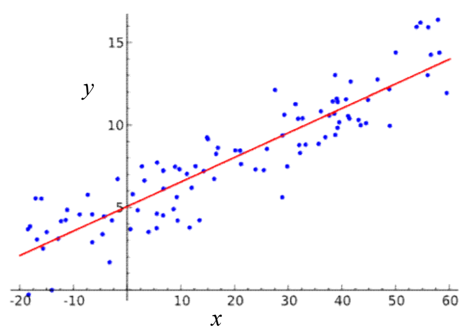
Linear Method for Classification

Linear Regression

- Linear approach to model the relationship between a scalar response, (y) (or dependent variable) and one or more predictor variables, (x *or* $\mathbf{x}$) (or independent variables)
- The response is going to be the linear function of input (one or more independent variables)
- Optimal coefficient vector $\mathbf{w}$ is given by

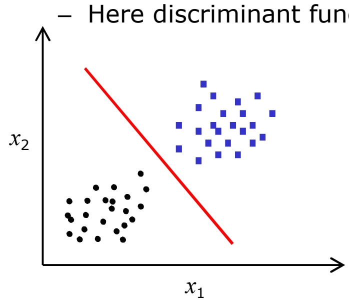
$$\hat{\mathbf{w}} = \left(\mathbf{X}^\top \mathbf{X}\right)^{-1} \mathbf{X}^\top \mathbf{y}$$

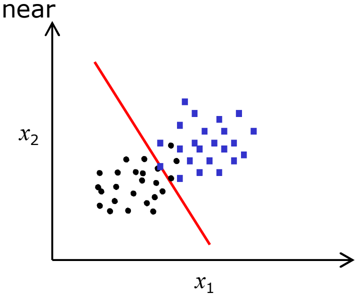


2

Linear Method for Classification

- The boundary that separates the region of classes is linear
- Separating surface is linear i.e. hyperplane
- A hyperplane that best fit the region of separation between the classes
- Discriminant function: Function that indicate the boundary between the classes
  - Here discriminant function is linear





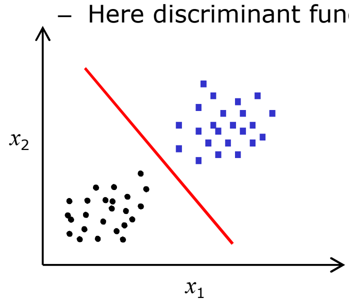
- Discriminant function in 2-dimensional space :

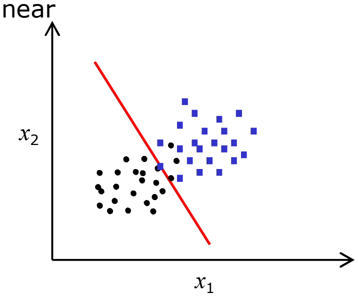
$\mathbf{x}_n=[x_{n1}, x_{n2}]^\mathsf{T}$ $$f(\mathbf{x}_n, w_1, w_2, w_0) = w_1\, x_{n1} + w_2\, x_{n2} + w_0$$

3

Linear Method for Classification

- The boundary that separates the region of classes is linear
- Separating surface is linear i.e. hyperplane
- A hyperplane that best fit the region of separation between the classes
- Discriminant function: Function that indicate the boundary between the classes
  - Here discriminant function is linear





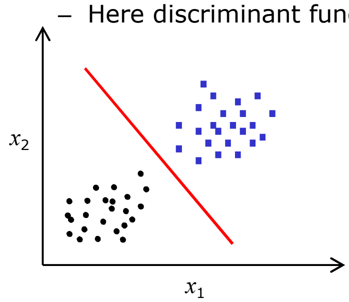
- Discriminant function in 2-dimensional space :

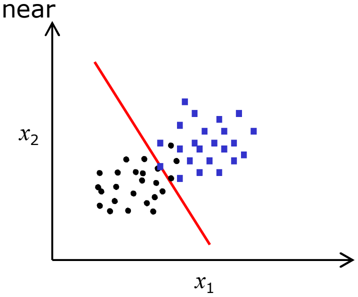
$\mathbf{x}_n=[x_{n1}, x_{n2}]^\mathsf{T}$ $$f(\mathbf{x}_n, w_1, w_2, w_0) = w_1\, x_{n1} + w_2\, x_{n2} + w_0 = 0$$

4

Linear Method for Classification

- The boundary that separates the region of classes is linear
- Separating surface is linear i.e. hyperplane
- A hyperplane that best fit the region of separation between the classes
- Discriminant function: Function that indicate the boundary between the classes
  - Here discriminant function is linear





- Discriminant function in 2-dimensional space :

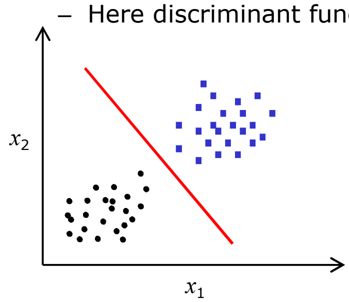
$\mathbf{x}_n = [x_{n1}, x_{n2}]^{\mathsf{T}}$

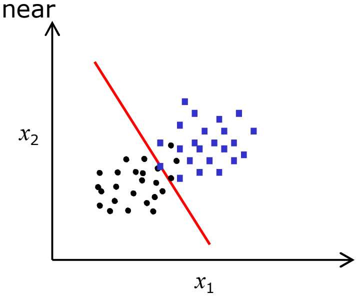
$$x_{n2} = -\frac{w_1}{w_2} x_{n1} - \frac{w_0}{w_2} = m x_{n1} + c$$

5

Linear Method for Classification

- The boundary that separates the region of classes is linear
- Separating surface is linear i.e. hyperplane
- A hyperplane that best fit the region of separation between the classes
- Discriminant function: Function that indicate the boundary between the classes
  - Here discriminant function is linear





- Discriminant function in d-dimensional space :

$\mathbf{x}_n = [x_{n1}, x_{n2}, \ldots, x_{nd}]^{\mathsf{T}}$

$$f(\mathbf{x}_n, \mathbf{w}) = \mathbf{w}^{\mathsf{T}} \mathbf{x}_n + w_0 = \sum_{i=0}^{d} w_i x_i$$

6

Two classes of Approaches for Linear Classification

1. Modeling a discriminating function:
   - For each class, a linear discriminant function $f_i(\mathbf{x},\mathbf{w}_i)$ is defined
   - Let C_1, C_2, …, C_i, …, C_M be the M classes
   - Let $f_i(\mathbf{x},\mathbf{w}_i)$ be the linear discriminant function for i^{th} class

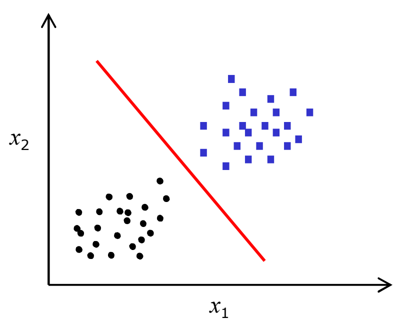
$$\text{Class label for } \mathbf{x} = \arg\max_i f_i(\mathbf{x},\mathbf{w}_i) \qquad i = 1, 2, \ldots, M$$

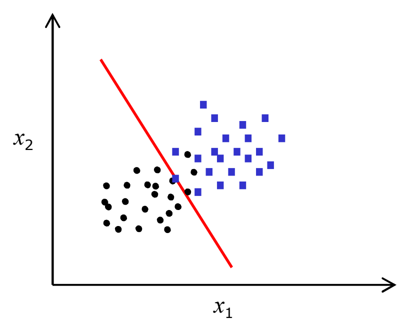
   - Discriminant function is defined independent of the classes
   - Linear regression can be used to learn linear discriminant function
     - Do the linear regression by considering dependent variable as indicator variable (categorical variable)
   - Logistic regression

7

Two classes of Approaches for Linear Classification

2. Directly learn a discriminant function (hyperplane):
   - Classic method: Discriminant function between the classes is learnt





   - Perceptron (linear discriminant function is learnt)
   - Support vector machine (SVM) (linear discriminant function is learnt)
   - Neural networks (when the discriminant function is nonlinear)

8

Classification Using Linear Regression

- Given:-Training data: $\mathcal{D} = \{\mathbf{x}_n, \mathbf{y}_n\}_{n=1}^{N}, \; \mathbf{x}_n \in \mathbb{R}^d$ and $\mathbf{y}_n \in \mathbb{R}^M$
  - $\mathbf{x}_n$ is input vector (d dependent variable)
  - There are M classes, represented by M indicator variables
  - $\mathbf{y}_n$ is response vector (dependent variables) which is M-dimensional binary vector i.e. one of the M values is 1
- Illustration: Iris (Flower) Data – 3 classes

X				Y		
Sepal-Length	Sepal_Width	Petal_Length	Petal_Width	Class1	Class2	Class3
5.1	3.5	1.4	0.2	1	0	0
4.9	3.0	1.4	0.2	1	0	0
7.0	3.2	4.7	1.4	0	1	0
6.4	3.2	4.5	1.5	0	1	0
6.3	3.3	6.0	2.5	0	0	1
5.8	2.7	5.1	1.9	0	0	1

9

Classification Using Linear Regression

- Given:-Training data: $\mathcal{D} = \{\mathbf{x}_n, \mathbf{y}_n\}_{n=1}^{N}, \; \mathbf{x}_n \in \mathbb{R}^d$ and $\mathbf{y}_n \in \mathbb{R}^M$
  - $\mathbf{x}_n$ is input vector (d dependent variable)
  - There are M classes, represented by M indicator variables
  - $\mathbf{y}_n$ is response vector (dependent variables) which is M-dimensional binary vector i.e. one of the M values is 1
  - For N examples, $\mathbf{X}$ is data matrix of size N x $(d+1)$ and $\mathbf{Y}$ is response matrix of size N x M
- Linear regression on response vector: $\hat{\mathbf{W}} = \left(\mathbf{X}^{\top}\mathbf{X}\right)^{-1}\mathbf{X}^{\top}\mathbf{Y}$
  - $\hat{\mathbf{W}}$ is of the size $(d+1)$ x M

    $$\hat{\mathbf{W}} = \left[\hat{\mathbf{w}}_1, \hat{\mathbf{w}}_2, \ldots, \hat{\mathbf{w}}_M\right]$$

  - Each column of $\hat{\mathbf{W}}$ is $(d+1)$ coefficients corresponding to a class

10

Classification Using Linear Regression

- For any test example $\mathbf{x}$, the discriminant value for class i is:

$$f_i(\mathbf{x}, \hat{\mathbf{w}}_i) = \hat{\mathbf{w}}_i^{\mathsf{T}} \mathbf{x} = \sum_{j=0}^{d} \hat{w}_{ij} x_i$$

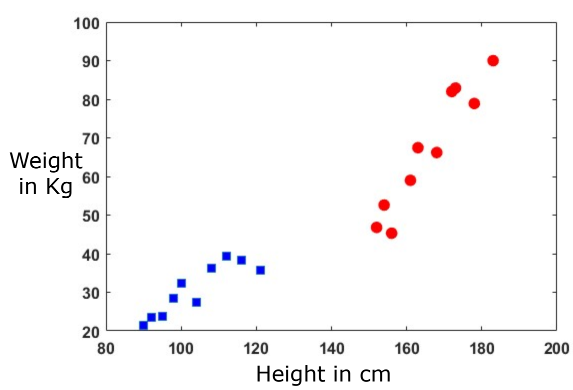
Class label for $\mathbf{x}$ = $\underset{i}{\arg\max}\, f_i(\mathbf{x}, \hat{\mathbf{w}}_i) \qquad i = 1, 2, \ldots, M$

11

Illustration of Classification using Linear Regression

Height	Weight	Class y1	Class y2
90	21.5	1	0
95	23.67	1	0
100	32.45	1	0
116	38.21	1	0
98	28.43	1	0
108	36.32	1	0
104	27.38	1	0
112	39.28	1	0
121	35.8	1	0
92	23.56	1	0
152	46.8	0	1
178	78.9	0	1
163	67.45	0	1
173	82.9	0	1
154	52.6	0	1
168	66.2	0	1
183	90	0	1
172	82	0	1
156	45.3	0	1
161	59	0	1

- Number of training examples (N) = 20
- Dimension of a training example = 2
- Number of classes: 2
- Each output variable is a 2-dimensional binary vector
- Class: Child (C1) Adult (C2)



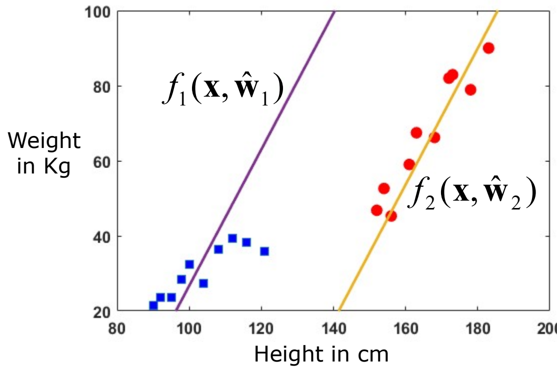
12

Illustration of Classification using Linear Regression

Height	Weight	Class y1	Class y2
90	21.5	1	0
95	23.67	1	0
100	32.45	1	0
116	38.21	1	0
98	28.43	1	0
108	36.32	1	0
104	27.38	1	0
112	39.28	1	0
121	35.8	1	0
92	23.56	1	0
152	46.8	0	1
178	78.9	0	1
163	67.45	0	1
173	82.9	0	1
154	52.6	0	1
168	66.2	0	1
183	90	0	1
172	82	0	1
156	45.3	0	1
161	59	0	1

- Training: $\hat{\mathbf{W}} = \left(\mathbf{X}^\top \mathbf{X}\right)^{-1} \mathbf{X}^\top \mathbf{Y}$
- $\mathbf{X}$ is data matrix of size 20×3
- $\mathbf{Y}$ is response matrix of size 20×2

$$\hat{\mathbf{W}} = [\hat{\mathbf{w}}_1 \quad \hat{\mathbf{w}}_2] = \begin{bmatrix} 2.8897 & -1.8897 \\ -0.0222 & 0.0222 \\ 0.0122 & -0.0122 \end{bmatrix}$$

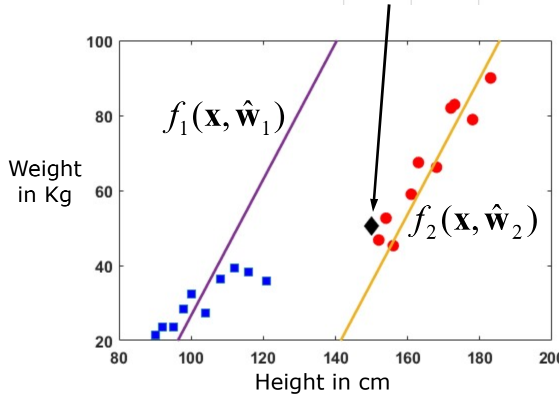


13

Illustration of Classification using Linear Regression

Height	Weight	Class y1	Class y2
90	21.5	1	0
95	23.67	1	0
100	32.45	1	0
116	38.21	1	0
98	28.43	1	0
108	36.32	1	0
104	27.38	1	0
112	39.28	1	0
121	35.8	1	0
92	23.56	1	0
152	46.8	0	1
178	78.9	0	1
163	67.45	0	1
173	82.9	0	1
154	52.6	0	1
168	66.2	0	1
183	90	0	1
172	82	0	1
156	45.3	0	1
161	59	0	1

Test Example: 150 | 50.6



$f_1(\mathbf{x}, \hat{\mathbf{w}}_1) = 0.1842 \qquad f_2(\mathbf{x}, \hat{\mathbf{w}}_2) = 0.8158$

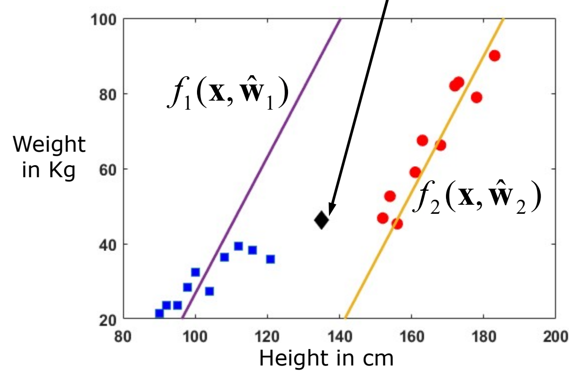
- Class: Adult (C2)

14

Illustration of Classification using Linear Regression

Height	Weight	Class y1	Class y2
90	21.5	1	0
95	23.67	1	0
100	32.45	1	0
116	38.21	1	0
98	28.43	1	0
108	36.32	1	0
104	27.38	1	0
112	39.28	1	0
121	35.8	1	0
92	23.56	1	0
152	46.8	0	1
178	78.9	0	1
163	67.45	0	1
173	82.9	0	1
154	52.6	0	1
168	66.2	0	1
183	90	0	1
172	82	0	1
156	45.3	0	1
161	59	0	1

Test Example: | 135 | 46.29 |



$f_1(\mathbf{x}, \hat{\mathbf{w}}_1) = 0.4639$ $f_2(\mathbf{x}, \hat{\mathbf{w}}_2) = 0.5361$

- Class: Adult (C2)

15

Classification Using Linear Regression

- Dependent variable is categorical (indicator variable)
- Output is multiple outputs (multiple dependent variables)

- If the input $\mathbf{x}$ belongs to C_i, then y_i is 1
- The expected output for $\mathbf{x}$ should be close to 1
- During linear regression for classification, we are trying to predict the expected output value
- In other way, we are trying to predict probability of class

$$\boxed{\mathrm{E}[y_i \mid \mathbf{x}] = P(y_i = C_i \mid \mathbf{x})}$$

- This is the ideal situation
- Linear regression gives the hope of getting this
- The notion of predicting probability of class is given nicely by *logistic regression*

16

Logistic Regression

Logistic Regression

- Requirement: The discriminant function $f_i(\mathbf{x},\mathbf{w}_i)$ should give the probability of class C_i

$$\boxed{\mathrm{E}[y_i \mid \mathbf{x}] = P(y_i = C_i \mid \mathbf{x})}$$

- Look for some kind of transformation of probability and fit that
- Logit transformation: $\boxed{\log\left(\frac{P(\mathbf{x})}{1-P(\mathbf{x})}\right)}$
- *2-class classification*:
  - Class label: 0 or 1
  - $P(\mathbf{x})$ is $P(C_i=1|\mathbf{x})$ i.e. probability that output is 1 given input (probability of success)
  - $1\text{-}P(\mathbf{x})$ is $P(C_i=0|\mathbf{x})$ i.e. probability that output is 0 given input (probability of failure)

Logistic Regression

- Logit function: Log of odds function
- Odds function: $\boxed{\dfrac{P(\mathbf{x})}{1-P(\mathbf{x})}}$
  - Probability of success divided by the probability of failure
- Fit a linear model to logit function:

$$\boxed{\log\left(\frac{P(\mathbf{x})}{1-P(\mathbf{x})}\right) = w_0 + w_1 x_1 + \ldots + w_d x_d = \mathbf{w}^\mathsf{T}\hat{\mathbf{x}}}$$

where $\mathbf{w} = [w_0, w_1, \ldots, w_d]^\mathsf{T}$ and $\hat{\mathbf{x}} = [1, x_1, \ldots, x_d]^\mathsf{T}$

  - For 1-dimensional (d=1) space, x

$$\log\left(\frac{P(x)}{1-P(x)}\right) = w_0 + w_1 x \qquad \frac{P(x)}{1-P(x)} = e^{(w_0 + w_1 x)}$$

19

Logistic Regression

- Logit function: Log of odds function
- Odds function: $\boxed{\dfrac{P(\mathbf{x})}{1-P(\mathbf{x})}}$
  - Probability of success divided by the probability of failure
- Fit a linear model to logit function:

$$\boxed{\log\left(\frac{P(\mathbf{x})}{1-P(\mathbf{x})}\right) = \mathbf{w}^\mathsf{T}\hat{\mathbf{x}}}$$

where $\mathbf{w} = [w_0, w_1, \ldots, w_d]^\mathsf{T}$ and $\hat{\mathbf{x}} = [1, x_1, \ldots, x_d]^\mathsf{T}$

  - For 1-dimensional (d=1) space, x

$$\frac{P(x)}{1-P(x)} = e^{(w_0 + w_1 x)}$$

$$P(x) = \frac{e^{(w_0 + w_1 x)}}{1 + e^{(w_0 + w_1 x)}} = \frac{1}{1 + e^{-(w_0 + w_1 x)}}$$

20

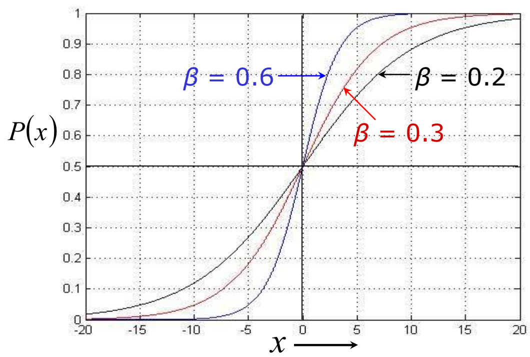
Logistic Regression

$$P(x) = \frac{1}{1+e^{-(w_0+w_1 x)}}$$

- This function is a sigmoidal function, specifically called as logistic function
- Logistic function:

$$P(x) = \frac{1}{1+e^{-(w_0+w_1 x)}}$$

$$P(x) = \frac{1}{1+e^{-(\beta x)}}$$



21

Logistic Regression

- Logit function: Log of odds function
- Odds function: $\boxed{\dfrac{P(\mathbf{x})}{1-P(\mathbf{x})}}$
  - Probability of success divided by the probability of failure
- Fit a linear model to logit function: $\boxed{\log\left(\dfrac{P(\mathbf{x})}{1-P(\mathbf{x})}\right) = \mathbf{w}^\top \hat{\mathbf{x}}}$
  - For d-dimensional space, $\mathbf{x}=[x_1, x_2, \ldots, x_d]^\top$

$$\log\left(\frac{P(\mathbf{x})}{1-P(\mathbf{x})}\right) = w_0 + w_1 x_1 + \ldots + w_d x_d = \mathbf{w}^\top \hat{\mathbf{x}}$$

where $\mathbf{w} = [w_0, w_1, \ldots, w_d]^\top$ and $\hat{\mathbf{x}} = [1, x_1, \ldots, x_d]^\top$

$$\frac{P(\mathbf{x})}{1-P(\mathbf{x})} = e^{(\mathbf{w}^\top \hat{\mathbf{x}})}$$

22

Logistic Regression

- Logit function: Log of odds function
- Odds function: $\boxed{\dfrac{P(\mathbf{x})}{1-P(\mathbf{x})}}$
  - Probability of success divided by the probability of failure
- Fit a linear model to logit function: $\boxed{\log\left(\dfrac{P(\mathbf{x})}{1-P(\mathbf{x})}\right) = \mathbf{w}^\top \hat{\mathbf{x}}}$
  - For d-dimensional space, $\mathbf{x}=[x_1, x_2, \ldots, x_d]^\top$

$$\frac{P(\mathbf{x})}{1-P(\mathbf{x})} = e^{(\mathbf{w}^\top \hat{\mathbf{x}})}$$

where $\mathbf{w} = [w_0, w_1, \ldots, w_d]^\top$

and $\hat{\mathbf{x}} = [1, x_1, \ldots, x_d]^\top$

$$P(\mathbf{x}) = \frac{e^{(\mathbf{w}^\top \hat{\mathbf{x}})}}{1+e^{(\mathbf{w}^\top \hat{\mathbf{x}})}}$$

$$P(\mathbf{x}) = \frac{1}{1+e^{-(\mathbf{w}^\top \hat{\mathbf{x}})}}$$

What about the classifier learning here?

It is still a linear classifier – Boundary is linear surface i.e. hyperplane

23

Logistic Regression

- Logit function: Log of odds function
- Odds function: $\boxed{\dfrac{P(\mathbf{x})}{1-P(\mathbf{x})}}$
  - Probability of success divided by the probability of failure
- Fit a linear model to logit function: $\boxed{\log\left(\dfrac{P(\mathbf{x})}{1-P(\mathbf{x})}\right) = \mathbf{w}^\top \hat{\mathbf{x}}}$
  - For d-dimensional space, $\mathbf{x}=[x_1, x_2, \ldots, x_d]^\top$

$$\frac{P(\mathbf{x})}{1-P(\mathbf{x})} = e^{(\mathbf{w}^\top \hat{\mathbf{x}})}$$

where $\mathbf{w} = [w_0, w_1, \ldots, w_d]^\top$

and $\hat{\mathbf{x}} = [1, x_1, \ldots, x_d]^\top$

$$P(\mathbf{x}) = \frac{e^{(\mathbf{w}^\top \hat{\mathbf{x}})}}{1+e^{(\mathbf{w}^\top \hat{\mathbf{x}})}}$$

$$P(\mathbf{x}) = \frac{1}{1+e^{-(\mathbf{w}^\top \hat{\mathbf{x}})}}$$

For any test example $\mathbf{x}$:

If $P(\mathbf{x}) \geq 0.5$ then $\mathbf{x}$ is assigned with label 1

If $P(\mathbf{x}) < 0.5$ then $\mathbf{x}$ is assigned with label 0

24

Estimation of Parameter in Logistic Regression

- Criterion considered is different than linear regression to estimate the parameter
- Optimize the likelihood of data
- As that goal is to model the probability of class, we are maximizing the likelihood of data
- **Maximum likelihood (ML) method of parameter estimation**
- Given:- Training data: $\mathcal{D} = \{\mathbf{x}_n, y_n\}_{n=1}^{N}$, $\mathbf{x}_n \in \mathbb{R}^d$ and $y_n \in \{1,0\}$
- Data of a class is represented by parameter vector: $\mathbf{w}=[w_0, w_1,\ldots, w_d]^\mathsf{T}$ (parameter of linear function)
- Unknown: $\mathbf{w}$
- Likelihood of $\mathbf{x}_n$: $P(\mathbf{x}_n \mid \mathbf{w}) = P(\mathbf{x}_n)^{y_n}(1-P(\mathbf{x}_n))^{(1-y_n)}$

Probability that $\mathbf{x}$ has label 1

Probability that $\mathbf{x}$ has label 0

25

Estimation of Parameter in Logistic Regression

- Different criterion than linear regression to estimate the parameter
- Optimize the likelihood of data
- As that goal is to model the probability of class, we are maximizing the likelihood of data
- **Maximum likelihood (ML) method of parameter estimation**
- Given:- Training data: $\mathcal{D} = \{\mathbf{x}_n, y_n\}_{n=1}^{N}$, $\mathbf{x}_n \in \mathbb{R}^d$ and $y_n \in \{1,0\}$
- Data of a class is represented by parameter vector: $\mathbf{w}=[w_0, w_1,\ldots, w_d]^\mathsf{T}$ (parameter of linear function)
- Unknown: $\mathbf{w}$

Binomial distribution (Bernoulli Distribution)

- Likelihood of $\mathbf{x}_n$: $P(\mathbf{x}_n \mid \mathbf{w}) = \boxed{P(\mathbf{x}_n)^{y_n}(1-P(\mathbf{x}_n))^{(1-y_n)}}$
- Total data likelihood: $P(\mathcal{D} \mid \mathbf{w}) = \prod_{n=1}^{N} P(\mathbf{x}_n \mid \mathbf{w})$

26

Estimation of Parameter in Logistic Regression

- Different criterion than linear regression to estimate the parameter
- Optimize the likelihood of data
- As that goal is to model the probability of class, we are maximizing the likelihood of data
- **Maximum likelihood (ML) method of parameter estimation**
- Given:- Training data: $\mathcal{D} = \{\mathbf{x}_n, y_n\}_{n=1}^{N}$, $\mathbf{x}_n \in \mathbb{R}^d$ and $y_n \in \{1,0\}$
- Data of a class is represented by parameter vector: $\mathbf{w}=[w_0, w_1,\ldots, w_d]^\mathsf{T}$ (parameter of linear function)
- Unknown: $\mathbf{w}$
- Likelihood of $\mathbf{x}_n$: $P(\mathbf{x}_n \mid \mathbf{w}) = P(\mathbf{x}_n)^{y_n}(1 - P(\mathbf{x}_n))^{(1-y_n)}$
- Total data likelihood: $P(\mathcal{D} \mid \mathbf{w}) = \prod_{n=1}^{N} P(\mathbf{x}_n)^{y_n}(1 - P(\mathbf{x}_n))^{(1-y_n)}$

27

Estimation of Parameter in Logistic Regression

- Total data log likelihood:

$$l(\mathbf{w}) = \ln\big(P(\mathcal{D} \mid \mathbf{w})\big)$$

$$l(\mathbf{w}) = \sum_{n=1}^{N} y_n \ln(P(\mathbf{x}_n)) + (1 - y_n)\ln(1 - P(\mathbf{x}_n))$$

- Choose the parameters for which the total data log likelihood is maximum:

$$\mathbf{w}_{\text{ML}} = \arg\max_{\mathbf{w}} l(\mathbf{w})$$

- Cost function for optimization:

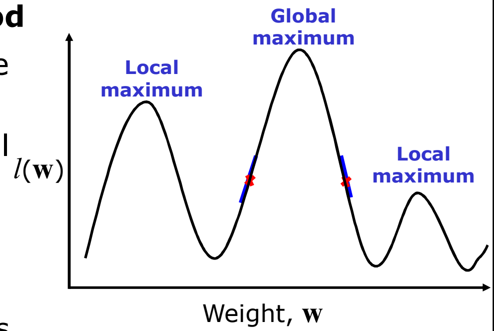
$$l(\mathbf{w}) = \sum_{n=1}^{N} y_n \ln(P(\mathbf{x}_n)) + (1 - y_n)\ln(1 - P(\mathbf{x}_n))$$

- Conditions for optimality: $\dfrac{\partial\, l(\mathbf{w})}{\partial \mathbf{w}} = \mathbf{0}$
- Unfortunately, solving this, no closed form expression for $\mathbf{w}$ is obtained
- Solution: Gradient accent method

28

Estimation of Parameter in Logistic Regression

- **Gradient accent method**
- It is an iterative procedure
- We start with an initial value for $\mathbf{w}$
- At each iteration:
  - Estimate change in $\mathbf{w}$
  - The change in $\mathbf{w}$ ($\Delta\mathbf{w}$) is proportional to the slope (gradient) of the likelihood surface
  - Then, the $\mathbf{w}$ is updated using $\Delta\mathbf{w}$
- This indicate, we move in the positive slope of the likelihood surface, likelihood is maximum in each iteration



$$\Delta\mathbf{w} \propto -\frac{\partial\, l(\mathbf{w})}{\partial\mathbf{w}} \qquad \Delta\mathbf{w} = -\eta\,\frac{\partial\, l(\mathbf{w})}{\partial\mathbf{w}}$$

where $0 \le \eta \le 1$ is proportionality constant

29

Estimation of Parameter in Logistic Regression – Gradient Accent Method

- Given a training dataset, the goal is to maximize the likelihood function with respect to the parameters of linear function

1. Initialize the $\mathbf{w}$
   - Evaluate the initial value of the log likelihood, $l(\mathbf{w})$
2. Determine the change in $\mathbf{w}$ ($\Delta\mathbf{w}$): $\Delta\mathbf{w} = -\eta\,\dfrac{\partial\, l(\mathbf{w})}{\partial\mathbf{w}}$
3. Update the $\mathbf{w}$: $\mathbf{w} = \mathbf{w} + \Delta\mathbf{w}$
4. Evaluate the log likelihood and check for convergence of the log likelihood
   - If the convergence criterion is not satisfied repeat from steps 2 to 4

- Convergence criterion: Difference between log likelihoods of successive iterations fall below a threshold (E.g. threshold=10^{-3})

30

Illustration of Classification using Logistic Regression

Height	Weight	Class
90	21.5	0
95	23.67	0
100	32.45	0
116	38.21	0
98	28.43	0
108	36.32	0
104	27.38	0
112	39.28	0
121	35.8	0
92	23.56	0
152	46.8	1
178	78.9	1
163	67.45	1
173	82.9	1
154	52.6	1
168	66.2	1
183	90	1
172	82	1
156	45.3	1
161	59	1

- Number of training examples (N) = 20
- Dimension of a training example = 2
- Class label attribute is 3[rd] dimension
- Class:
  - Child (0)
  - Adult (1)

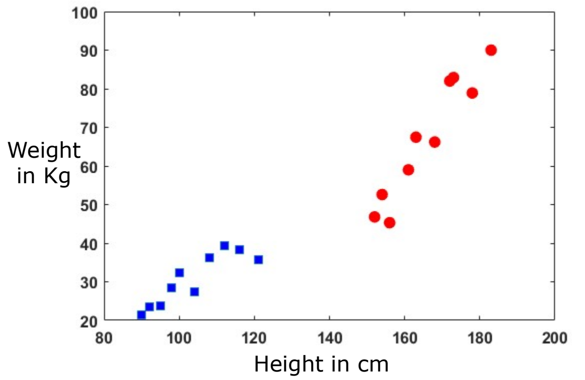


Illustration of Classification using Logistic Regression

Height	Weight	Class
90	21.5	0
95	23.67	0
100	32.45	0
116	38.21	0
98	28.43	0
108	36.32	0
104	27.38	0
112	39.28	0
121	35.8	0
92	23.56	0
152	46.8	1
178	78.9	1
163	67.45	1
173	82.9	1
154	52.6	1
168	66.2	1
183	90	1
172	82	1
156	45.3	1
161	59	1

- Training:

$$\mathbf{w} = \begin{bmatrix} -378.2085 \\ 2.2065 \\ 1.8818 \end{bmatrix}$$

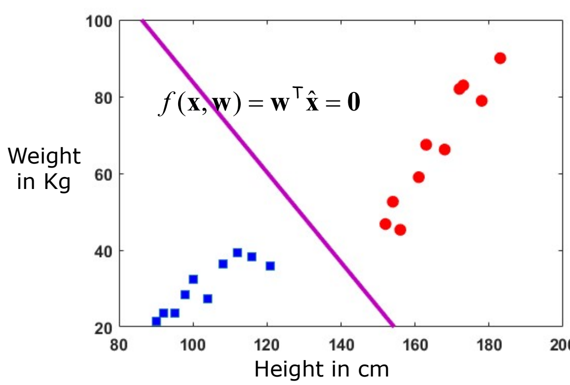
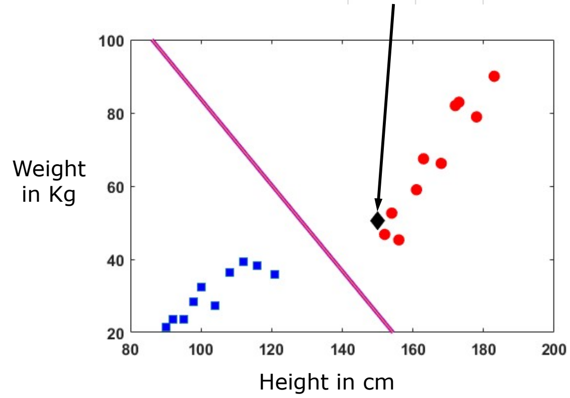


Illustration of Classification using Linear Regression

Height	Weight	Class
90	21.5	0
95	23.67	0
100	32.45	0
116	38.21	0
98	28.43	0
108	36.32	0
104	27.38	0
112	39.28	0
121	35.8	0
92	23.56	0
152	46.8	1
178	78.9	1
163	67.45	1
173	82.9	1
154	52.6	1
168	66.2	1
183	90	1
172	82	1
156	45.3	1
161	59	1

Test Example:

150	50.6



$$\mathbf{w}^{\mathsf{T}}\hat{\mathbf{x}} = 47.9851 \qquad P(\mathbf{x}) = \frac{1}{1+e^{-(\mathbf{w}^{\mathsf{T}}\hat{\mathbf{x}})}} = 1$$

- Class: Adult (C2)

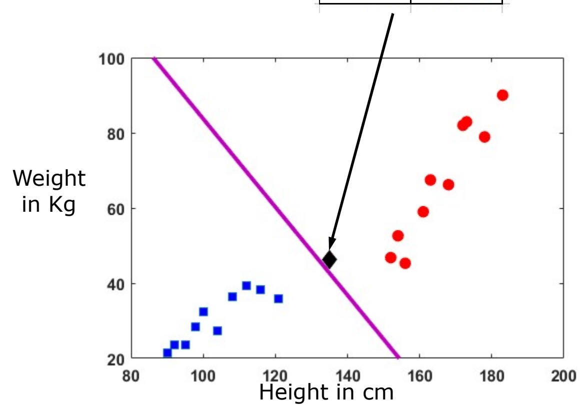
33

Illustration of Classification using Linear Regression

Height	Weight	Class
90	21.5	0
95	23.67	0
100	32.45	0
116	38.21	0
98	28.43	0
108	36.32	0
104	27.38	0
112	39.28	0
121	35.8	0
92	23.56	0
152	46.8	1
178	78.9	1
163	67.45	1
173	82.9	1
154	52.6	1
168	66.2	1
183	90	1
172	82	1
156	45.3	1
161	59	1

Test Example:

135	46.29



$$\mathbf{w}^{\mathsf{T}}\hat{\mathbf{x}} = 6.7771 \qquad P(\mathbf{x}) = \frac{1}{1+e^{-(\mathbf{w}^{\mathsf{T}}\hat{\mathbf{x}})}} = 0.9$$

- Class: Adult (C2)

34

Logistic Regression

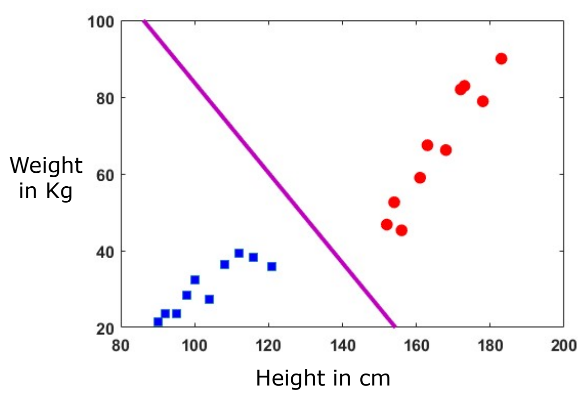
- Logistic regression is a linear classifier
- Logistic regression looks simple, but yields a very powerful classifier
- It is used not just building classifier, but also used in sensitivity analysis
- Logistic regression is used to identify how each attribute contribute to output
  - How much each attribute is important for predicting class label
- Perform logistic regression and observe $\mathbf{w}$
- The value of each element of $\mathbf{w}$ indicate how much each attribute is contributing to the output

35

Illustration of Sensitivity Analysis using Logistic Regression

Height	Weight	Class
90	21.5	0
95	23.67	0
100	32.45	0
116	38.21	0
98	28.43	0
108	36.32	0
104	27.38	0
112	39.28	0
121	35.8	0
92	23.56	0
152	46.8	1
178	78.9	1
163	67.45	1
173	82.9	1
154	52.6	1
168	66.2	1
183	90	1
172	82	1
156	45.3	1
161	59	1

- Training: $$\mathbf{w} = \begin{bmatrix} -378.2085 \\ 2.2065 \\ 1.8818 \end{bmatrix} = \begin{bmatrix} w_0 \\ w_1 \\ w_2 \end{bmatrix}$$



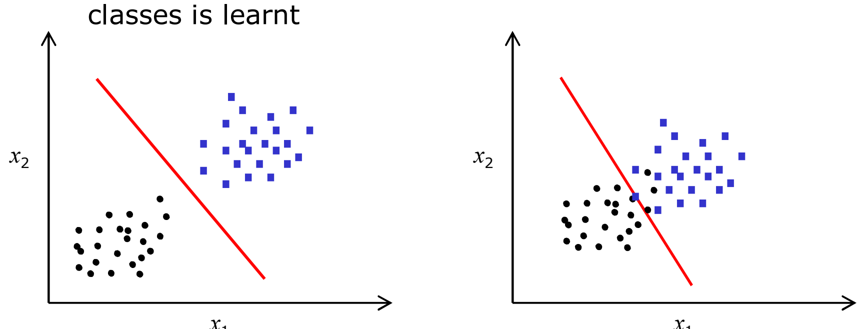
- Both the attributes are equally important

36

Discriminative Learning Methods for Classification

Two classes of Approaches for Linear Classification

1. Modeling a discriminating function:
   - Linear regression and Logistic regression
2. Directly learn a discriminant function (hyperplane):
   - Classic method: Discriminant function between the classes is learnt

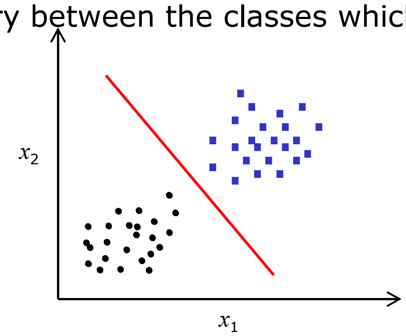


   - Perceptron (linear discriminant function is learnt)
   - Neural networks (when the discriminant function is nonlinear)

38

Discriminative Learning Methods

- Learn the surface that better separates the region of classes
- Learning discriminant function: Learns a function that maps input data to output
- Linear discriminant function: Function that indicate the boundary between the classes which is linear



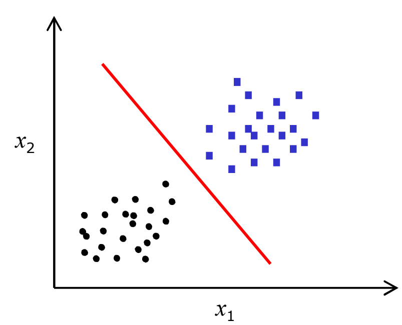
39

Linear Discriminant Function

- Regions of two classes are separable by a linear surface (line, plane or hyperplane)
- 2-dimensional space: The decision boundary is a **line** specified by

$$w_1 x_1 + w_2 x_2 + w_0 = 0$$

$$x_2 = -\frac{w_1}{w_2} x_1 - \frac{w_0}{w_2}$$



- d-dimensional space: The decision surface is a **hyperplane** specified by

$$w_d x_d + \ldots. + w_2 x_2 + w_1 x_1 + w_0 = \sum_{i=0}^{d} w_i x_i = \mathbf{w}^{\mathsf{T}} \hat{\mathbf{x}} = 0$$

where $\mathbf{w} = [w_0, w_1, \ldots, w_d]^{\mathsf{T}}$ and $\hat{\mathbf{x}} = [1, x_1, \ldots, x_d]^{\mathsf{T}}$

40

Discriminant Function of a Hyperplane

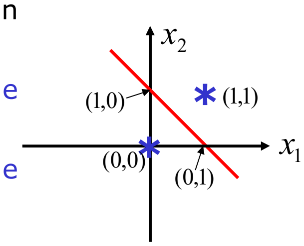
- The discriminant function of a hyperplane:

$$g(\mathbf{x}) = \sum_{i=1}^{d} w_i x_i + w_0 = \mathbf{w}^{\mathsf{T}}\mathbf{x} + w_0$$

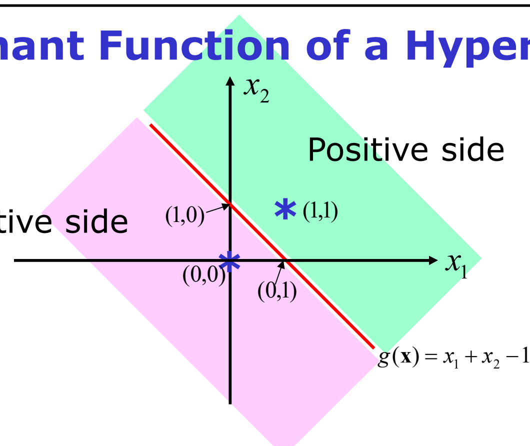
- For any point the lies on the hyperplane

$$g(\mathbf{x}) = \sum_{i=1}^{d} w_i x_i + w_0 = \mathbf{w}^{\mathsf{T}}\mathbf{x} + w_0 = 0$$

- Example:
  - Consider a straight line with its equation as $x_2+x_1-1=0$
  - Discriminant function of the straight line is $g(\mathbf{x})=x_2+x_1-1$
  - For points $(1,0)$ and $(0,1)$ that lie on this straight line $g(\mathbf{x})=0$
  - For the point $(0,0)$, $g(\mathbf{x})=-1$ i.e. the value of $g(\mathbf{x})$ is negative
  - For the point $(1,1)$, $g(\mathbf{x})=+1$ i.e. the value of $g(\mathbf{x})$ is positive



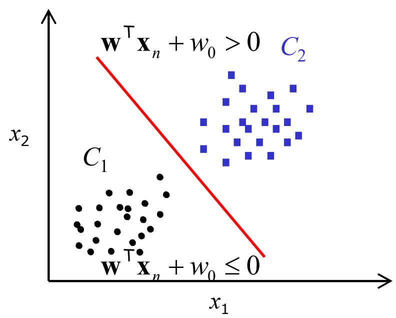
Discriminant Function of a Hyperplane



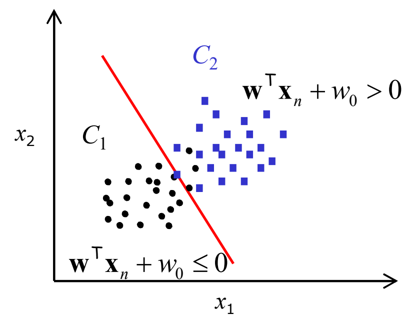
- A hyperplane has a positive side and a negative side
  - For any point on the positive side, the value of discriminant function, $g(\mathbf{x})$, is positive
  - For any point on the negative side, the value of discriminant function, $g(\mathbf{x})$, is negative

Perceptron Learning

- Given - training data: $\mathcal{D} = \{\mathbf{x}_n, y_n\}_{n=1}^{N}$, $\mathbf{x}_n \in \mathbb{R}^d$ and $y_n \in \{+1, -1\}$
- Goal: To estimate parameter vector $\mathbf{w}=[w_0, w_1, \ldots, w_d]^\mathsf{T}$
  - such that linear function (hyperplane) is places between the training data of two classes so that training error (classification error) is minimum



OR



Perceptron Learning

- Given - training data: $\mathcal{D} = \{\mathbf{x}_n, y_n\}_{n=1}^{N}$, $\mathbf{x}_n \in \mathbb{R}^d$ and $y_n \in \{+1, -1\}$

1. Initialize the $\mathbf{w}$ with random values
2. Choose a training example $\mathbf{x}_n$
3. Update the $\mathbf{w}$, if $\mathbf{x}_n$ is misclassified

   $$\mathbf{w} = \mathbf{w} + \eta\, \mathbf{x}_n, \text{ for } \mathbf{w}^\mathsf{T}\mathbf{x}_n + w_0 \leq 0 \text{ and } \mathbf{x}_n \in \text{class with label} +1$$

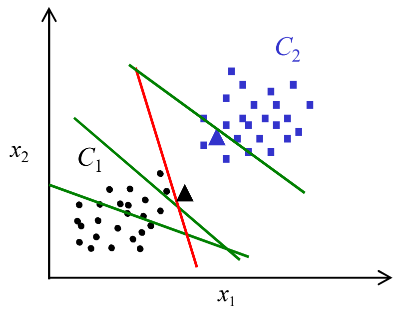
   $$\mathbf{w} = \mathbf{w} - \eta\, \mathbf{x}_n, \text{ for } \mathbf{w}^\mathsf{T}\mathbf{x}_n + w_0 > 0 \text{ and } \mathbf{x}_n \in \text{class with label} -1$$

   - Here $0 < \eta < 1$ is a positive, learning rate parameter
   - Increment the misclassification count by 1
4. Repeat steps 2 and 3 till all the training examples are presented
5. Repeat steps 2 to 4 by setting misclassification count to 0, till the convergence criterion is satisfied

- Convergence criterion:
  - Total misclassification count is 0 **OR**
  - Total misclassification count is minimum (falls below threshold)

Perceptron Learning

- Training:



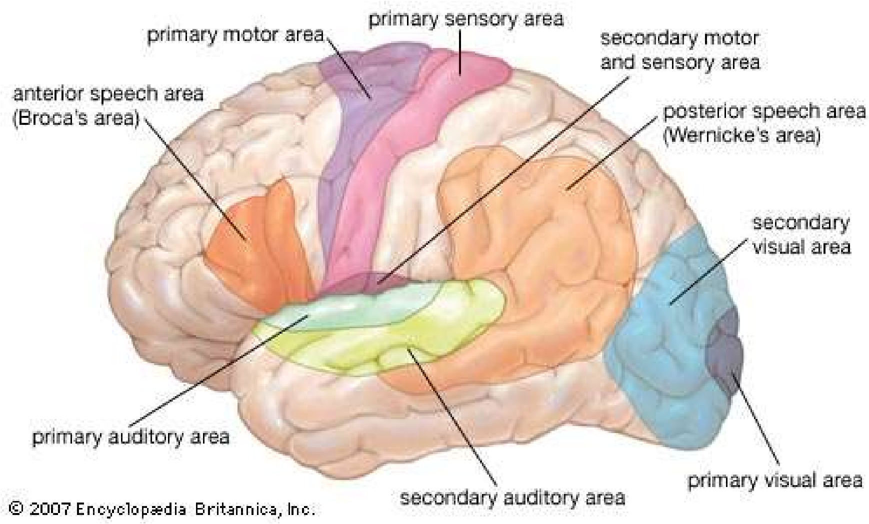
- **Test phase:**
- Classification of a test pattern $\mathbf{x}$ using the weights $\mathbf{w}$ obtained by training the model:
  - If $\mathbf{w}^T\mathbf{x} + w_0 > 0$ then $\mathbf{x}$ is assigned to class with label +1 (C_2)
  - If $\mathbf{w}^T\mathbf{x} + w_0 \leq 0$ then $\mathbf{x}$ is assigned to class with label -1 (C_1)

45

Discriminative Learning Methods for Classification:

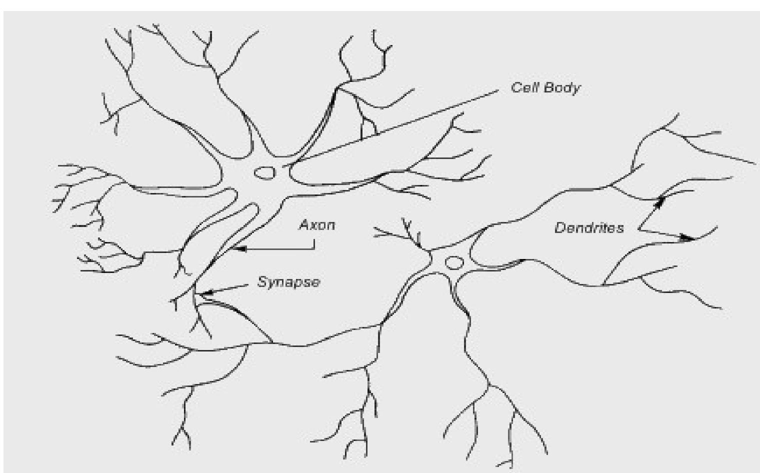
Neural Networks

Human Brain



47

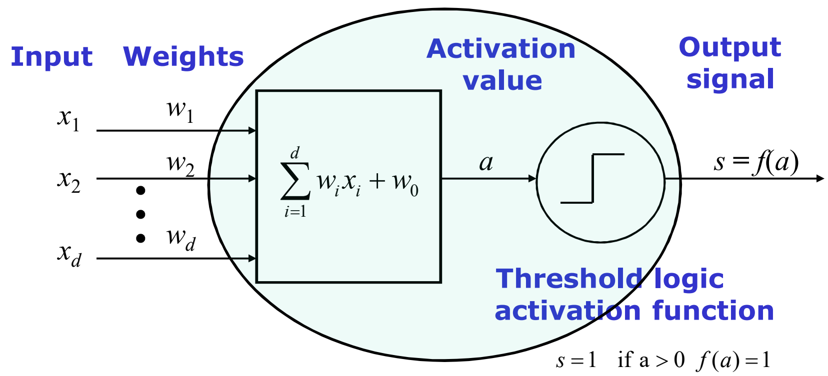
Biological Neural Networks



- Several neurons are connected to one another to form a neural network or a layer of a neural network

48

Neuron with Threshold Logic Activation Function



$s = 1 \quad \text{if } a > 0 \quad f(a) = 1$

$s = 0 \quad \text{if } a < 0 \quad f(a) = 0$

- McCulloch-Pitts Neuron [1]
- Suitable for ***2-class classification problem***

[1] W.S.McCulloch and W.Pitts. A logival calculus of the ideas imminent in nervous activity. 1943.

49

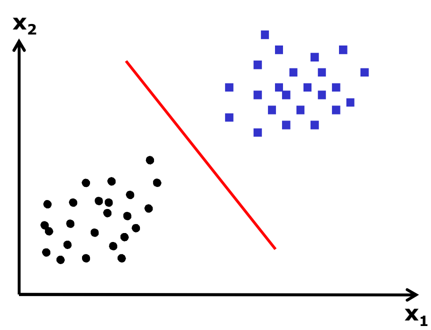
Linearly Separable Classes – Perceptron Model

- Regions of two classes are separable by a linear surface (line, plane or hyperplane)
- Perceptron model that uses a single MuCulloch-Pitts neuron can be trained using the perceptron learning algorithm [2]

Decision surface in a 2-dimensional space is a line:

$$w_1 x_1 + w_2 x_2 + w_0 = 0$$

$$x_2 = -\frac{w_1}{w_2} x_1 - \frac{w_0}{w_2}$$



Decision surface in a d-dimensional space is a hyperplane:

$$\sum_{i=0}^{d} w_i x_i = \mathbf{w}^{\mathrm{T}} \mathbf{x} + w_0 = 0$$

[2] A.G. Ivakhnenko and V.G. Lapa. Cybernetic predicting devices. 1965.

50

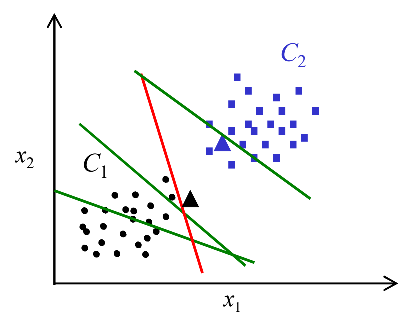
Perceptron Learning – Training Phase

- Given - training data: $\mathcal{D} = \{\mathbf{x}_n, y_n\}_{n=1}^{N}, \ \mathbf{x}_n \in \mathbb{R}^d$ and $y_n \in \{+1, -1\}$
  1. Initialize the $\mathbf{w}$
  2. Choose a training example $\mathbf{x}_n$
  3. Update the $\mathbf{w}$, if $\mathbf{x}_n$ is misclassified
     $\mathbf{w} = \mathbf{w} + \eta\,\mathbf{x}_n, \text{ for } \mathbf{w}^{\mathsf{T}}\mathbf{x}_n + w_0 \le 0 \text{ and } \mathbf{x}_n \in C_2(+1)$
     $\mathbf{w} = \mathbf{w} - \eta\,\mathbf{x}_n, \text{ for } \mathbf{w}^{\mathsf{T}}\mathbf{x}_n + w_0 > 0 \text{ and } \mathbf{x}_n \in C_1(-1)$
     - Here η is a positive, learning rate parameter
     - Increment the misclassification count by 1
  4. Repeat steps 2 and 3 till all the training examples are presented
  5. Repeat steps 2 to 4 by setting misclassification count to 0, till the convergence criterion is not satisfied
- Convergence criterion:
  - Total misclassification count is 0 **OR**
  - Total misclassification count is minimum (falls below threshold)

51

Perceptron Learning

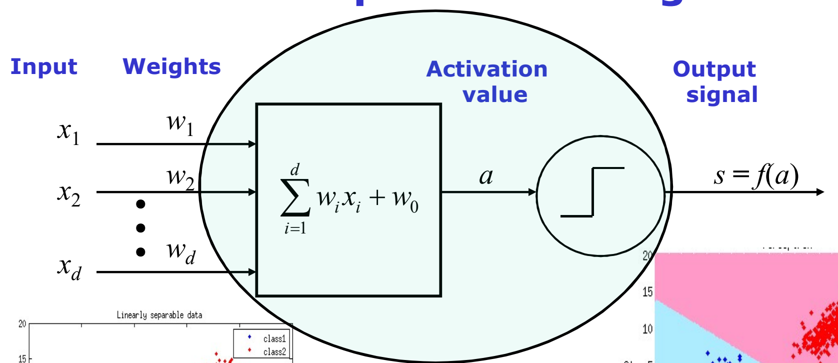
- Training:

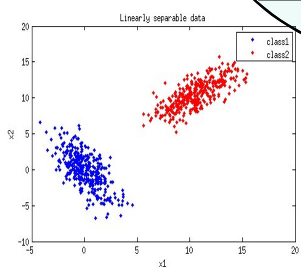


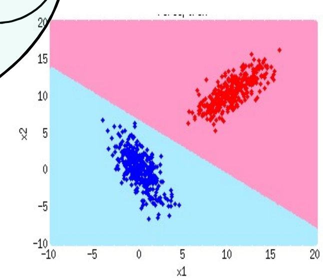
- **Test phase:**
- Classification of a test pattern $\mathbf{x}$ using the weights $\mathbf{w}$ obtained by training the model:
  - If $\mathbf{w}^{\mathsf{T}}\mathbf{x} + w_0 > 0$ then $\mathbf{x}$ is assigned to C_2
  - If $\mathbf{w}^{\mathsf{T}}\mathbf{x} + w_0 \le 0$ then $\mathbf{x}$ is assigned to C_1

52

Neuron with Threshold Logic Activation Function and Perceptron Learning







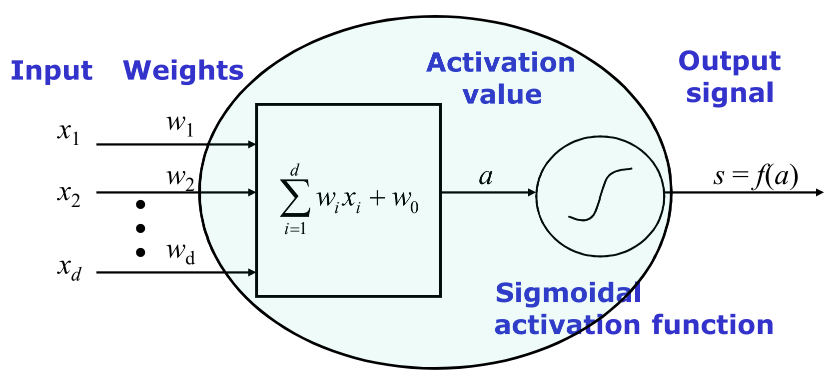
Threshold logic activation function

if a > 0 $f(a) = 1$

if a < 0 $f(a) = 0$

53

Neuron with Continuous Activation Function



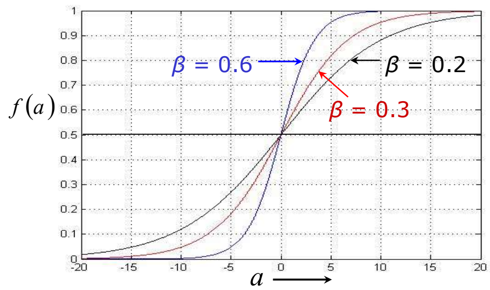
54

Sigmoidal Activation Functions

Logistic function:

$$f(a) = \frac{1}{1+e^{-\beta a}}$$

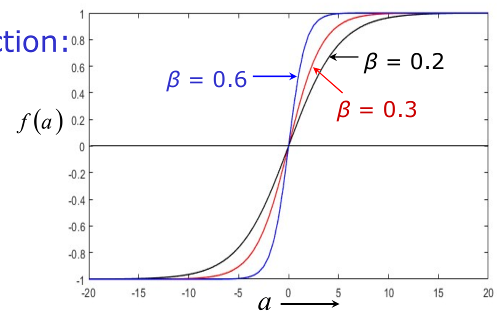
$$\frac{df(a)}{da} = \beta f(a)\left(1 - f(a)\right)$$



Hyperbolic tangent function:

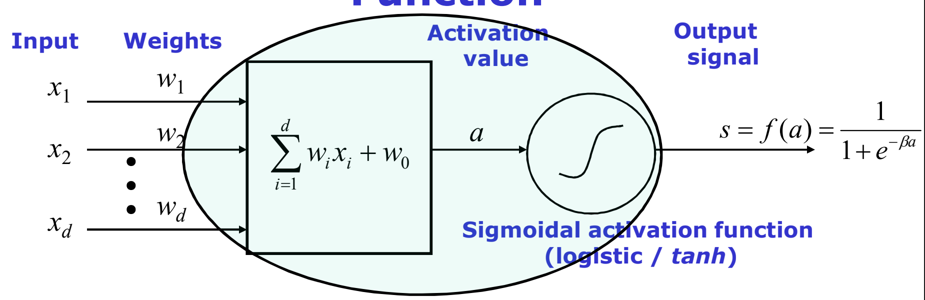
$$f(a) = \tanh(\beta a)$$

$$\frac{df(a)}{da} = \beta\left(1 - f^2(a)\right)$$



5

Neuron with Sigmoidal Activation Function



- Given - training data: $\mathcal{D} = \{\mathbf{x}_n, y_n\}_{n=1}^{N}$, $\mathbf{x}_n \in \mathbb{R}^d$ and
  $y_n \in \{0,1\}$ if logistic activation function, or
  $y_n \in \{+1,-1\}$ if tan hyperbolic activation function
- Instantaneous error for the n^{th} sample is given as,

$$E_n = \frac{1}{2}(y_n - s)^2$$

- Parameter ($\mathbf{w}$) learning is done by minimizing the error using Gradient descent method

56

Neuron with Sigmoidal Activation Function: Parameter Learning – Gradient Descent Method

- Given - training data: $\mathcal{D} = \{\mathbf{x}_n, y_n\}_{n=1}^{N}$, $\mathbf{x}_n \in \mathbb{R}^d$ and $y_n \in \{0,1\}$ or $y_n \in \{+1,-1\}$
  1. Initialize the $\mathbf{w}$ with random values
  2. Choose a training example $\mathbf{x}_n$
  3. Compute output of the neuron:
     - Here $a_n = \mathbf{w}^{\mathsf{T}}\mathbf{x}_n + w_0$

     $s = f(a) = \dfrac{1}{1+e^{-\beta a}}$ **or** $s = f(a) = \tanh(\beta a)$
  4. Compute instantaneous error: $E_n = \dfrac{1}{2}(y_n - s_n)^2$
  5. Change in weight ($\Delta\mathbf{w}$):

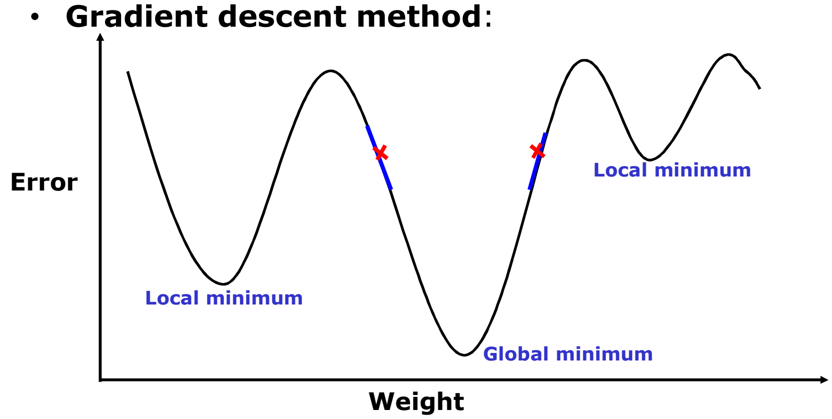
     $$\Delta\mathbf{w} = -\eta \frac{\partial E_n}{\partial \mathbf{w}} \qquad \Delta\mathbf{w} = \eta \delta^o s_n \qquad \text{where } \delta^o = (y_n - s_n)\frac{df(a_n)}{da_n}$$

     where $0 \le \eta \le 1$ is proportionality constant
  6. update the weight ($\mathbf{w}$): $\mathbf{w} = \mathbf{w} + \Delta\mathbf{w}$
  7. Repeat steps 2 and 6 till all the training examples are presented once (Epoch)
  8. Compute the average error: $E_{av} = \dfrac{1}{2N}\sum_{n=1}^{N} E_n$
  9. Repeat steps 2 to 8 till the convergence criterion is satisfied

57

Neuron with Sigmoidal Activation Function: Parameter Learning – Gradient Descent Method

- Convergence criterion:
  - A fixed number of epoch is reached **OR**
  - Difference between average error of successive epochs fall below a threshold (E.g. threshold=10^{-3})
- **Gradient descent method**:



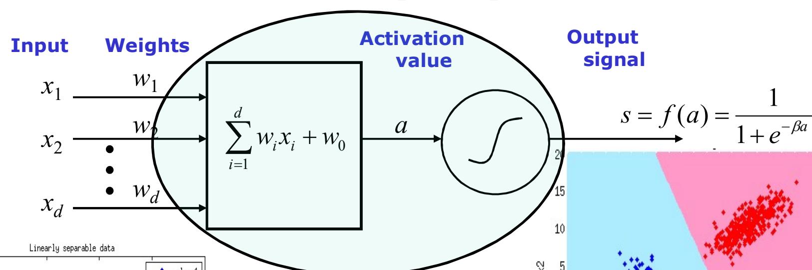
58

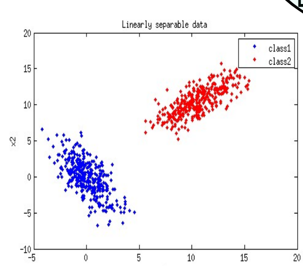
Neuron with Sigmoidal Activation Function: Test Phase

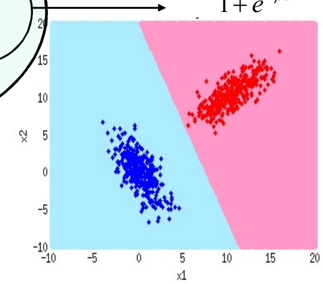
- **Test phase**:
- For a test example $\mathbf{x}$ compute the output of the neuron using the weights $\mathbf{w}$ and w_0 obtained by training the model: $$s = f(a) = \frac{1}{1+e^{-\beta a}} \textbf{ or } s = f(a) = \tanh(\beta a)$$
  - Here $a = \mathbf{w}^\mathsf{T}\mathbf{x}+w_0$
- If $s > 0.5$ (Logistic activation) or $s > 0$ (Tan hyperbolic activation) then $\mathbf{x}$ is assigned to class with label 1 or +1
- If $s \leq 0.5$ (Logistic activation) or $s \leq 0$ (Tan hyperbolic activation) then $\mathbf{x}$ is assigned to class with label 0 or -1

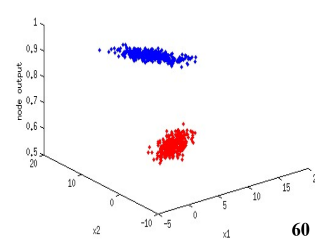
59

Illustration: Linearly Separable Data





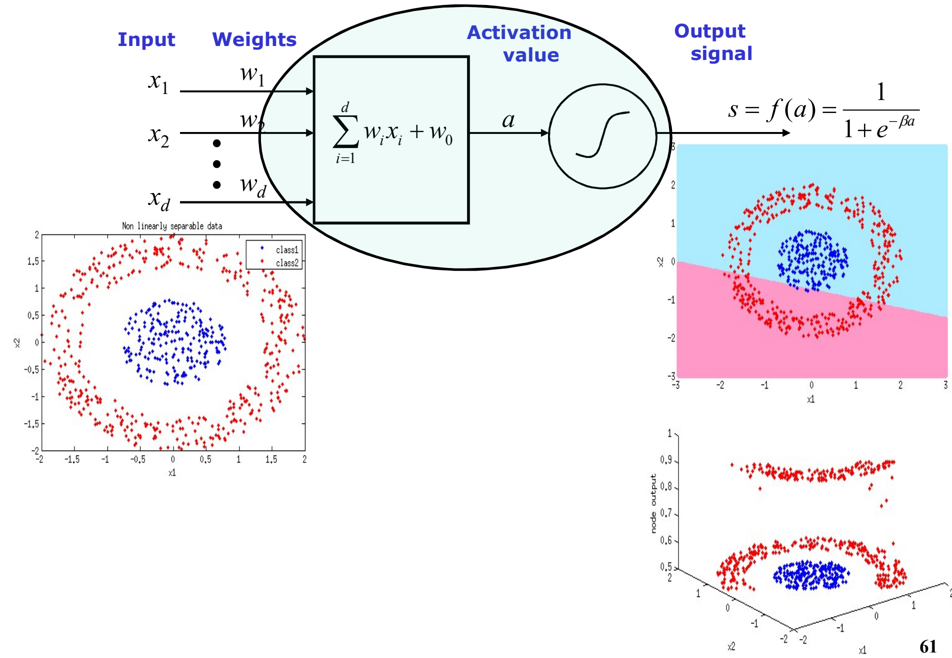




- A single neuron based model using logistic activation function is same as logistic regression

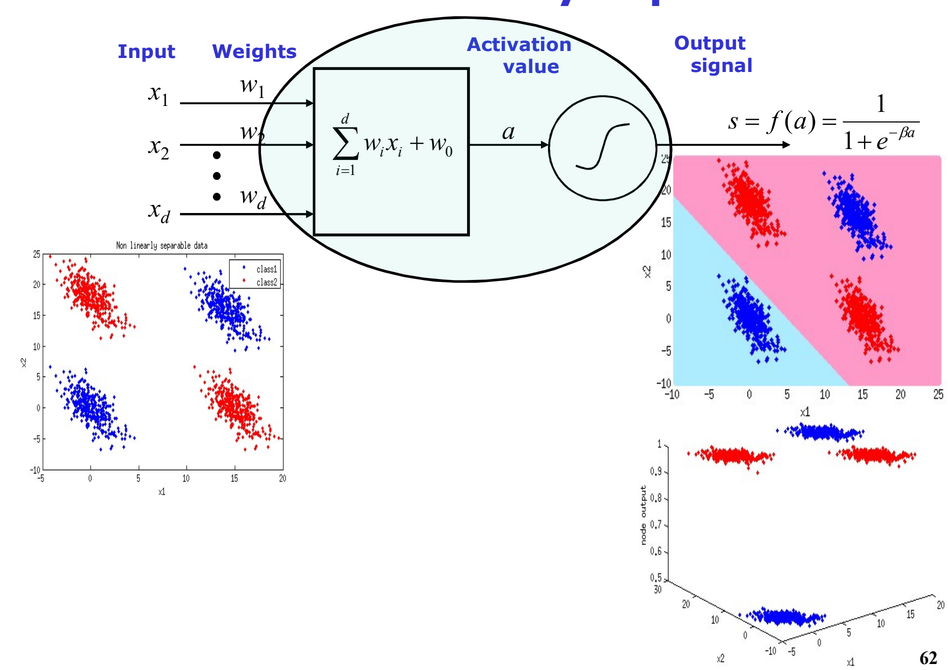
60

Illustration: Non-Linearly Separable Data



61

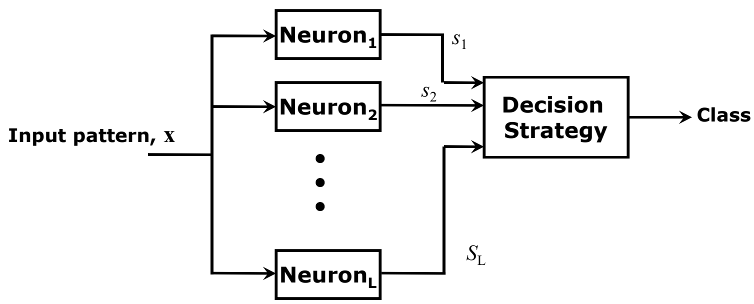
Illustration: Non-Linearly Separable Data



62

Multi-class Pattern Classification using Single Neuron Model

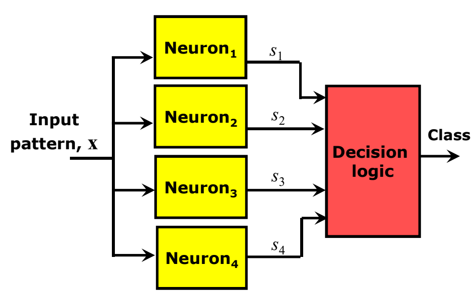
- Multi-class pattern classification for M classes is solved using a combination of several binary (2-class) classifiers and a decision strategy



- Approaches to multi-class pattern classification using single neuron models:
  - One-against-the-rest approach
  - One-against-one approach

63

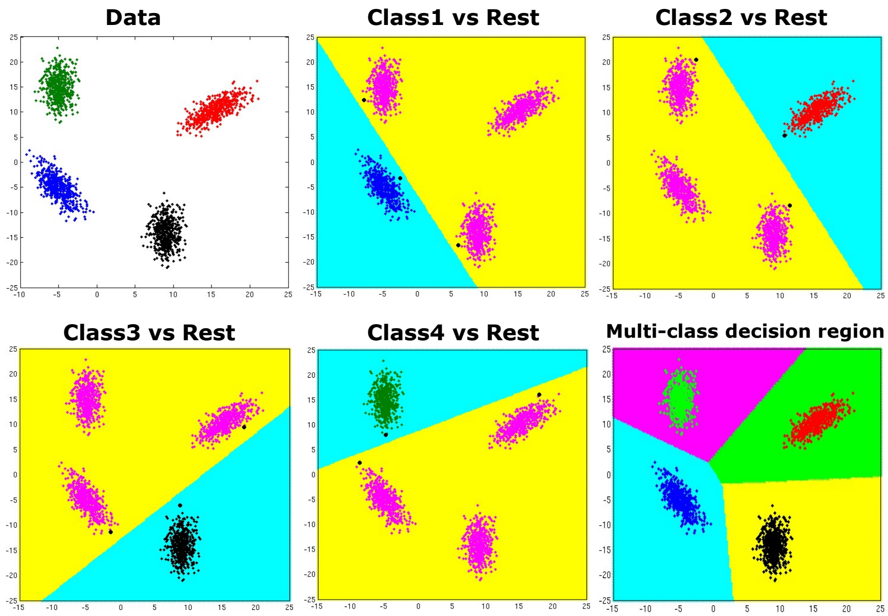
One-against-the-rest Approach



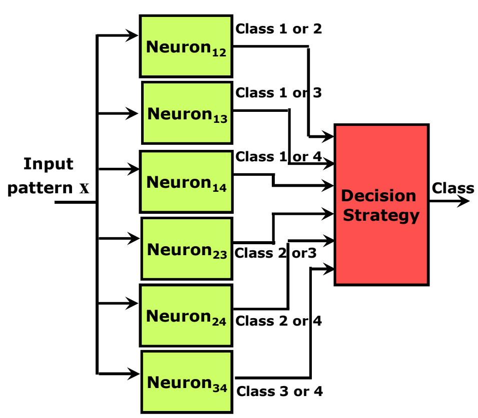
- A single neuron model is built for each class to form a boundary between the region of the class and the regions of the other classes
  - It consider class label 1 or +1 for the examples of a class and 0 or -1 for the examples of rest of the classes
- Let M=4 be the total number of classes. Then, number of single neuron models is L = 4
- A test pattern $\mathbf{x}$ is classified by using winner-takes-all strategy

64

One-against-the-rest Approach: Example



One-against-one Approach

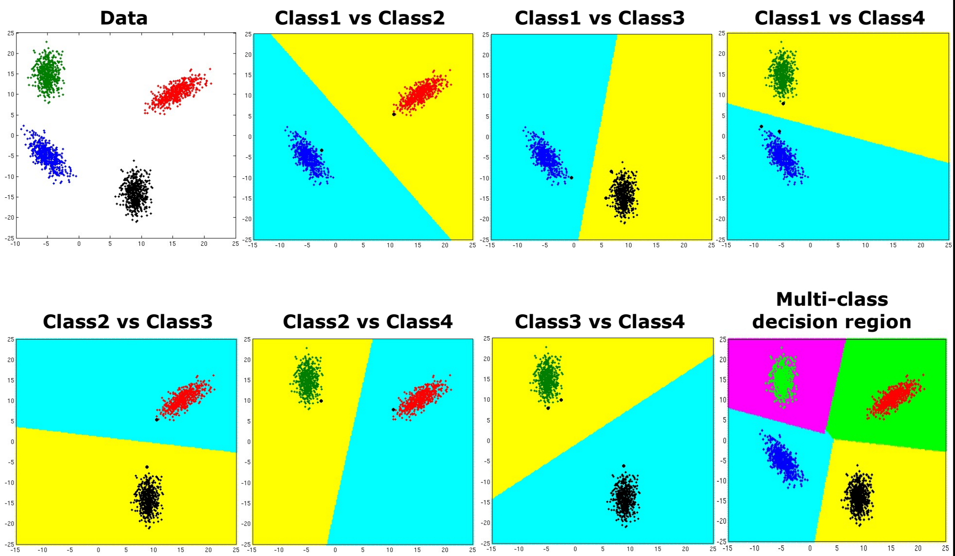


- A single neuron model is built for every pair of classes to form a boundary between their regions
- Let number of classes, M=4 be the total number of classes
- Number of pairwise single neuron models is

  $L=M(M\text{-}1)/2 = 6$
- The maxwins strategy is used to determine the class of a test pattern $\mathbf{x}$
  - In this strategy, a majority voting scheme is used

66

One-against-one Approach: Example

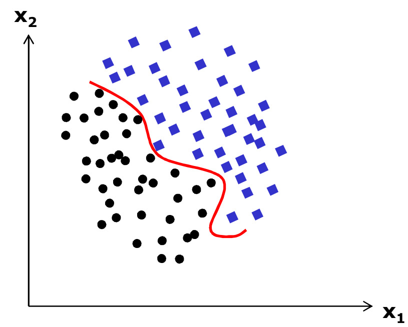


67

Nonlinear Method for Classification

Hard Problems

- Nonlinearly separable classes
- Activation function:
  - Linear (Threshold logic function)
  - Sigmoidal function
  - Gaussian function
  - Rectifier function
  - Spiking function



- Single neuron based classifier is not sufficient
- A network of neurons (neural network) is used that approximates a nonlinear boundary by stitching the boundary from each neuron

69

Artificial Neural Networks

- Learning method:
  - Error correction learning (Backpropagation algorithm [3])
- Structure of network:
  - Feedforward neural networks
    - Fully Connected Neural Network (FCNN),
    - Auto Encoders
    - Convolutional Neural Networks (CNN)
  - Feedback neural networks
    - Recurrent Neural Networks (RNN)
    - Long Short Term Memory (LSTM)
  - Feedforward and feedback neural networks
    - Bidirectional LSTM
    - Self Organizing Maps (SOM)

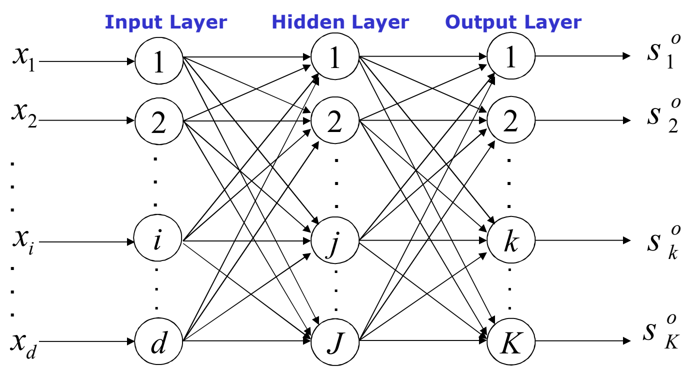
[3] D. E. Rumelhart, G. E. Hinton, and R. J. Williams. Learning internal representations by error propagation. In D. E. Rumelhart and J. L. McClelland, editors, Parallel Distributed Processing, volume 1, pages 318-362. MIT Press, 1986.

70

Neural Networks:
Feedforward Neural Networks
(Fully Connected Network)

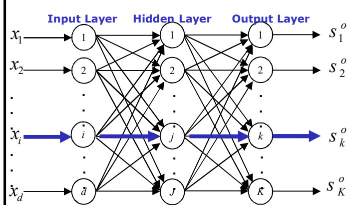
Multilayer Feedforward Neural Network (MLFFNN)

- Architecture of an MLFFNN
  - Input layer: Linear neurons
  - Hidden layers (1 or 2 or more): Sigmoidal neurons
  - Output layer:
    - Linear neurons (for function approximation (regression) task)
    - Sigmoidal neurons (for pattern classification task)



Multilayer Feedforward Neural Network (MLFFNN)

- Architecture of an MLFFNN
  - Input layer: Linear neurons
  - Hidden layers (1 or 2 or more): Sigmoidal neurons
  - Output layer:
    - Linear neurons (for function approximation (regression) task)
    - Sigmoidal neurons (for pattern classification task)

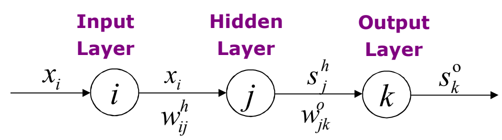


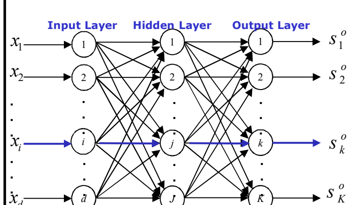
- Number of neurons in input layer (d): Dimension of the data vector
- Number of neurons in the output layer (K): Number of classes in classification or number of output variable in function approximation (regression)

- Number of neurons in the hidden layer is decided experimentally

73

Multilayer Feedforward Neural Network (MLFFNN): Classification





$$s_j^h = f_j^h(a_j^h)$$

$$a_j^h = \sum_{i=1}^{d} w_{ij}^h x_i + w_{0j}^h$$

- $f_j^h(.)$: Activation function of j^{th} hidden neuron

$$s_k^o = f_k^o(a_k^o)$$

$$a_k^o = \sum_{j=1}^{J} w_{jk}^o s_j^h + w_{0k}^o$$

- $f_k^o(.)$: Activation function of k^{th} output neuron

$$s_k^o = f_k^o\left[\sum_{j=1}^{J} w_{jk}^o f_j^h\left(\sum_{i=1}^{d} w_{ji}^h x_i + w_{0j}^h\right) + w_{0k}^o\right]$$

74

Backpropagation Learning [3]

- Gradient descent method
- Error backpropagation algorithm
- Forward computation:
  - Innerproduct computation
  - Activation function computation
- Backward operation:
  - Error calculation and propagation
  - Modification of weights

[3] D. E. Rumelhart, G. E. Hinton, and R. J. Williams. Learning internal representations by error propagation. In D. E. Rumelhart and J. L. McClelland, editors, Parallel Distributed Processing, volume 1, pages 318-362. MIT Press, 1986.

75

Backpropagation Learning

- Given - training data: $\mathcal{D} = \{\mathbf{x}_n, \mathbf{y}_n\}_{n=1}^{N}, \ \mathbf{x}_n \in \mathbb{R}^d$
  - $\mathbf{x}_n$ is n^{th} training example and $\mathbf{y}_n$ is corresponding label vector

$$\mathbf{y}_n = \left[y_{n1}, y_{n2}, \ldots, y_{nk}, \ldots, y_{nK}\right]^{\mathrm{T}}$$

  - If activation function is hyperbolic tangent function:
    - $\mathbf{y}_n$ is a K-dimensional vector with only one element is 1 and rest are -1
  - If activation function is logistic function:
    - $\mathbf{y}_n$ is a K-dimensional binary vector where only one element is 1 and rest are 0
- Instantaneous error for the n^{th} example is given as,

$$E_n = \frac{1}{2}\sum_{k=1}^{K}(y_{nk} - s_k^{\mathrm{o}})^2$$

76

Mode of Presentation of Patterns

- Stochastic Gradient Descent Method
- **Pattern Mode:**
  - At m^{th} epoch: Weights are updated after the presentation of each pattern

$$\Delta w_{jk}(m) = -\eta \frac{\partial E(m)}{\partial w_{jk}} \qquad w_{jk}(m+1) = w_{jk}(m) + \Delta w_{jk}(m)$$

  - Epoch: Presentation of all the patterns once (all training examples)
- **Batch Mode:**
  - Weights are updated after the presentation of all the patterns once – weights are updated using average error

$$\Delta w_{jk}(m) = -\eta \frac{\partial E_{av}}{\partial w_{jk}} \qquad w_{jk}(m+1) = w_{jk}(m) + \Delta w_{jk}(m)$$

where $E_{av} = \frac{1}{2N} \sum_{l=1}^{N} \sum_{k=1}^{K} \left(y_{nk} - s_{nk}^{o} \right)^2$

77

MLFFNN: Parameter Learning (Pattern Mode) – Backpropagation Learning

- Given - training data: $\mathcal{D} = \{\mathbf{x}_n, \mathbf{y}_n\}_{n=1}^{N},\ \mathbf{x}_n \in \mathbb{R}^d$ and $\mathbf{y}_n \in \mathbb{R}^K$
- Architecture:
  - Input layer: d neurons (nodes)
  - One hidden layer: J neurons
  - Output layer: K neurons
- Target: Estimate parameters $\mathbf{W}^h$ and $\mathbf{W}^o$ of MLFFNN
  - $\mathbf{W}^h$ is the weight matrix of size d x J :
    - Indicate the weights w_{ij}^h in the connections between input and hidden layers
    - w_{ij}^h is the weight from i^{th} input neuron to j^{th} neuron in the hidden layer
  - $\mathbf{W}^o$ is the weight matrix of size J x K :
    - Indicate the weights w_{jk}^o in the connections between hidden and output layers
    - w_{jk}^o is the weight from j^{th} hidden neuron to k^{th} neuron in the output layer

78

MLFFNN: Parameter Learning (Pattern Mode) – Backpropagation Learning

- Given - training data: $\mathcal{D} = \{\mathbf{x}_n, \mathbf{y}_n\}_{n=1}^{N}$, $\mathbf{x}_n \in \mathbb{R}^d$ and $y_n \in \mathbb{R}^K$

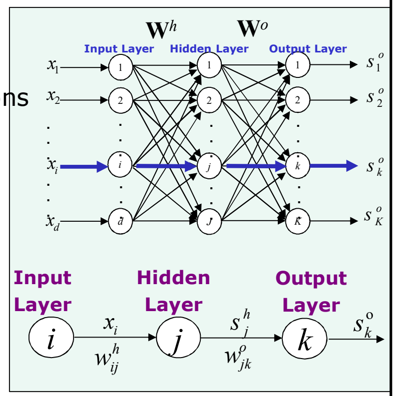
1. Initialize the $\mathbf{W}^h$ and $\mathbf{W}^o$ with random values
2. Choose a training example $\mathbf{x}_n$
3. Forward computation:
   - Compute output of all output neurons (k=1, 2, … K): s_{nk}^o [Refer slide 73]
4. Backward operation:
   - Compute instantaneous error:

$$E_n = \frac{1}{2}\sum_{k=1}^{K}(y_{nk} - s_{nk}^{o})^2$$

   - Update weights between hidden and output layer (j=1, 2, … J and k=1, 2, … K):

$$\Delta w_{jk}^{o} = -\eta \frac{\partial E_n}{\partial w_{jk}^{o}} \qquad w_{jk}^{o} = w_{jk}^{o} + \Delta w_{jk}^{o}$$

   - Update weights between input and hidden layer (i=1, 2, … d and j=1, 2, … J):



$0 \le \eta \le 1$ is proportion ality constant

$$\Delta w_{ij}^{h} = -\eta \frac{\partial E_n}{\partial w_{ij}^{h}} \qquad w_{ij}^{h} = w_{ij}^{h} + \Delta w_{ij}^{h}$$

79

MLFFNN: Parameter Learning (Pattern Mode) – Backpropagation Learning

- Given - training data: $\mathcal{D} = \{\mathbf{x}_n, \mathbf{y}_n\}_{n=1}^{N}$, $\mathbf{x}_n \in \mathbb{R}^d$ and $y_n \in \mathbb{R}^K$

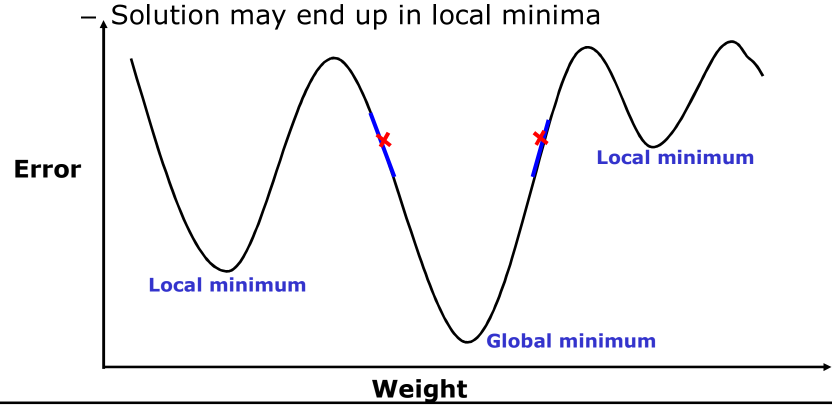
1. Initialize the $\mathbf{W}^h$ and $\mathbf{W}^o$ with random values
2. Choose a training example $\mathbf{x}_n$
3. Forward computation:
   - Compute output of all output neurons (k=1, 2, … K): s_{nk}^o [Refer slide 73]
4. Backward operation:
   - Compute instantaneous error: $E_n = \frac{1}{2}\sum_{k=1}^{K}(y_{nk} - s_{nk}^{o})^2$
   - Update weights between hidden and output layer (j=1, 2, … J and k=1, 2, … K): $\Delta w_{jk}^{o} = -\eta \frac{\partial E_n}{\partial w_{jk}^{o}}$ $\quad w_{jk}^{o} = w_{jk}^{o} + \Delta w_{jk}^{o}$
   - Update weights between input and hidden layer (i=1, 2, … d and j=1, 2, … J): $\Delta w_{ij}^{h} = -\eta \frac{\partial E_n}{\partial w_{ij}^{h}}$ $\quad w_{ij}^{h} = w_{ij}^{h} + \Delta w_{ij}^{h}$

   $0 \le \eta \le 1$ is proportionality constant
5. Repeat steps 2 and 4 till all the training examples are presented once (Epoch)
6. Compute the average error: $E_{av} = \frac{1}{2N}\sum_{n=1}^{N} E_n$
7. Repeat steps 2 to 6 till the convergence criterion is satisfied

80

MLFFNN: Parameter Learning (Pattern Mode) – Backpropagation Learning

- Convergence criterion:
  - A fixed number of epoch is reached **OR**
  - Difference between average error of successive epochs fall below a threshold (E.g. threshold=10^{-3})
- **Gradient descent method**:
  - Solution may end up in local minima



81

MLFFNN: Test Phase

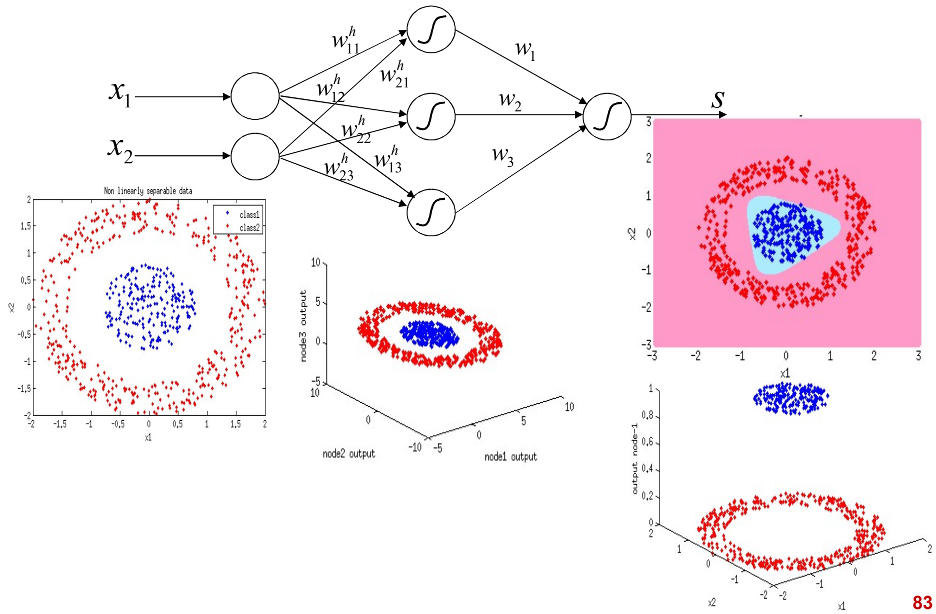
- **Test phase**:
- For a test example $\mathbf{x}$ compute the output of each of the neurons in output layer (s_k^o , k=1,2,... K) using the weights obtained by training the model:

Class label for $\mathbf{x}$ = $\arg\max_k \; s_k^o \qquad k = 1, 2, ..., K$

82

Illustration: Ring Data

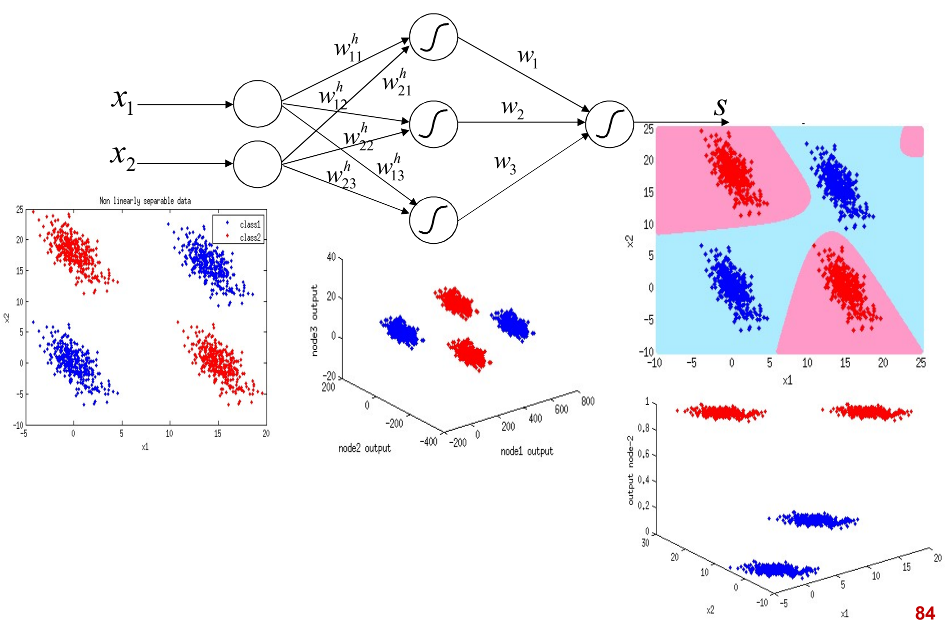
Input Layer **Hidden Layer** **Output Layer**



83

Illustration: X-OR Data

Input Layer **Hidden Layer** **Output Layer**



84

Practical Considerations

- Stopping Criterion:
  - Threshold on average error
  - Threshold on average gradient
- Number of Weights:
  - Depends on number of input nodes, output nodes, hidden nodes and hidden layers
- Number of Hidden Nodes
  - Cross-validation method – Experimentally determined
- Data Requirements
  - Large when the number of weights are large
- Limitations:
  - Slow convergence
  - Local minima problem

85

Neural Networks: Feedforward Neural Networks
(Fully Connected Network)

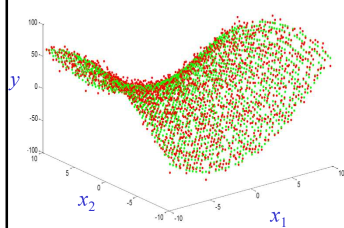
Regression

86

Neural Networks for Regression (Function Approximation)

- Given:- Training data: $\mathcal{D} = \{\mathbf{x}_n, y_n\}_{n=1}^{N}$, $\mathbf{x}_n \in \mathbb{R}^d$ and $y_n \in \mathbb{R}^1$
- Function governing the relationship between input and output governed by some function $f(.)$:

$$y_n = f(\mathbf{x}_n, \mathbf{W})$$



$y = f(\mathbf{x}_n, \mathbf{W})$

$\mathbf{x} = [x_1, x_2]^{\mathrm{T}}$

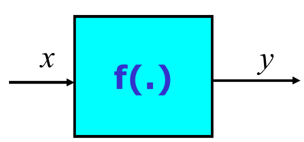
Fitting a surface

- The coefficient matrix $\mathbf{W}$ are the parameters of curve or surface (regression coefficients) ***- Unknown***
- Neural networks approximates the function $f(\mathbf{x}_n, \mathbf{W})$ and is a nonlinear function of coefficients $\mathbf{W}$
  - ***Nonlinear model for regression***

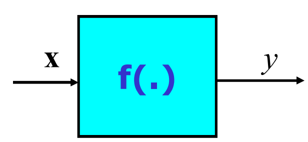
87

Regression using Multilayer Feedforward Neural Network (MLFFNN)

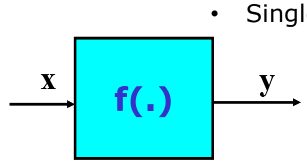
- Neural network learns (approximates) the complex underlying function between a dependent variable (one or more) and one or more independent variables



- Single independent variable (x)
- Single dependent variable (y)



- Multiple independent variable ($\mathbf{x} \in \mathbb{R}^d$)
- Single dependent variable (y)

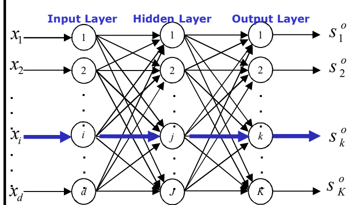


- Multiple independent variable ($\mathbf{x} \in \mathbb{R}^d$)
- Multiple dependent variable ($\mathbf{y} \in \mathbb{R}^K$)

88

Regression using MLFFNN

- Architecture of an MLFFNN
  - Input layer: Linear neurons
  - Hidden layers (1 or 2 or more): Sigmoidal neurons
  - Output layer:
    - Linear neurons (if the dependent variable is not normalized)
    - Sigmoidal neurons (if the dependent variable is normalized)



- Number of neurons in input layer (d): Number of independent variables
- Number of neurons in the output layer (K): Number of dependent variables

- Number of hidden layers and neurons in the hidden layer are decided experimentally

89

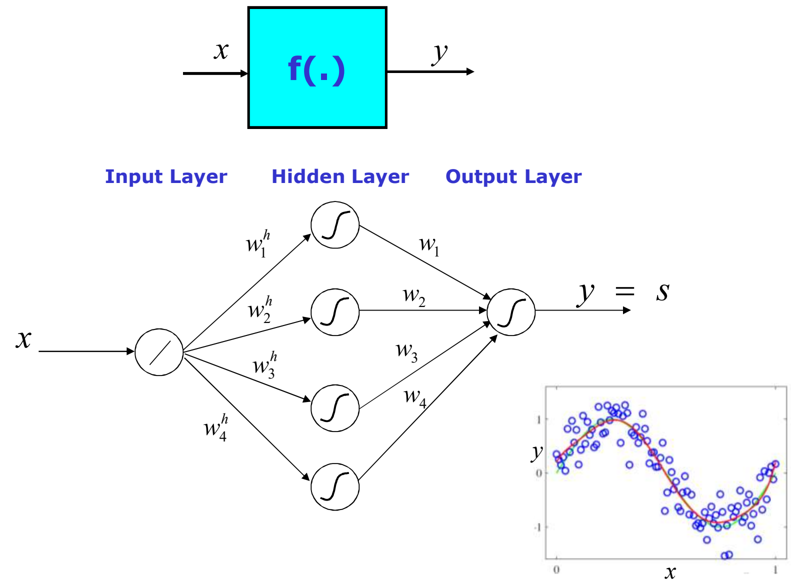
Backpropagation Learning [3]

- Gradient descent method
- Error backpropagation algorithm
- Forward computation:
  - Innerproduct computation
  - Activation function computation
- Backward operation:
  - Error calculation and propagation
  - Modification of weights

[3] D. E. Rumelhart, G. E. Hinton, and R. J. Williams. Learning internal representations by error propagation. In D. E. Rumelhart and J. L. McClelland, editors, Parallel Distributed Processing, volume 1, pages 318{362. MIT Press, 1986.

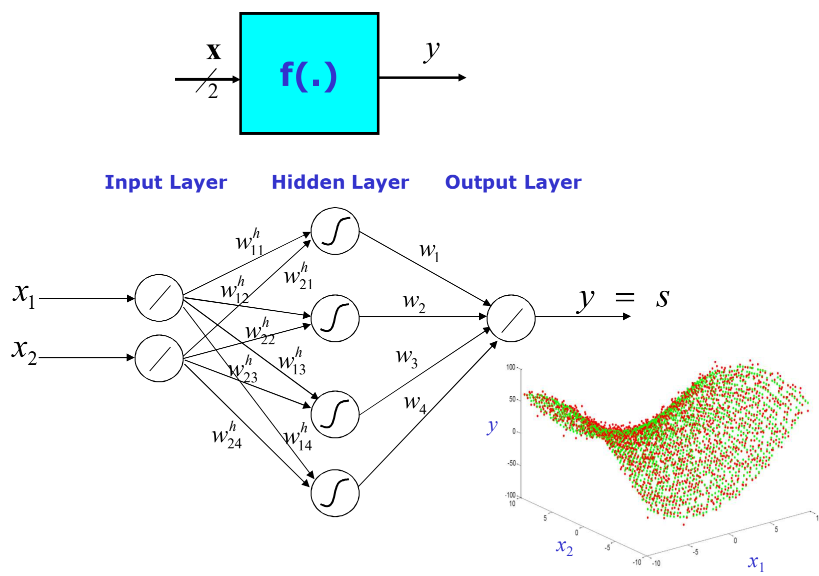
90

Illustration: Single Independent and Dependent Variable



91

Illustration: Two Independent and Single Dependent Variable



92

Feedforward Neural Networks: Summary

- Perceptrons, with threshold logic function as activation function, are suitable for pattern classification tasks that involve linearly separable classes
- Multilayer feedforward neural networks, with sigmoidal function as activation function, are suitable for nonlinearly separable classes
  - Complexity of the model depends on
    - Dimension of the input pattern vector
    - Number of classes
    - Shapes of the decision surfaces to be formed
  - Architecture of the model is empirically determined
  - Large number of training examples are required when the complexity of the model is high
  - Local minima problem
- Multilayer feedforward neural network models are suitable for regression (function approximation) tasks also
- Multilayer feedforward neural network with one or two hidden layers is now called a shallow network

93

Text Books

1. J. Han and M. Kamber, *Data Mining: Concepts and Techniques*, Third Edition, Morgan Kaufmann Publishers, 2011.
2. S. Theodoridis and K. Koutroumbas, *Pattern Recognition*, Academic Press, 2009.
3. C. M. Bishop, *Pattern Recognition and Machine Learning*, Springer, 2006.
4. B. Yegnanarayana, Artificial Neural Networks, Prentice-Hall of India, 1999.
5. Satish Kumar, Neural Networks - A Class Room Approach, Second Edition, Tata McGraw-Hill, 2013.
6. S. Haykin, Neural Networks and Learning Machines, Prentice Hall of India, 2010.

94