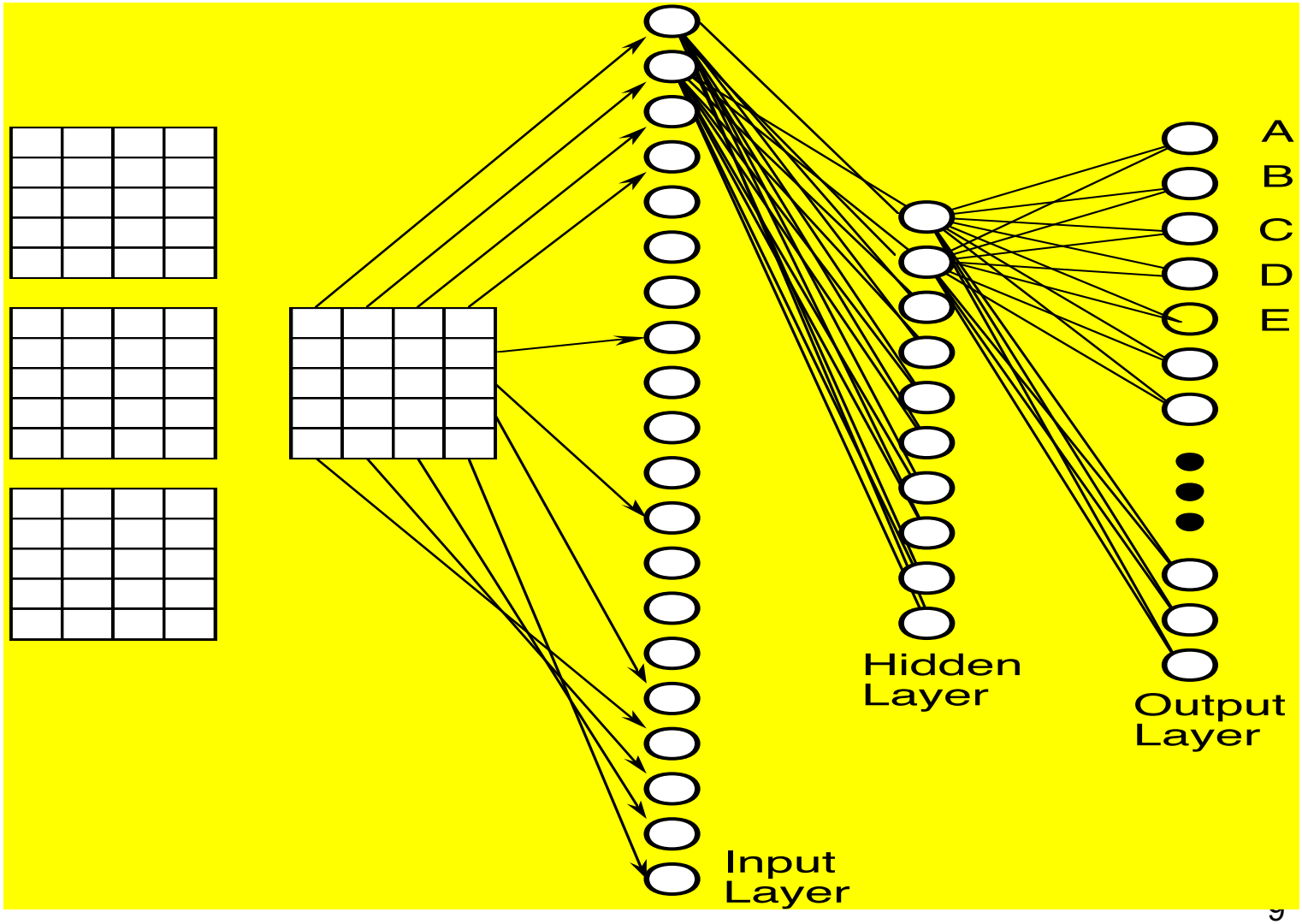


Figure 1 Different Facial Expressions of same person [23]

Fingerprint recognition

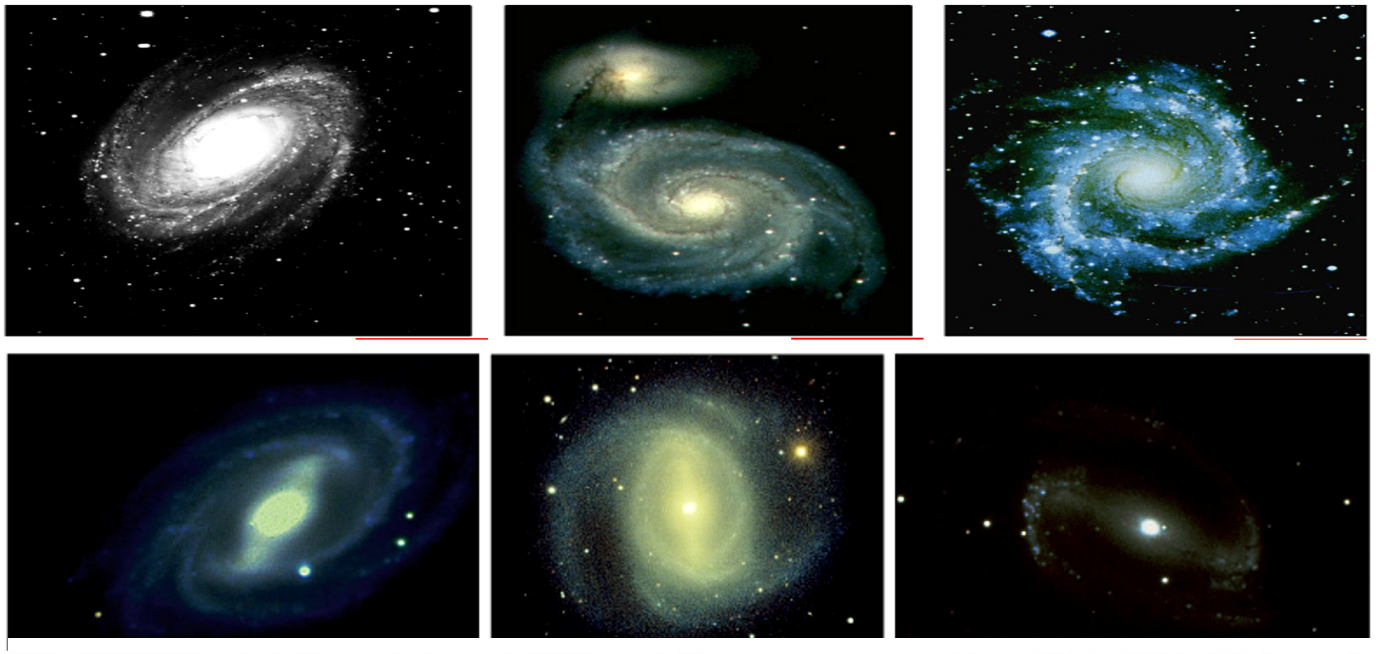
Optical Character Recognition



Signature recognition

- Each person's signature is different.
- There are structural similarities which are difficult to quantify.
- One company has manufactured a machine which recognizes signatures to within a high level of accuracy.
  - **Makes forgery even more difficult.**

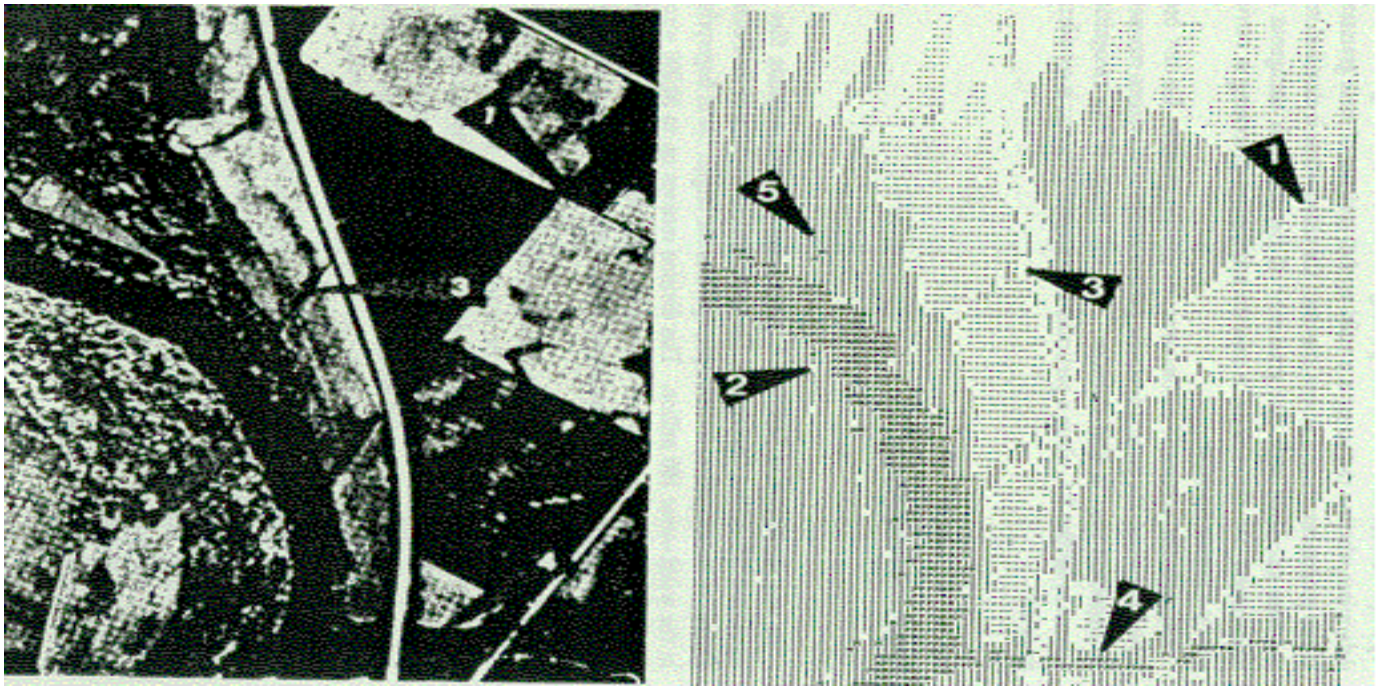
Galaxy Classification



(a) M49 Type E1 (b) M84 Type E3

Detection of Oil Slicks

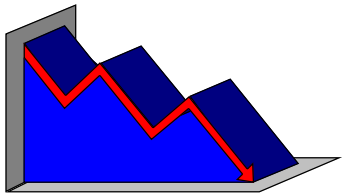
- Given radar satellite images of coastal waters
  Problem: **Detect Oil Slicks**



Loan Assessment

- Assess risk of lending to an individual.
- Difficult to decide on marginal cases.
- Neural networks have been trained to make decisions, based upon the opinions of expert underwriters.
- Neural network produced a 12% reduction in delinquencies compared with human experts.

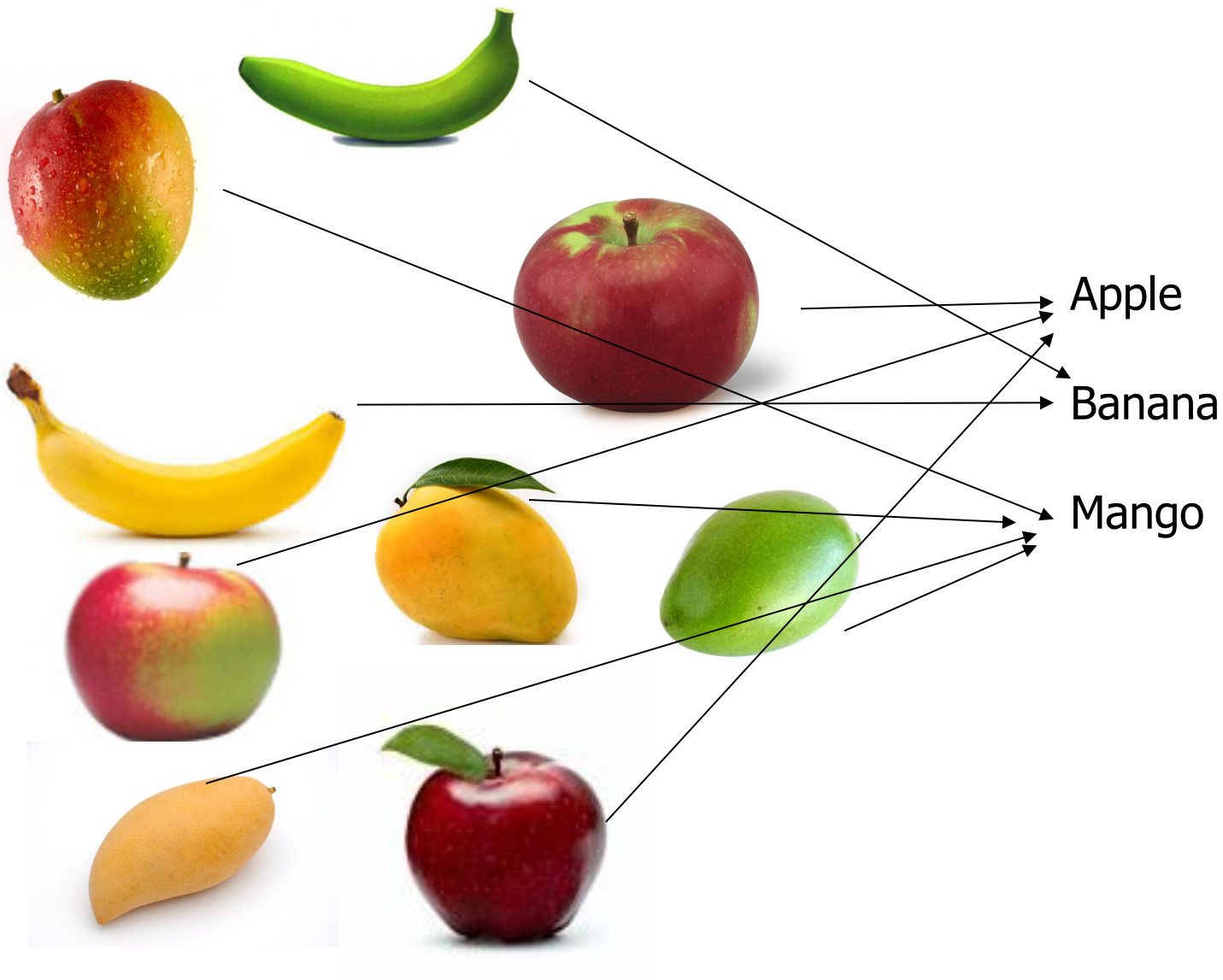
Stock market prediction



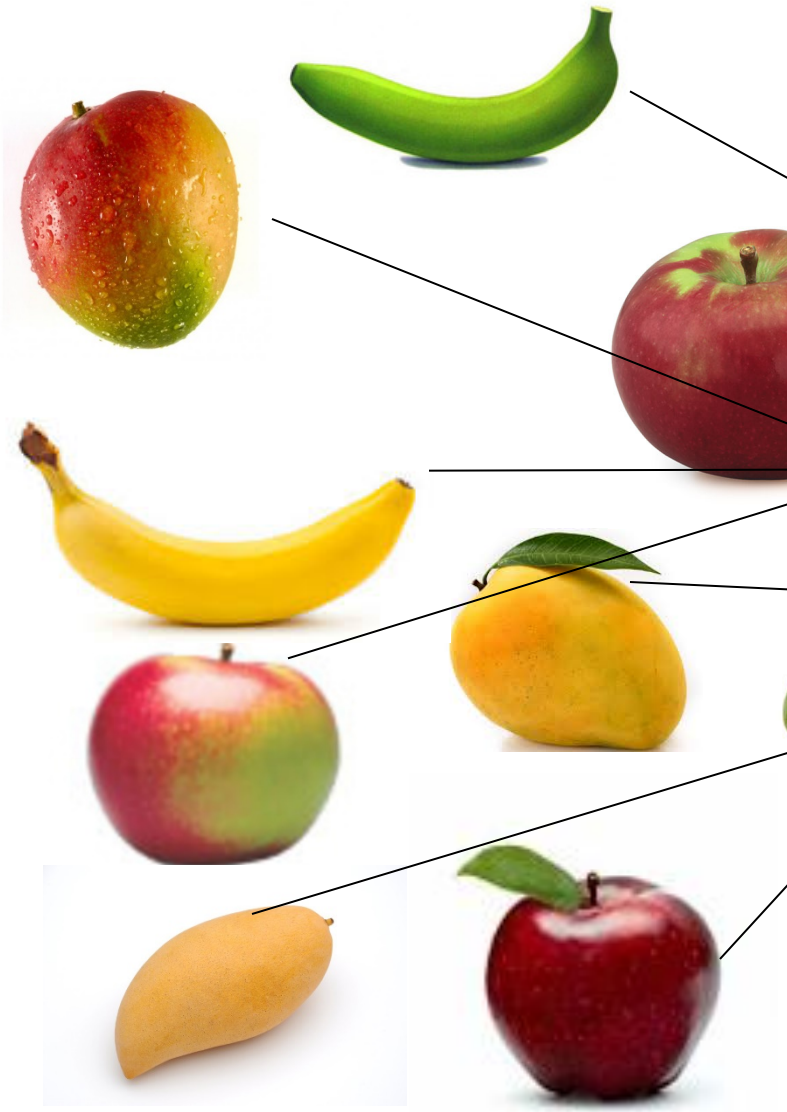
- "Technical trading" refers to trading based solely on known statistical parameters; e.g. previous price
- Neural networks have been used to attempt to predict changes in prices.
- Difficult to assess success since companies using these techniques are reluctant to disclose information.


2


A
B


Apple
Banana
Mango

Fruit



Name of Fruit

Apple

Banana

Mango

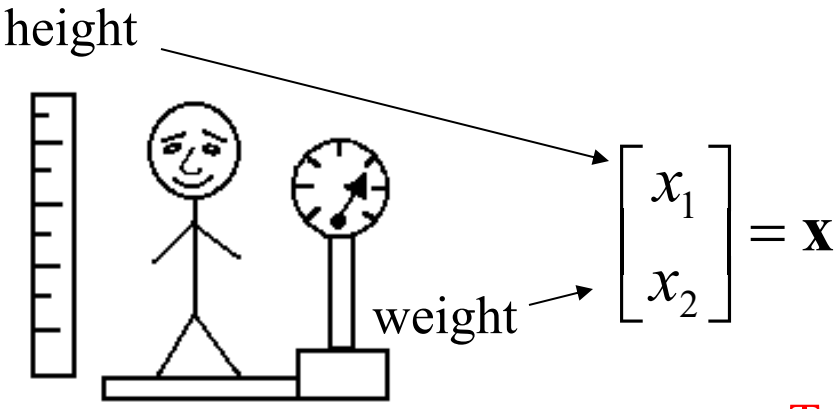
f(Fruit) ⟶ Name of Fruit

Classification

f(**Feature_vec**) → **Fruit_type**

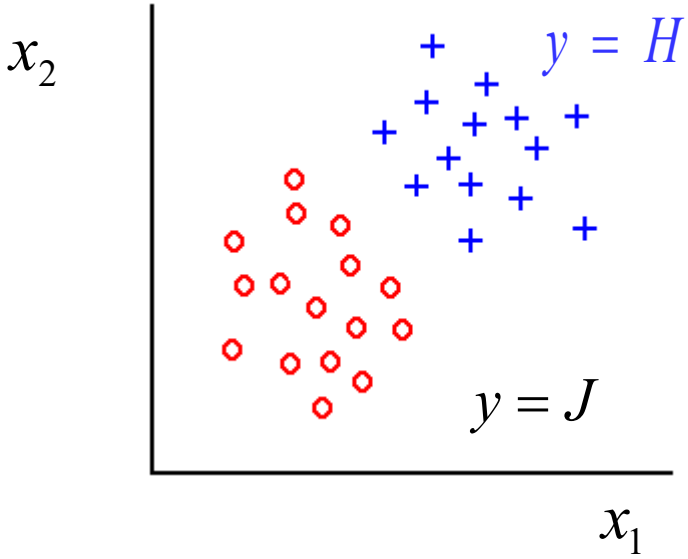
Feature Vector		**Fruit_type**
Color	**Shape**	
Red	Elliptical	Apple
Yellow	Elongated	Banana
Yellow	Elliptical	Mango
Green	Elliptical	Mango
Green	Elongated	Banana

Classification

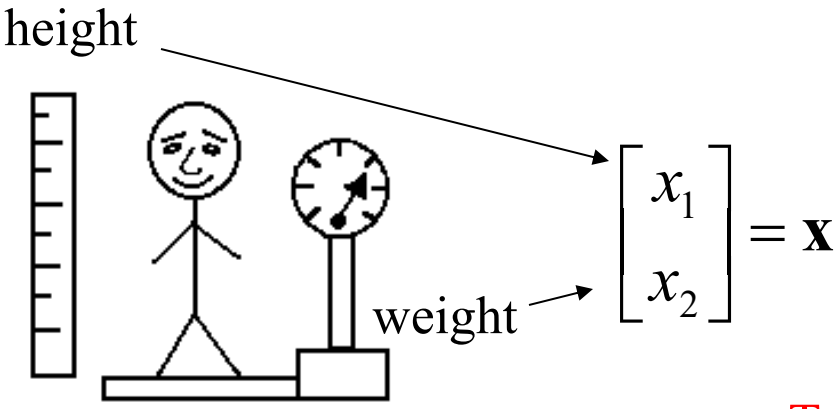


$$X = \Re^2$$

Training examples $\{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_l, y_l)\}$



Classification

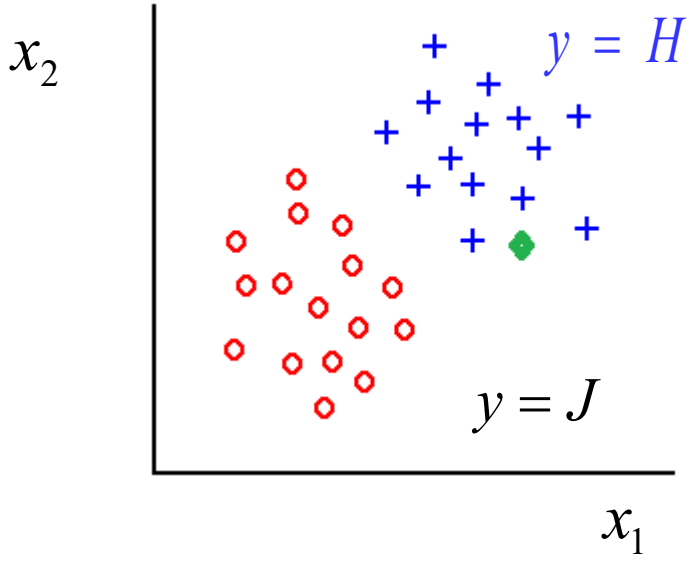


$X = \Re^2$

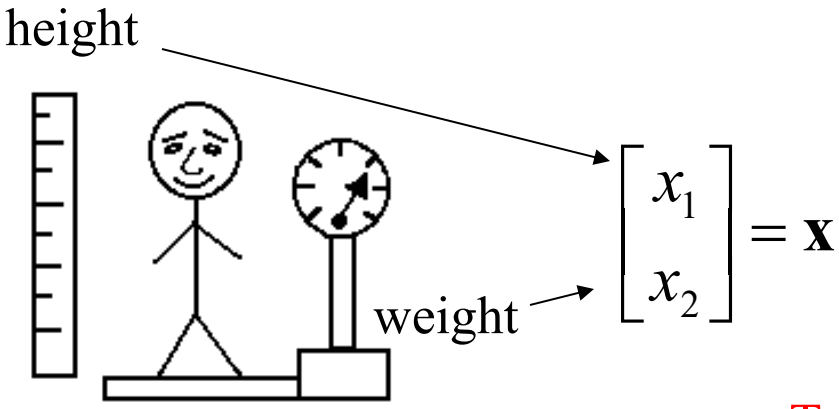
$\{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_l, y_l)\}$

Training examples

D_i = distance($\mathbf{x}$, $\mathbf{x}_i$), i=1,2,…,l



Classification



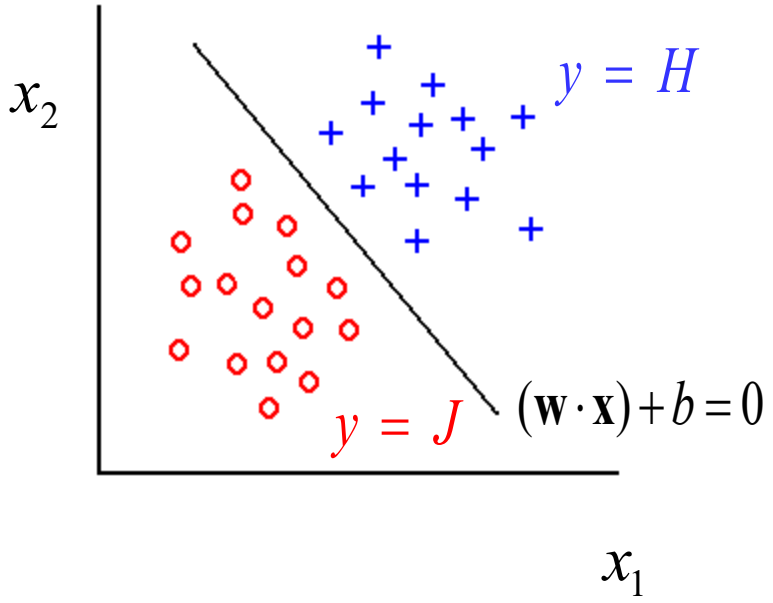
$$X = \Re^2$$

Training examples

$$\{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_l, y_l)\}$$

Linear classifier:

$$f(\mathbf{x}) = \begin{cases} H & if \quad (\mathbf{w} \cdot \mathbf{x}) + w_0 \geq 0 \\ J & if \quad (\mathbf{w} \cdot \mathbf{x}) + w_0 < 0 \end{cases}$$



Classification: Definition

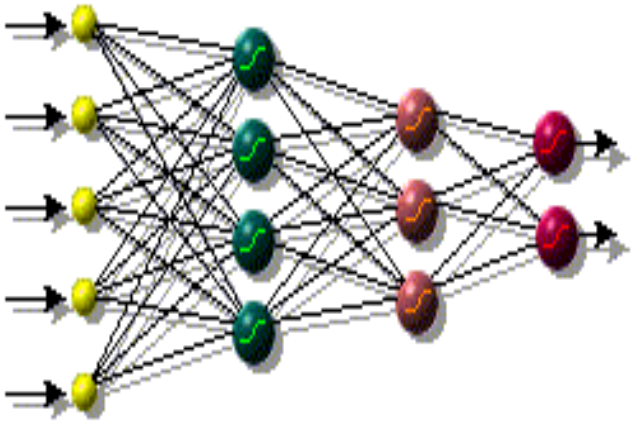
- Given a collection of records (*training set*)
  - Each record contains a set of *attributes*, one of the attributes is the *class* label.

- Find a *model* for class attribute as a function of the values of other attributes.

$q : X \rightarrow Y$

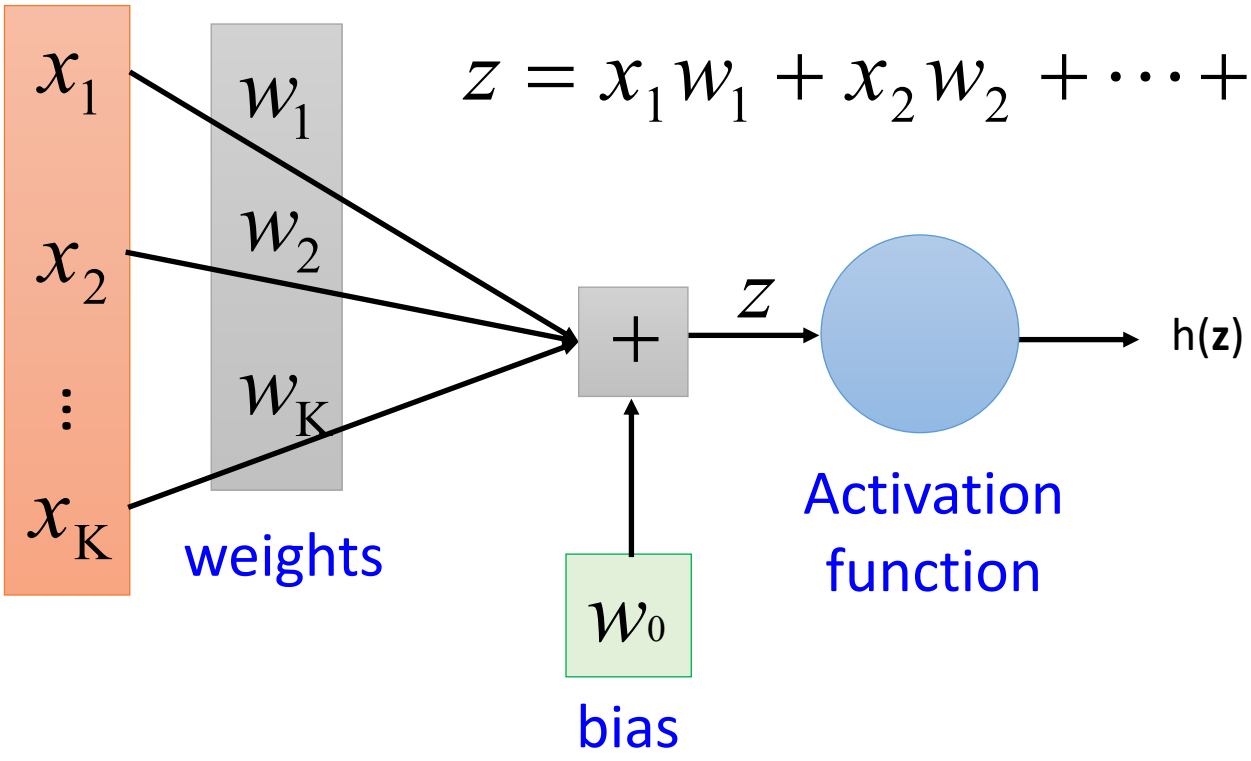
- Goal: <u>previously unseen</u> records should be assigned a class as accurately as possible.

Neural Network

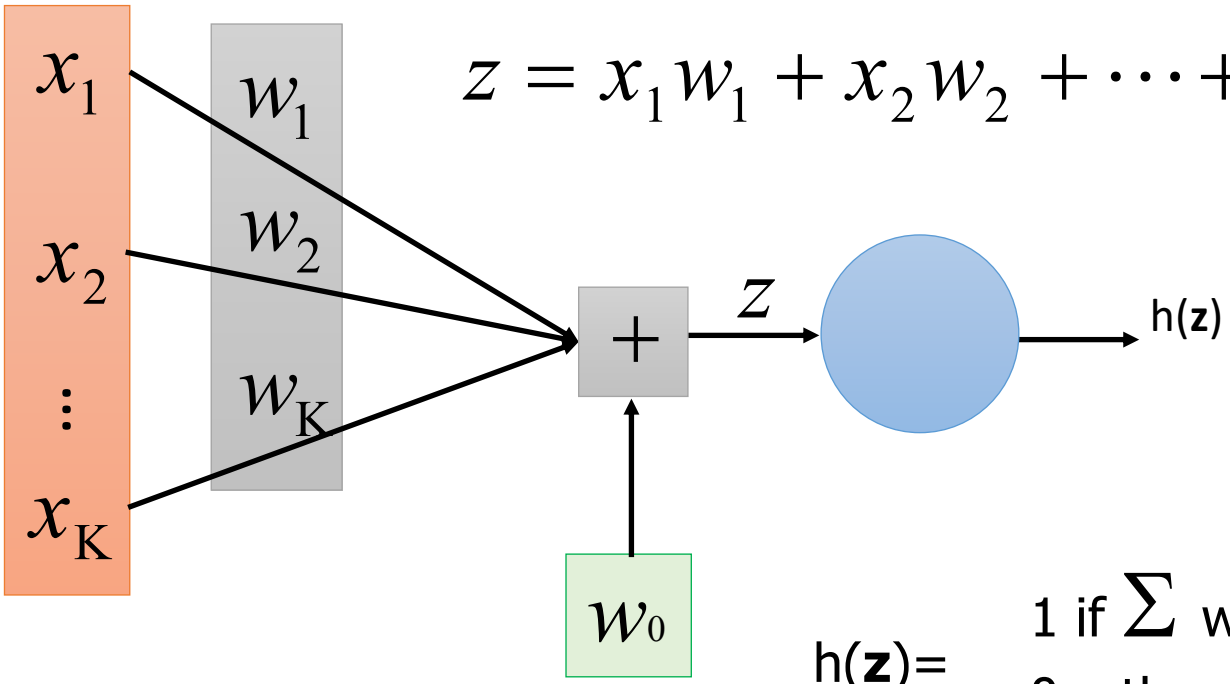


Elements of Neural Network

Neuron $f: R^K \rightarrow R$



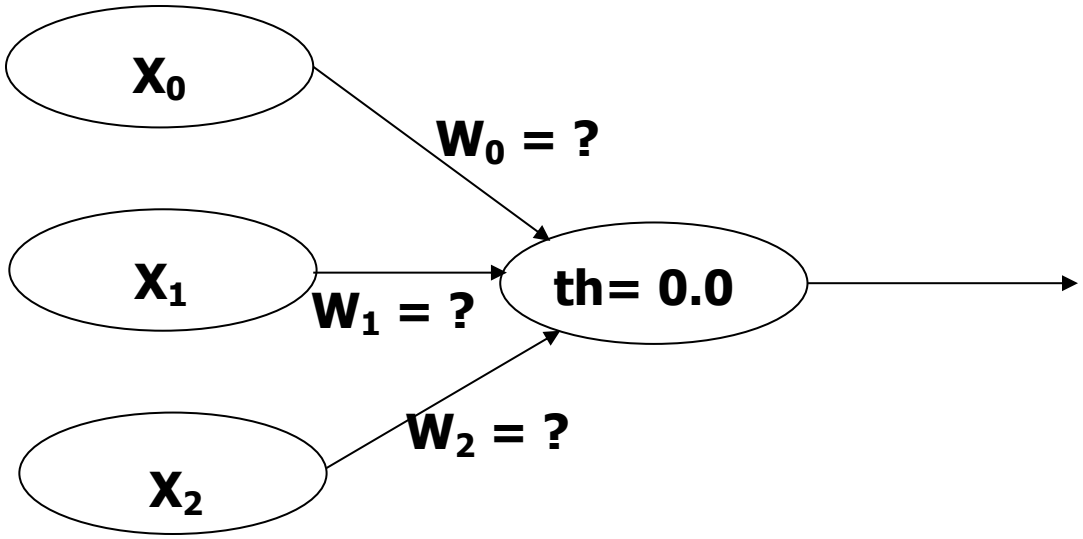
Single Perceptron



$$z = x_1 w_1 + x_2 w_2 + \cdots + x_K w_K + w_0$$

h(**z**)= 1 if $\sum$ w_i x_i > 0
0 otherwise

Training Perceptrons

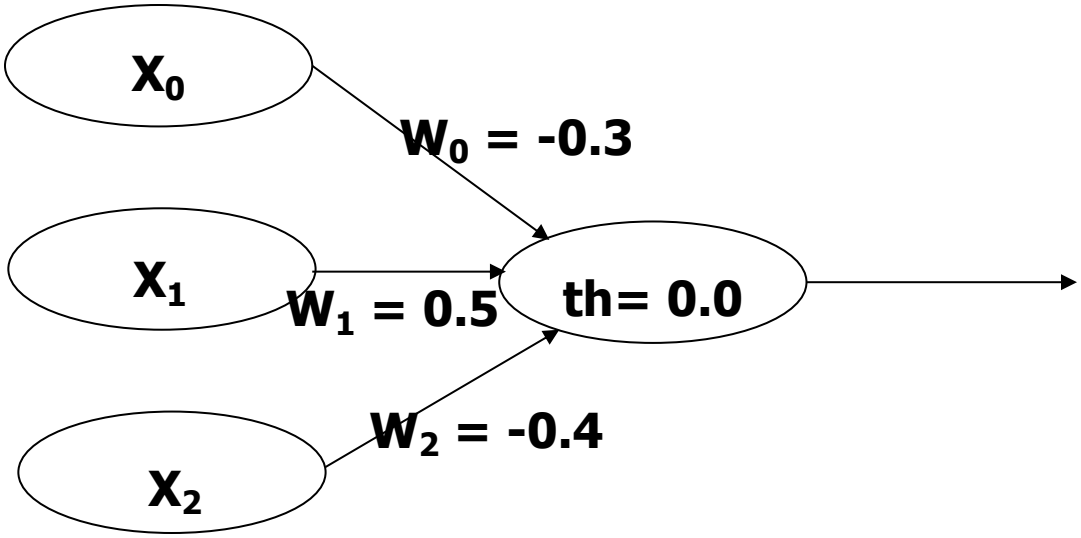


For AND

X_1	X_2	y
0	0	0
0	1	0
1	0	0
1	1	1

- What are the weight values?
- Initialize with random weight values

Training Perceptrons



For AND

X_1	X_2	y
0	0	0
0	1	0
1	0	0
1	1	1

X_0	X_1	X_2	Summation	Output
1	0	0	(-1*0.3) + (0*0.5) + (0*-0.4) = -0.3	0
1	0	1	(-1*0.3) + (0*0.5) + (1*-0.4) = -0.7	0
1	1	0	(-1*0.3) + (1*0.5) + (0*-0.4) = 0.2	1
1	1	1	(-1*0.3) + (1*0.5) + (1*-0.4) = -0.2	0

Gradient Descent Learning Rule

- Train the w_i's such that they minimize the squared error
  - $E[w_1,\ldots,w_n] = \frac{1}{2} \sum_{d \in D} (y_d - h_d)^2$

  where D is the set of training examples

Gradient Descent

Gradient:

$\nabla E[w]=[\partial E/\partial w_0, \ldots \partial E/\partial w_n]$

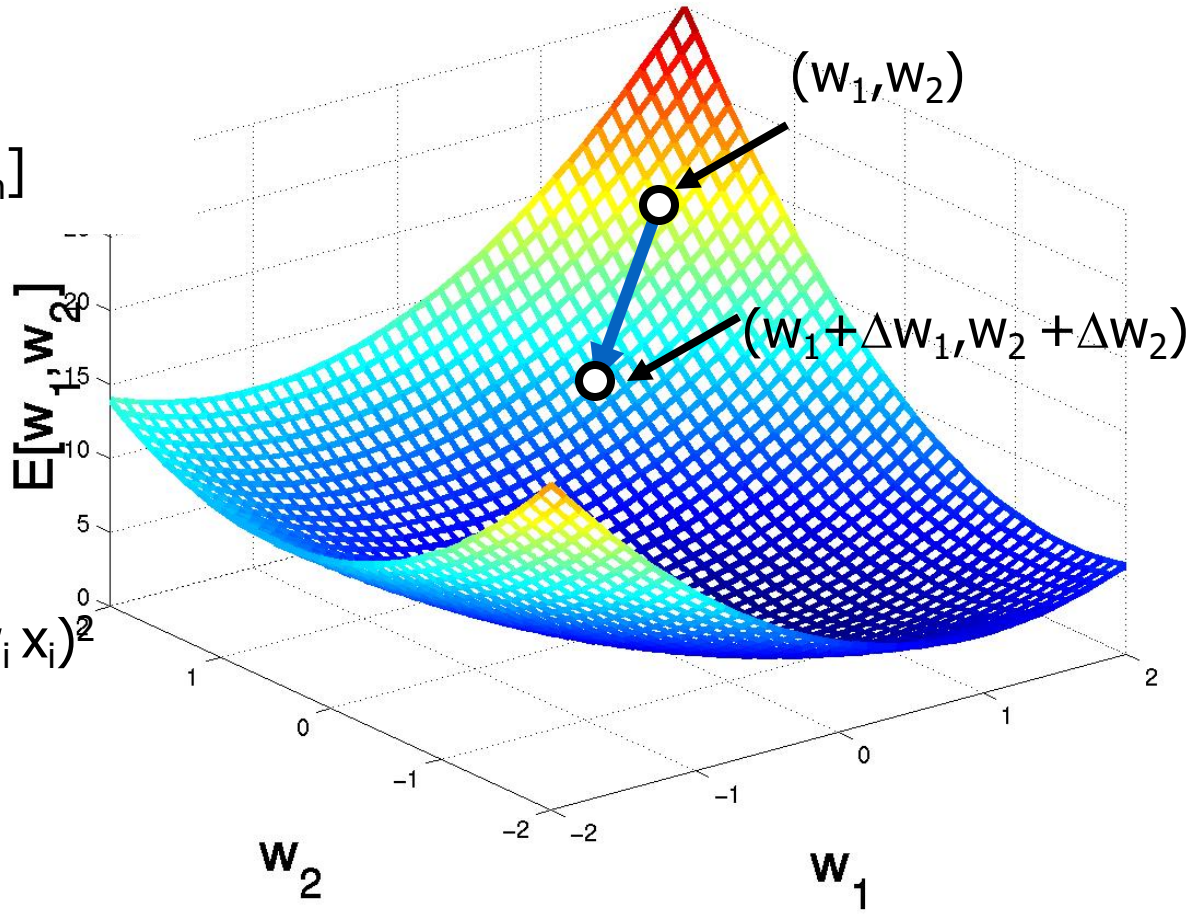
$\Delta w = - \nabla E[w]$

$\Delta w_i = - \partial E/\partial w_i$

$= - \partial/\partial w_i \; 1/2 \sum_d (y_d - h_d)^2$

$= - \partial/\partial w_i \; 1/2 \sum_d (y_d - \sum_i w_i x_i)^2$

$= \sum_d (y_d - h_d)(x_i)$



Weight Updation

- W_0= -0.3 + [(0-0)1+(0-0)1+(0-1)1+(1-0)1]= -0.3
- W_1= 0.5 + [(0-0)0+(0-0)0+(0-1)1+(1-0)1]= 0.5
- W_2= -0.4 + [(0-0)0+(0-0)1+(0-1)0+(1-0)1]= 0.6

X_0	X_1	X_2	Summation	Output
1	0	0	(-1*0.3) + (0*0.5) + (0*0.6) = -0.3	0
1	0	1	(-1*0.3) + (0*0.5) + (1*0.6) = 0.3	1
1	1	0	(-1*0.3) + (1*0.5) + (0*0.6) = 0.2	1
1	1	1	(-1*0.3) + (1*0.5) + (1*0.6) = 0.8	1

Weight Updation

- W_0= -0.3 + [(0-0)1+(0-1)1+(0-1)1+(1-1)1]= -2.3
- W_1= 0.5 + [(0-0)0+(0-1)0+(0-1)1+(1-1)1]= -0.5
- W_2= 0.6 + [(0-0)0+(0-1)1+(0-1)0+(1-1)1]= -0.4

X_0	X_1	X_2	Summation	Output
1	0	0	(-1*2.3) + (-0*0.5) + (-0*0.4) = -2.3	0
1	0	1	(-1*2.3) + (-0*0.5) + (-1*0.4) = -2.7	0
1	1	0	(-1*2.3) + (-1*0.5) + (-0*0.4) = -2.8	0
1	1	1	(-1*2.3) + (-1*0.5) + (-1*0.4) = -3.2	0

Weight Updation

- W_0= -2.3 + [(0-0)1+(0-0)1+(0-0)1+(1-0)1]= -1.3
- W_1= -0.5 + [(0-0)0+(0-0)0+(0-0)1+(1-0)1]= 0.5
- W_2= -0.4 + [(0-0)0+(0-0)1+(0-0)0+(1-0)1]= 0.6

X_0	X_1	X_2	Summation	Output
1	0	0	(-1*1.3) + (0*0.5) + (0*0.6) = -1.3	0
1	0	1	(-1*1.3) + (0*0.5) + (1*0.6) = -0.7	0
1	1	0	(-1*1.3) + (1*0.5) + (0*0.6) = -0.8	0
1	1	1	(-1*1.3) + (1*0.5) + (1*0.6) = -0.2	0

Weight Updation

- W_0= -1.3 + [(0-0)1+(0-0)1+(0-0)1+(1-0)1]= -0.3
- W_1= 0.5 + [(0-0)0+(0-0)0+(0-0)1+(1-0)1]= 1.5
- W_2= 0.6 + [(0-0)0+(0-0)1+(0-0)0+(1-0)1]= 1.6

X_0	X_1	X_2	Summation	Output
1	0	0	(-1*0.3) + (0*1.5) + (0*1.6) = -0.3	0
1	0	1	(-1*0.3) + (0*1.5) + (1*1.6) = -1.3	0
1	1	0	(-1*0.3) + (1*1.5) + (0*1.6) = 1.2	1
1	1	1	(-1*0.3) + (1*1.5) + (1*1.6) = 2.8	1

Weight Updation

- W_0= -0.3 + [(0-0)1+(0-0)1+(0-1)1+(1-1)1]= -1.3
- W_1= 1.5 + [(0-0)0+(0-0)0+(0-1)1+(1-1)1]= 0.5
- W_2= 1.6 + [(0-0)0+(0-0)1+(0-1)0+(1-1)1]= 1.6

X_0	X_1	X_2	Summation	Output
1	0	0	(-1*1.3) + (0*0.5) + (0*1.6) = -1.3	0
1	0	1	(-1*1.3) + (0*0.5) + (1*1.6) = 0.3	1
1	1	0	(-1*1.3) + (1*0.5) + (0*1.6) = -0.8	0
1	1	1	(-1*1.3) + (1*0.5) + (1*1.6) = 0.8	1

Weight Updation

- W_0= -1.3 + [(0-0)1+(0-1)1+(0-0)1+(1-1)1]= -2.3
- W_1= 0.5 + [(0-0)0+(0-1)0+(0-0)1+(1-1)1]= 0.5
- W_2= 1.6 + [(0-0)0+(0-1)1+(0-0)0+(1-1)1]= 0.6

X_0	X_1	X_2	Summation	Output
1	0	0	(-1*2.3) + (0*0.5) + (0*0.6) = -2.3	0
1	0	1	(-1*2.3) + (0*0.5) + (1*0.6) = -1.7	0
1	1	0	(-1*2.3) + (1*0.5) + (0*0.6) = -1.8	0
1	1	1	(-1*2.3) + (1*0.5) + (1*0.6) = -1.2	0

Weight Updation

- W_0= -2.3 + [(0-0)1+(0-0)1+(0-0)1+(1-0)1]= -1.3
- W_1= 0.5 + [(0-0)0+(0-0)0+(0-0)1+(1-0)1]= 1.5
- W_2= 0.6 + [(0-0)0+(0-0)1+(0-0)0+(1-0)1]= 1.6

X_0	X_1	X_2	Summation	Output
1	0	0	(-1*1.3) + (0*1.5) + (0*1.6) = -1.3	0
1	0	1	(-1*1.3) + (0*1.5) + (1*1.6) = 0.3	1
1	1	0	(-1*1.3) + (1*1.5) + (0*1.6) = 0.2	1
1	1	1	(-1*1.3) + (1*1.5) + (1*1.6) = 1.8	1

Weight Updation

- W_0= -1.3 + [(0-0)1+(0-1)1+(0-1)1+(1-1)1]= -3.3
- W_1= 1.5 + [(0-0)0+(0-1)0+(0-1)1+(1-1)1]= 0.5
- W_2= 1.6 + [(0-0)0+(0-1)1+(0-1)0+(1-1)1]= 0.6

X_0	X_1	X_2	Summation	Output
1	0	0	(-1*3.3) + (0*0.5) + (0*0.6) = -3.3	0
1	0	1	(-1*3.3) + (0*0.5) + (1*0.6) = -2.7	0
1	1	0	(-1*3.3) + (1*0.5) + (0*0.6) = -2.8	0
1	1	1	(-1*3.3) + (1*0.5) + (1*0.6) = -2.2	0

Weight Updation

- W_0= -3.3 + [(0-0)1+(0-0)1+(0-0)1+(1-0)1]= -2.3
- W_1= 0.5 + [(0-0)0+(0-0)0+(0-0)1+(1-0)1]= 1.5
- W_2= 0.6 + [(0-0)0+(0-0)1+(0-0)0+(1-0)1]= 1.6

X_0	X_1	X_2	Summation	Output
1	0	0	(-1*2.3) + (0*1.5) + (0*1.6) = -2.3	0
1	0	1	(-1*2.3) + (0*1.5) + (1*1.6) = -0.7	0
1	1	0	(-1*2.3) + (1*1.5) + (0*1.6) = -0.8	0
1	1	1	(-1*2.3) + (1*1.5) + (1*1.6) = 0.8	1

Single Perceptron

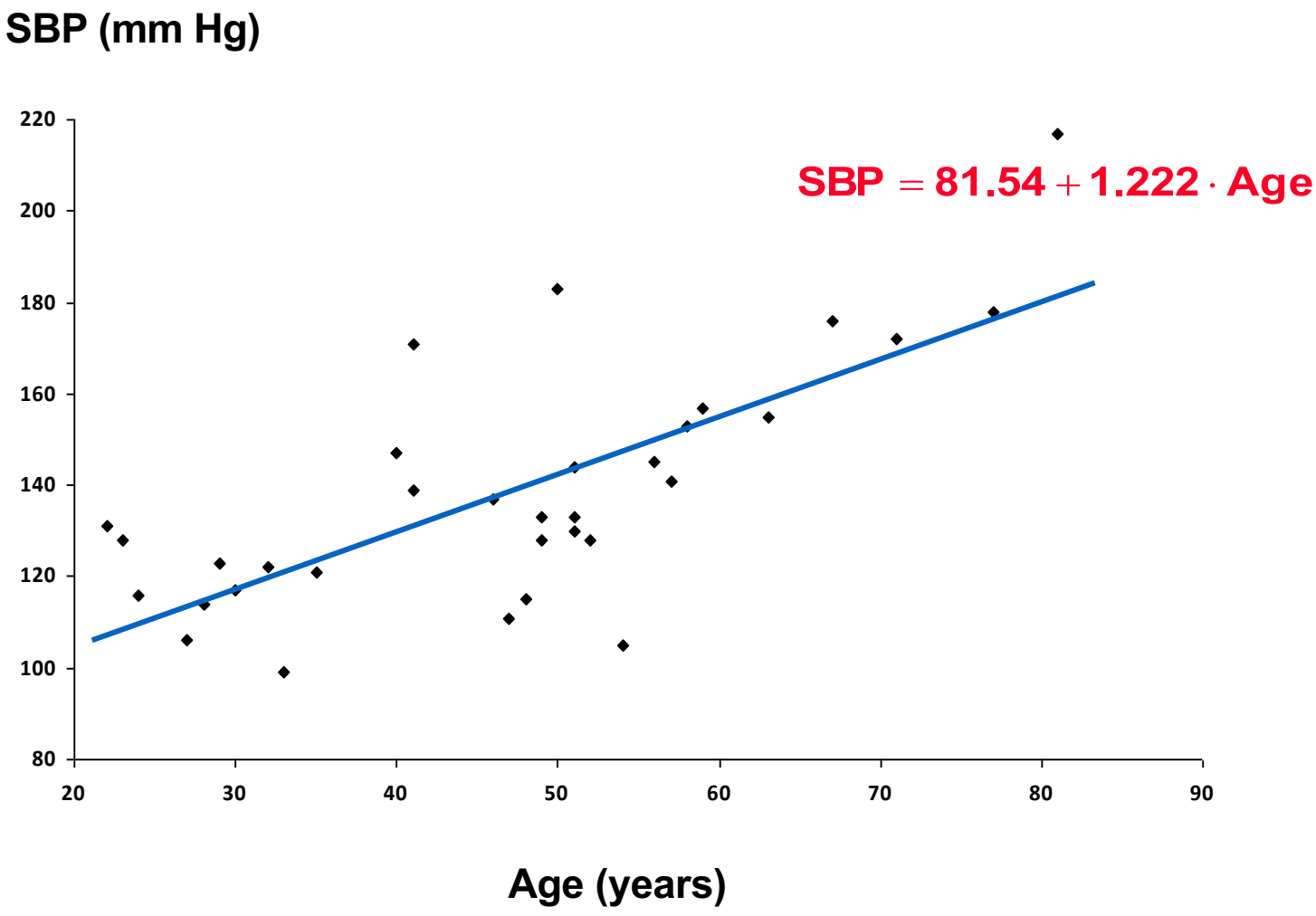
Each training example is a pair of the form $<(x_1,\ldots x_n),y>$ where $(x_1,\ldots,x_n)$ is the vector of input values, and t is the target output value

- Initialize each w_i to some small random value
- Until the termination condition is met, Do
  - Initialize each Δw_i to zero
  - For each $<(x_1,\ldots x_n),t>$ in *training_examples* Do
    - Input the instance $(x_1,\ldots,x_n)$ to the linear unit and compute the output o
    - For each linear unit weight w_i Do
      - $\Delta w_i = \Delta w_i + \sum_d (y_d - h_d)\, x_i$

Simple linear regression

Table 1 Age and systolic blood pressure (SBP) among 33 adult women

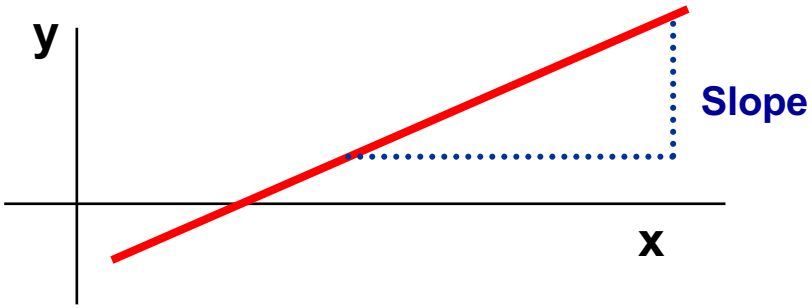
Age	SBP	Age	SBP	Age	SBP
22	131	41	139	52	128
23	128	41	171	54	105
24	116	46	137	56	145
27	106	47	111	57	141
28	114	48	115	58	153
29	123	49	133	59	157
30	117	49	128	63	155
32	122	50	183	67	176
33	99	51	130	71	172
35	121	51	133	77	178
40	147	51	144	81	217



adapted from Colton T. Statistics in Medicine. Boston: Little Brown, 1974

Simple linear regression

- Relation between 2 continuous variables (SBP and age)



$$y = \alpha + \beta_1 x_1$$

- Regression coefficient β_1
  - Measures association between y and x
  - Amount by which y changes on average when x changes by one unit
  - Least squares method

Multiple linear regression

- Relation between a continuous variable and a set of i continuous variables

$$y = \alpha + \beta_1 x_1 + \beta_2 x_2 + \ldots + \beta_i x_i$$

- Partial regression coefficients β_i
  - Amount by which y changes on average when x_i changes by one unit and all the other x_is remain constant
  - Measures association between x_i and y adjusted for all other x_i

- Example
  - SBP *versus* age, weight, height, etc

Multiple linear regression

$$y = \alpha + \beta_1 x_1 + \beta_2 x_2 + \ldots + \beta_i x_i$$

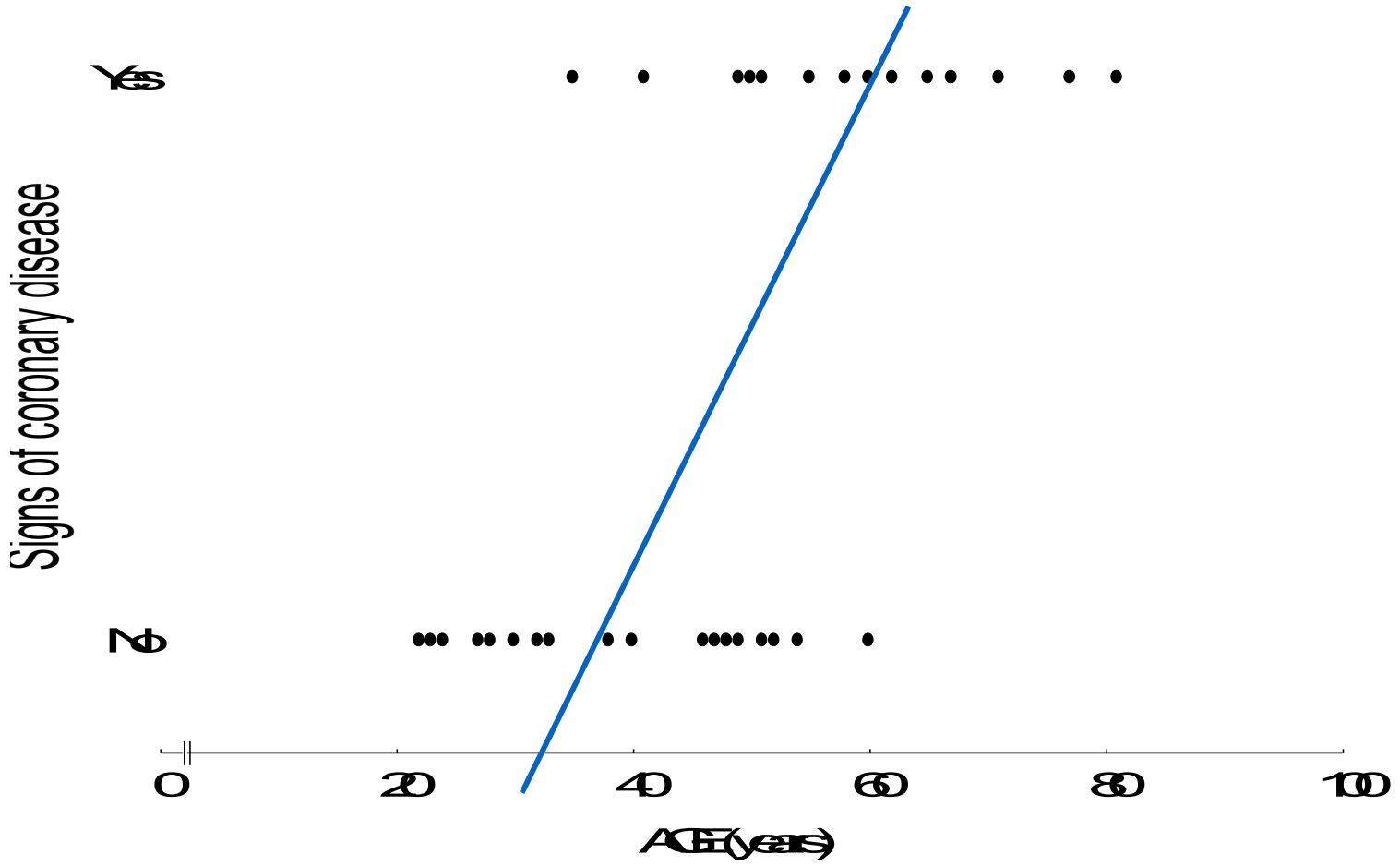
Predicted	Predictor variables
Response variable	Explanatory variables
Outcome variable	Covariables
Dependent	Independent variables

Logistic regression

Table 2 Age and signs of coronary heart disease (CD)

Age	CD	Age	CD	Age	CD
22	0	40	0	54	0
23	0	41	1	55	1
24	0	46	0	58	1
27	0	47	0	60	1
28	0	48	0	60	0
30	0	49	1	62	1
30	0	49	0	65	1
32	0	50	1	67	1
33	0	51	0	71	1
35	1	51	1	77	1
38	0	52	0	81	1

Dot-plot: Data from Table 2

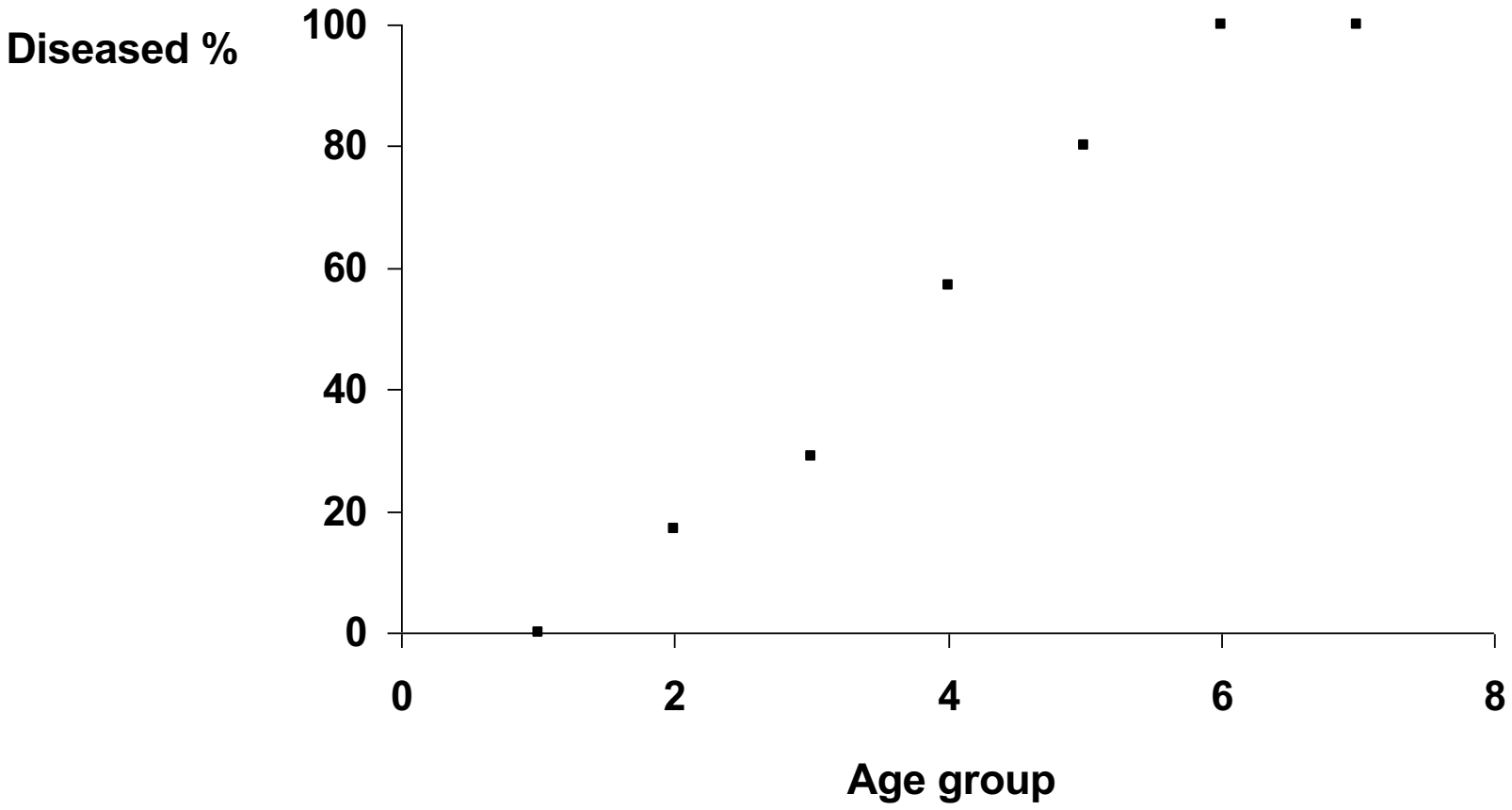


Logistic regression

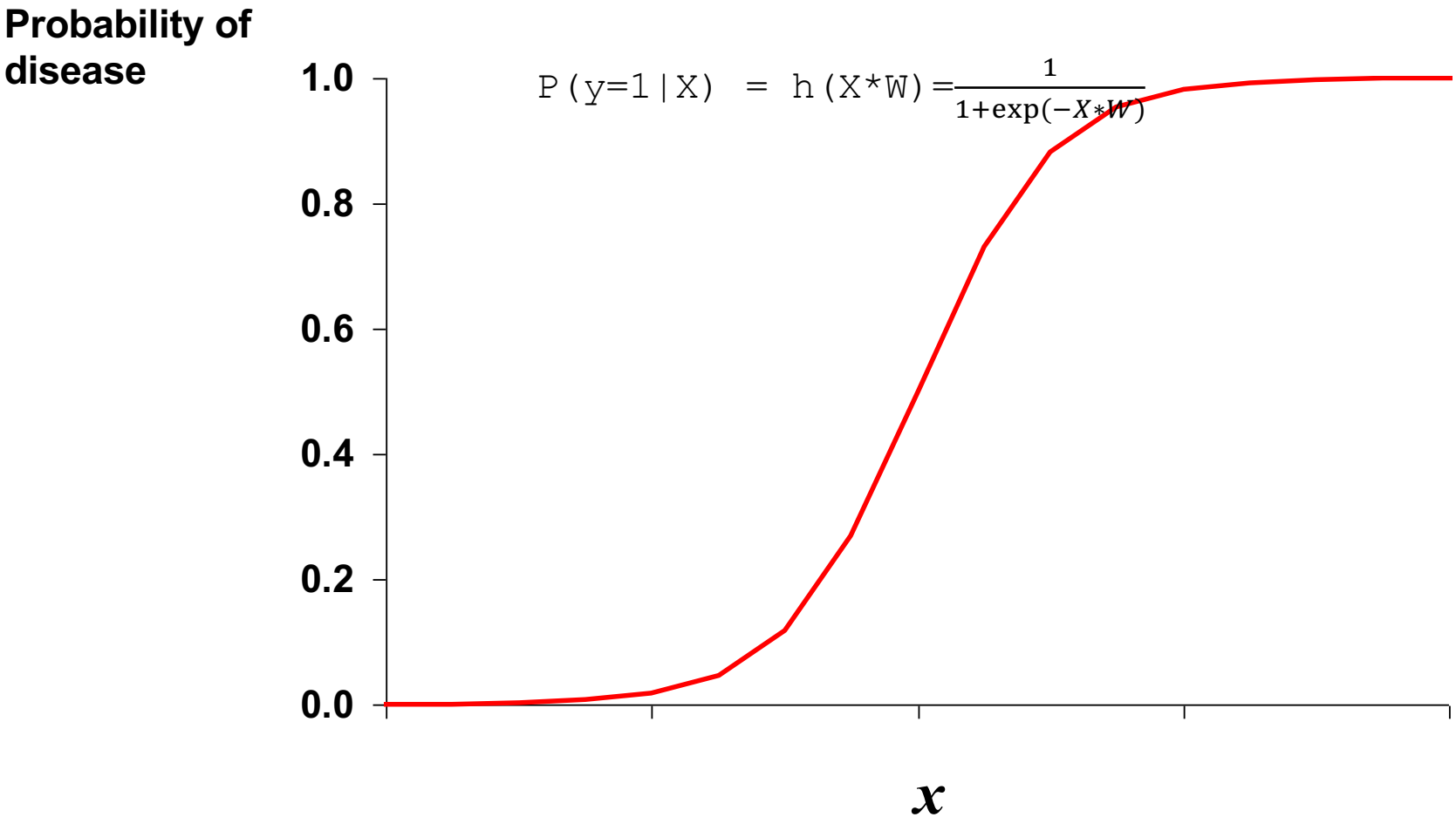
Table 3 Prevalence (%) of signs of CD according to age group

		Diseased	
Age group	# in group	#	%
20 - 29	5	0	0
30 - 39	6	1	17
40 - 49	7	2	29
50 - 59	7	4	57
60 - 69	5	4	80
70 - 79	2	2	100
80 - 89	1	1	100

Dot-plot: Data from Table 3



Logistic function



Logistic Regression

$$P(y=1|X) = h(X*W) = \frac{1}{1+\exp(-X*W)};$$

$$P(y=0|X) = 1-P(y=1|X) = \frac{\exp(-X*W)}{1+\exp(-X*W)};$$

$$P(y|X) = (h(X*W))^{y}(1-h(X*W))^{(1-y)}$$

$$L(\theta) = P(y \mid X; W)$$

$$= \prod_{i=1}^{n} P(y_i \mid \boldsymbol{X_i}; \mathbf{W})$$

$$= \prod_{i=1}^{n} (h(\boldsymbol{X_i}*W))^{y_i}(1-h(\boldsymbol{X_i}*W))^{(1-yi)}$$

$$l(\theta) = \log(L(\theta))$$

$$= \sum_{i=1}^{n} y_i \log(h(\boldsymbol{X_i}*W)) + (1-y_i)\log(1-h(\boldsymbol{X_i}*W))$$

Logistic Regression

$$\frac{\partial \ell(\theta)}{\partial W_j} = \left(y \frac{1}{h(X.W)} - (1-y) \frac{1}{(1-h(X.W))}\right) \frac{\partial h(X.W)}{\partial W_j}$$

$$= \left(y \frac{1}{h(X.W)} - (1-y) \frac{1}{(1-h(X.W))}\right) h(X.W)\,(1-h(X.W)) \frac{\partial (X.W)}{\partial W_j}$$

$$= (y\,(1-h(X.W)) - (1-y)\,h(X.W))\,\mathbf{x}_j$$

$$= (y - h(X.W))\,X_j$$

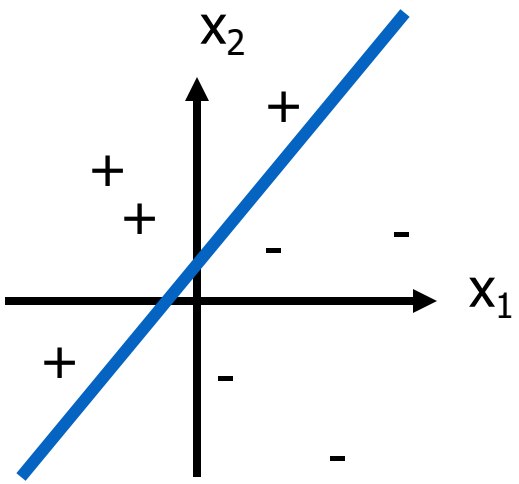
$$W_j = W_j + \text{alpha}*(y - h(X.W))\,X_j$$

Maximum likelihood

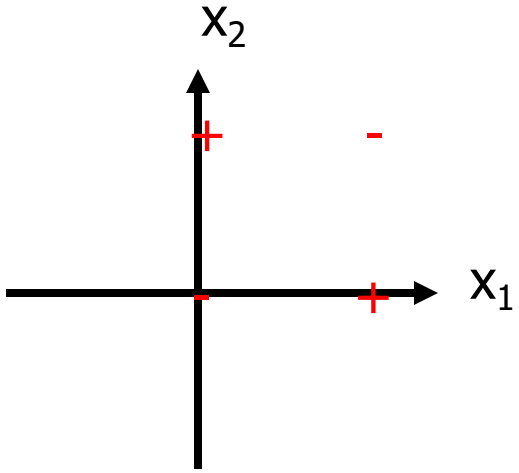
- Iterative computing
  - Choice of an arbitrary value for the coefficients (usually 0)
  - Computing of log-likelihood
  - Variation of coefficients' values
  - Reiteration until maximisation (plateau)

- Results
  - Maximum Likelihood Estimates (MLE) for α and β
  - Estimates of P(y) for a given value of x

Decision Surface of a Perceptron



Linearly separable

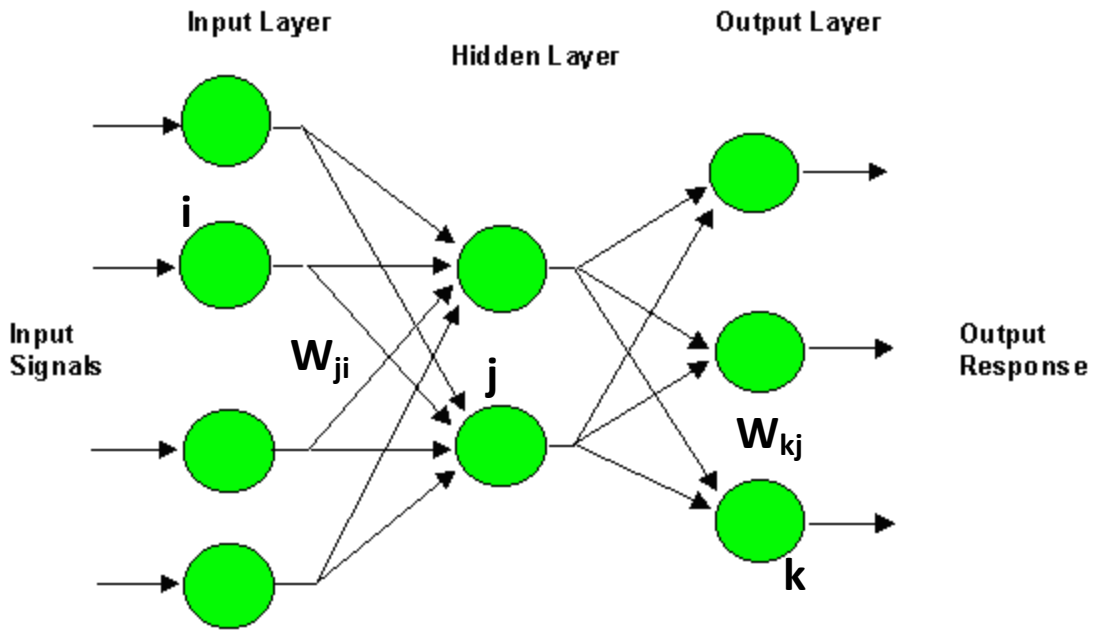


Non-Linearly separable

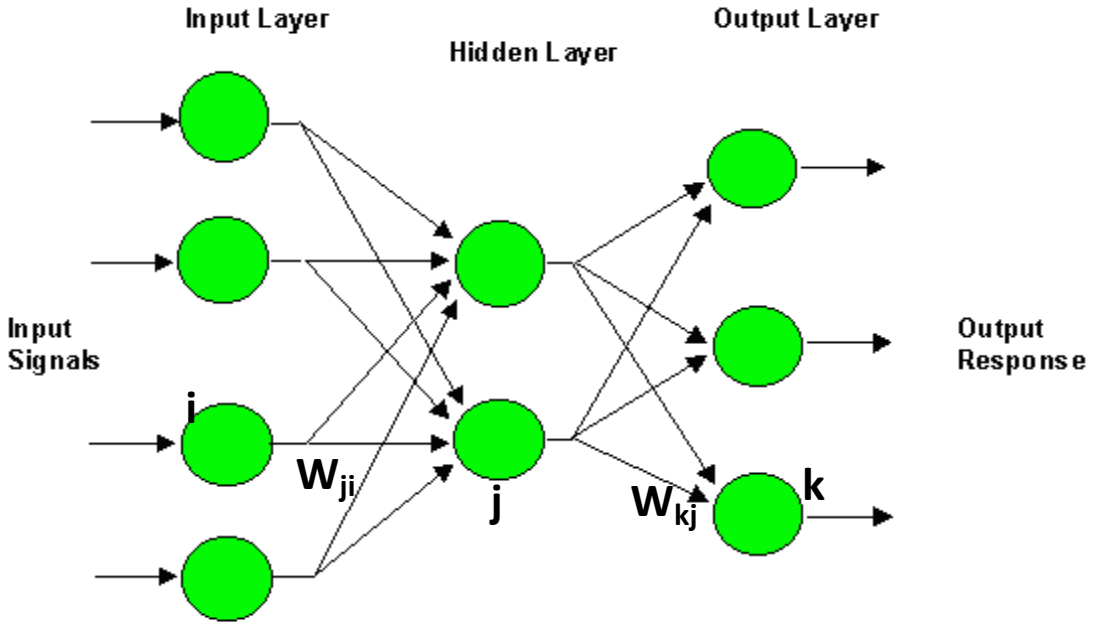
- But functions that are not linearly separable (e.g. XOR)
  XOR can solved as:

$XOR(x_1, x_2) = AND(\ OR(x_1, x_2), NAND(x_1, x_2))$

Multilayer Perceptron (MLP)



Multilayer Perceptron (MLP)



$$\mathbf{net}_j = \sum_{i=1}^{d} \mathbf{x}_i \mathbf{w}_{ji} + \mathbf{w}_{j0} = \sum_{i=0}^{d} \mathbf{x}_i \mathbf{w}_{ji} \equiv \mathbf{w}_j^t . \mathbf{x},$$

$$y_j = f(net_j)$$

$$net_k = \sum_{j=1}^{n_H} y_j w_{kj} + w_{k0} = \sum_{j=0}^{n_H} y_j w_{kj} = w_k^t . y, \qquad z_k = f(net_k)$$

Multilayer Perceptron (MLP)

$$J(w) = \frac{1}{2}\sum_{k=1}^{c}(t_k - z_k)^2 = \frac{1}{2}\|t - z\|^2$$

$$\Delta w = -\eta \frac{\partial J}{\partial w}$$

$$\frac{\partial J}{\partial w_{ki}} = \frac{\partial J}{\partial net_k} \cdot \frac{\partial net_k}{\partial w_{ki}} = -\delta_k \frac{\partial net_k}{\partial w_{ki}}$$

$$\delta_k = -\frac{\partial J}{\partial net_k} = -\frac{\partial J}{\partial z_k} \cdot \frac{\partial z_k}{\partial net_k} = (t_k - z_k) f'(net_k)$$

$$\frac{\partial net_k}{\partial w_{kj}} = y_j$$

$$\Delta w_{kj} = \eta \delta_k y_j = \eta (t_k - z_k) f'(net_k) y_j$$

Multilayer Perceptron (MLP)

$$\frac{\partial J}{\partial w_{ji}} = \frac{\partial J}{\partial y_j} \cdot \frac{\partial y_j}{\partial net_j} \cdot \frac{\partial net_j}{\partial w_{ji}}$$

$$\frac{\partial J}{\partial y_j} = \frac{\partial}{\partial y_j}\left[\frac{1}{2}\sum_{k=1}^{c}(t_k - z_k)^2\right] = -\sum_{k=1}^{c}(t_k - z_k)\frac{\partial z_k}{\partial y_j}$$

$$= -\sum_{k=1}^{c}(t_k - z_k)\frac{\partial z_k}{\partial net_k} \cdot \frac{\partial net_k}{\partial y_j} = -\sum_{k=1}^{c}(t_k - z_k) f'(net_k) w_{kj}$$

$$\delta_j \equiv f'(net_j)\sum_{k=1}^{c} w_{kj}\delta_k$$

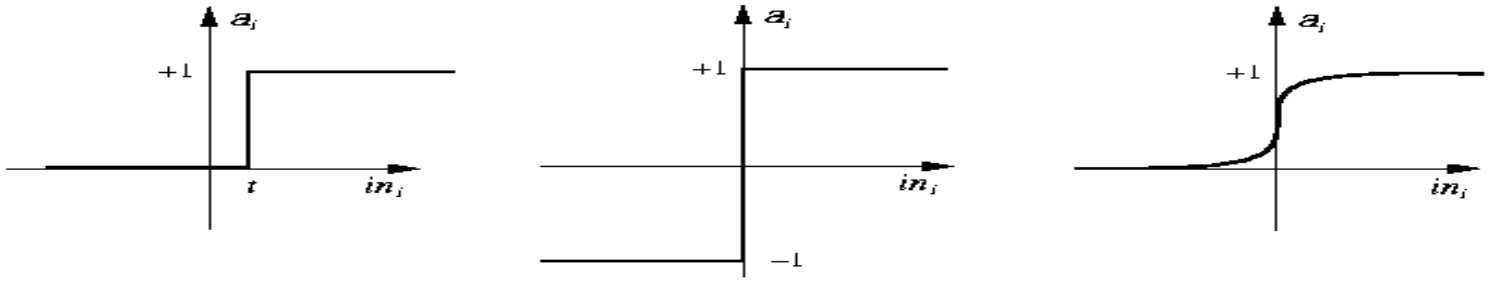
$$\Delta w_{ji} = \eta x_i \delta_j = \eta \underbrace{\left[\Sigma w_{kj}\delta_k\right] f'(net_j)}_{\delta_j} x_i$$

Types of Layers

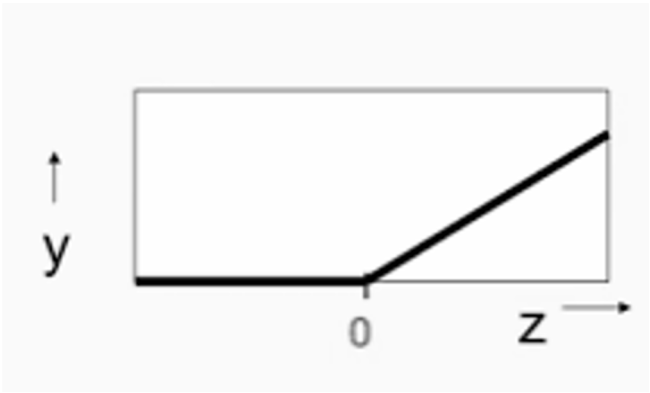
- The input layer.
  - Introduces input values into the network.
  - No activation function or other processing.
- The hidden layer(s).
  - Perform classification of features
  - Two hidden layers are sufficient to solve any problem
  - Features imply more layers may be better
- The output layer.
  - Functionally just like the hidden layers
  - Outputs are passed on to the world outside the neural network.

Activation functions

- Transforms neuron's input into output.



(a) Step function (b) Sign function (c) Sigmoid function



Rectified Linear Unit

Backpropagation Algorithm

Initialize **w** to some small random value

Do

- For each training example $<(x_1,\ldots x_n),t>$ Do
  - compute the network outputs o_k
  - For each output unit k, compute $\delta_k=o_k(1-o_k)(t_k-o_k)$
  - For each hidden unit j, $\delta_j=o_j(1-o_j)\sum_k w_{kj}\,\delta_k$
  - Compute $w_{ji}=w_{ji}+\Delta w_{ji}$ where $\Delta w_{ji}=\eta\,\delta_j\,x_i$
  - Compute $w_{kj}=w_{kj}+\Delta w_{kj}$ where $\Delta w_{kj}=\eta\delta_k y_j$

  Until the termination condition is met.

  Return **w**

Universal Function Approximator

A one hidden layer FFNN with sufficiently large number of hidden nodes can approximate any function (Hornik, 1991)

Handwritten Character Recognition

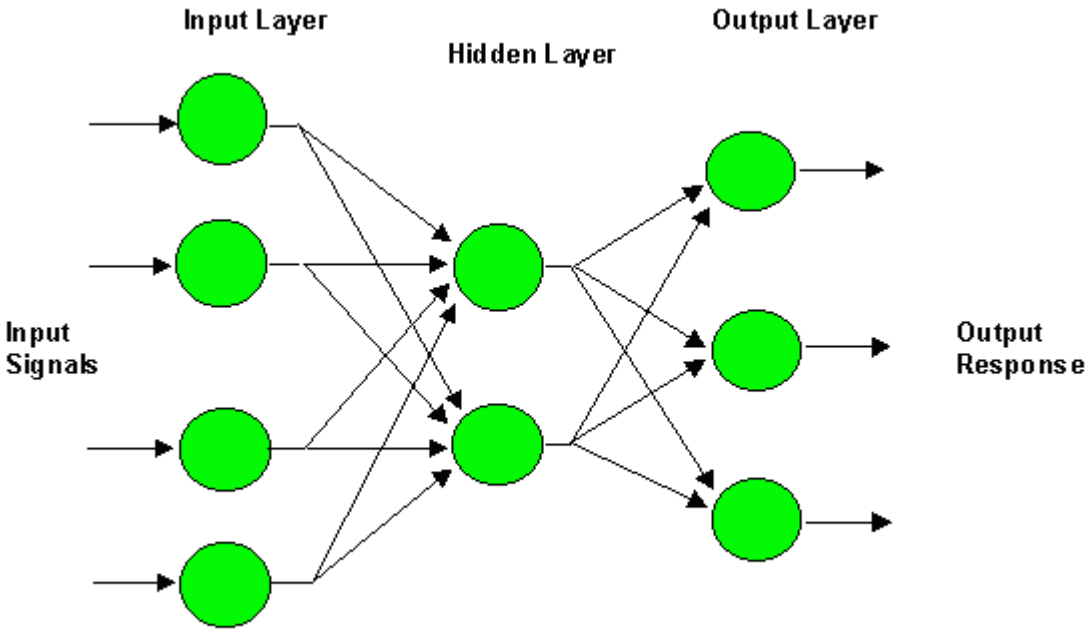


Image size= 100 x 100
No. of nodes at hidden layer= 10^6

No. of Classes =26
No. of Weights to be learned= 10^{10}

References

- Neural Networks for Pattern Recognition", Bishop, C.M., 1996
- Deep Belief Nets, 2007 NIPS tutorial , G . Hinton
- Slides adapted from Andrew NG and G . Hinton
- https://www.macs.hw.ac.uk/~dwcorne/

Thanks