THE UNIVERSITY OF RHODE ISLAND

NeuralPC Lab

Brain Signal processing & Applications

BME 473/ELE 573

Lecture 6

Dr. Yalda Shahriari

Introduction to Machine Learning

- Machine Learning (ML) is a branch of artificial intelligence (AI) that gives computers the ability to learn without explicitly being programmed.
- Unlike traditional programming, where humans explicitly define rules, ML algorithms learn patterns from data and make decisions or predictions.
- Methods or techniques used to learn from data. They can be simple linear models or complex neural networks.
- Learning Process:
  - **Training:** Feeding data into an algorithm to help it learn.
  - **Testing:** Evaluating the performance of the model on unseen data.

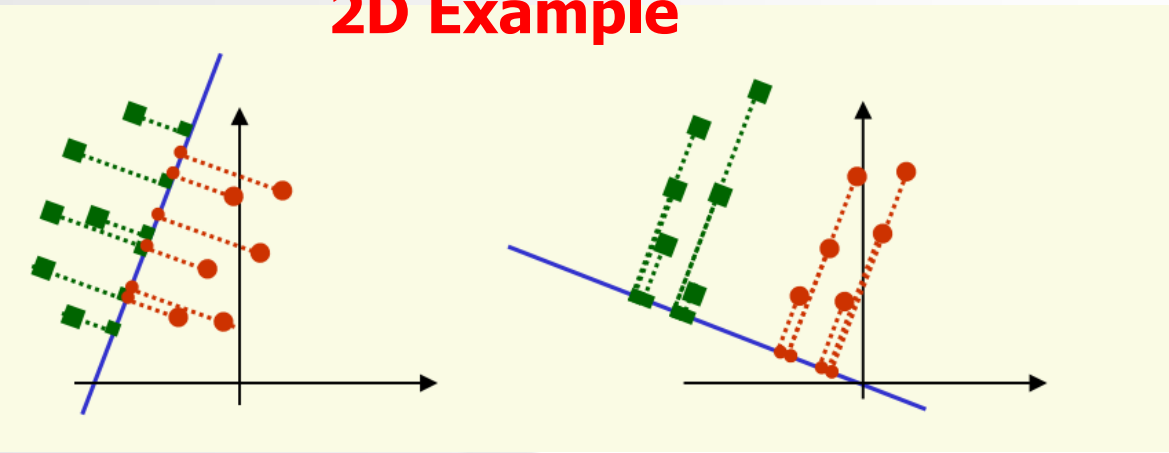
Introduction to Machine Learning

- **Types of Machine Learning**
- ✓ Supervised Learning: The algorithm learns from labeled training data to make predictions or decisions.
- ✓ Unsupervised Learning: The algorithm learns patterns and structures in data without labeled outputs.
- ✓ Semi-supervised Learning: The algorithm learns patterns and structures in small amount of data labeled outputs and large data without labels.
- ✓ Reinforcement Learning: A type of machine learning where an agent learns to make decisions by interacting with an environment and receiving rewards or punishments.

Linear Discriminant Analysis (LDA)

- **Objective:**
- ✓ Linear discriminant analysis (LDA) seeks to reduce the dimensionality while causing as much class discrimination as possible.
- ✓ Main idea: find a subspace that the projected samples into the new subspace become well-separated.

2D Example

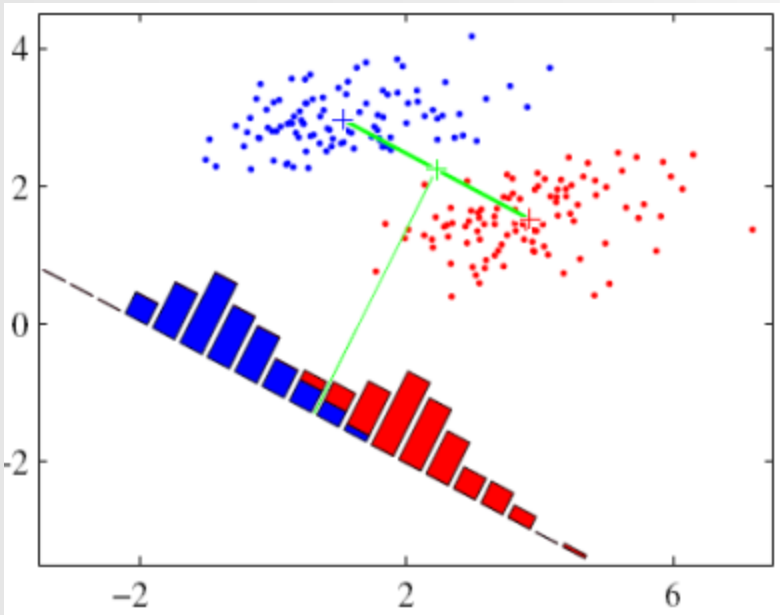


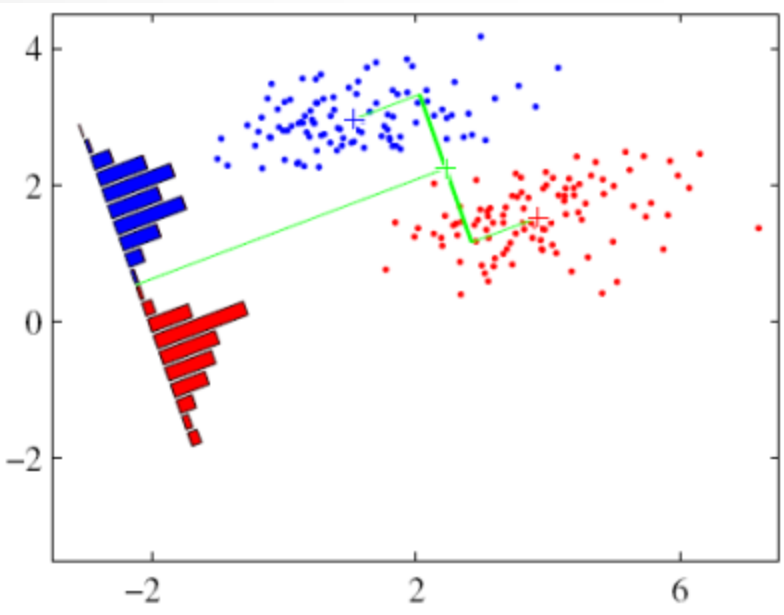
Bad line to project to,
Classes are mixed up

Good line to project to,
Classes are well separated

Linear Discriminant Analysis (LDA)

- **Another example:**





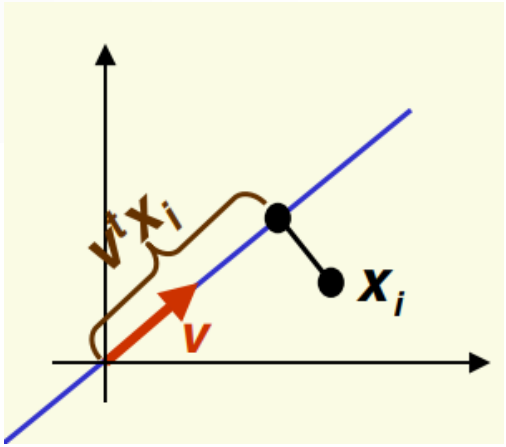
$$y = w^T x$$

Linear Discriminant Analysis (LDA)

- **Objective:**
- ✓ Assume we have a set of D-dimensional samples $\{x_1, x_2, \ldots, x_N\}$, where $N1$ samples belong to class $\omega 1$, and $N2$ samples to class $\omega 2$.
- ✓ We seek to obtain a scalar y by projecting the samples x onto a line w as below:

$$y = w^T x$$

- ✓ Of all the possible lines we would like to select the one that maximizes the separability of the scalars from different classes.
- ✓ **Reminder:** Let the line direction be given by vector v, thus, $v^T x$ is the projection of x_i into a one dimensional subspace v:



Linear Discriminant Analysis (LDA)

- **What is a good measure of seperability?**

- ✓ One possible solution: separating the means of the projected data.
- ✓ Let μ_1 and μ_2 be the means of class 1 and 2, and $\tilde{\mu}_1$ *and* $\tilde{\mu}_2$ be the means of the projected classes. Thus we have:

$$\mu_i = \frac{1}{N_i}\sum_{x\in\omega_i} x \text{ and } \tilde{\mu}_i = \frac{1}{N_i}\sum_{y\in\omega_i} y = \frac{1}{N_i}\sum_{x\in\omega_i} w^T x = w^T \mu_i$$

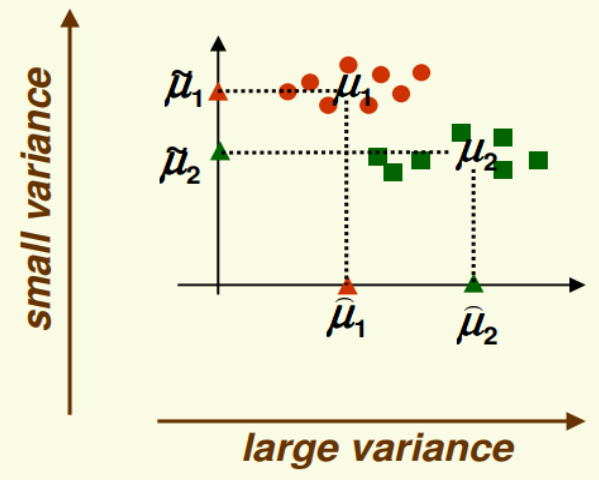
The mean vector of each class in x –space and y –space

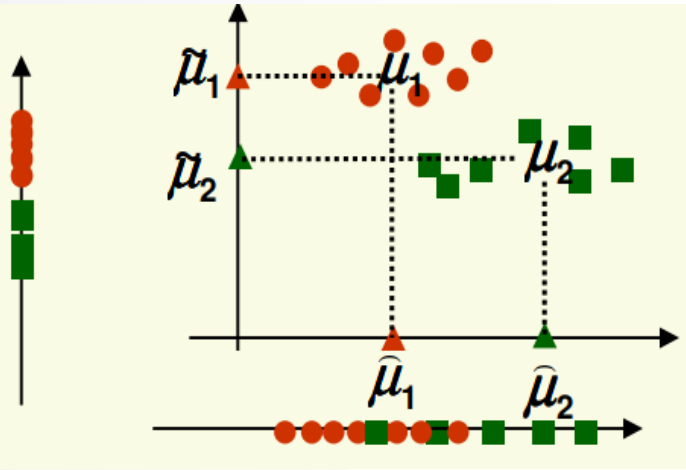
- ✓ Thus, the difference between the projected mean vectors between two classes are:

$$J(w) = |\tilde{\mu}_1 - \tilde{\mu}_2| = |w^T(\mu_1 - \mu_2)|$$

Linear Discriminant Analysis (LDA)

- **How good is $|\tilde{\mu}_1 - \tilde{\mu}_2|$ as a measure of separation?**
  - ✓ The larger $|\tilde{\mu}_1 - \tilde{\mu}_2|$, the better is the expected separation
- ✓ But this is not a sufficient measure since it does not account for the standard deviation within classes.





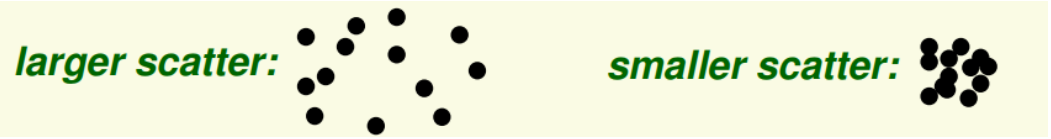
- ✓ Thus, the vertical axes is a better line than the horizontal axes to project for class separability although, $|\widehat{\mu_1} - \widehat{\mu_2}| > |\tilde{\mu}_1 - \tilde{\mu}_1|$.

Linear Discriminant Analysis (LDA)

- **Fisher's solution:**
- ✓ Fisher suggested maximizing the difference between the means but minimizing the within class variances.
- ✓ This will be achieved by normalizing the $|\tilde{\mu}_1 - \tilde{\mu}_2|$ by a factor of variance.
- ✓ Let's define: $s^2 = \sum_{i=1}^{n}(x_i - \mu_x)^2$ where s is the scatter (sample variance multiplied by n) and x_i are the samples with sample mean of $\mu_x = \frac{1}{n}\sum_{i=1}^{n} x_i$.
- ✓ Note: scatter measures the same thing as variance, the spread of data around the mean, but has different scale
- ✓ Scatter for projected samples from two classes:

$$\tilde{s}_1^{\,2} = \sum_{y_i \in Class1}(y_i - \tilde{\mu}_1)^2 \quad \& \quad \tilde{s}_2^{\,2} = \sum_{y_i \in Class2}(y_i - \tilde{\mu}_2)^2$$

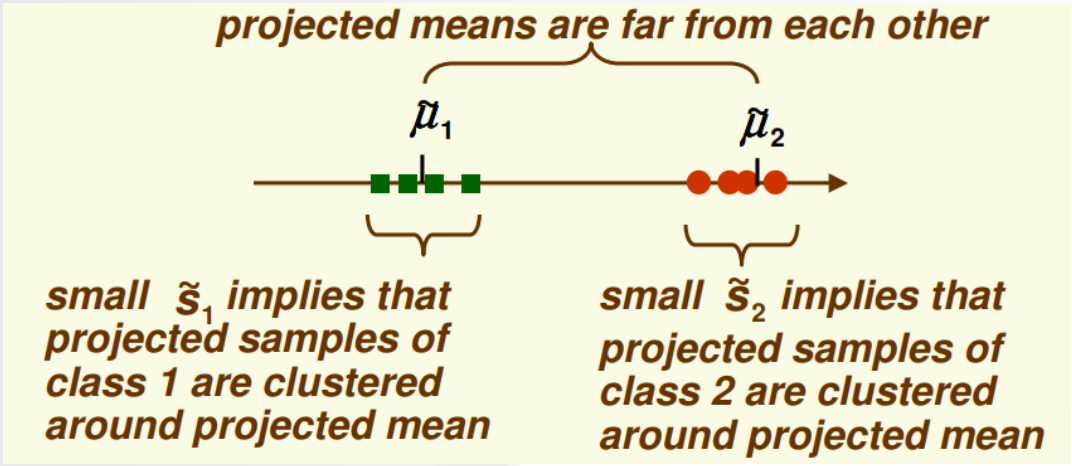
Example:



Linear Discriminant Analysis (LDA)

- ✓ After normalizing by the scatter of both classes we have the following objective function which we want to maximize it:

$$J(w) = \frac{|\tilde{\mu}_1 - \tilde{\mu}_2|^2}{\tilde{s}_1^2 + \tilde{s}_2^2}$$



Linear Discriminant Analysis (LDA)

- ✓ First, we define a measure of the scatter in feature space x

$$S_i = \sum_{x \in \omega_i} (x - \mu_i)(x - \mu_i)^T$$
$$S_1 + S_2 = S_W$$

- ✓ Where S_W is called the within-class scatter matrix
- ✓ Knowing that by projecting data we have: $y_i = w^T x_i$ and $\tilde{\mu}_1 = w^T \mu_1$ then the scatter of the projected y can then be expressed as below:

$$\tilde{s}_i^2 = \sum_{y \in \omega_i} (y - \tilde{\mu}_i)^2 = \sum_{x \in \omega_i} (w^T x - w^T \mu_i)^2 =$$
$$= \sum_{x \in \omega_i} w^T (x - \mu_i)(x - \mu_i)^T w = w^T S_i w$$

$$\tilde{s}_1^2 + \tilde{s}_2^2 = w^T S_W w$$ ← Projected within class scatter matrix

- ✓ Similarly the difference between the projected means can be expressed as below:

$$(\tilde{\mu}_1 - \tilde{\mu}_2)^2 = (w^T \mu_1 - w^T \mu_2)^2 = w^T \underbrace{(\mu_1 - \mu_2)(\mu_1 - \mu_2)^T}_{S_B} w = w^T S_B w$$

Where S_B is called the between-class scatter.

Linear Discriminant Analysis (LDA)

- ✓ Finally we can express the Fisher criteria in terms of S_W and S_B as below:

$$J(w) = \frac{w^T S_B w}{w^T S_W w}$$

Objective function

- ✓ Thus, to maximize the objective function we have:

$$\frac{d}{dw}[J(w)] = \frac{d}{dw}\left[\frac{w^T S_B w}{w^T S_W w}\right] = 0 \Rightarrow$$
$$[w^T S_W w]\frac{d[w^T S_B w]}{dw} - [w^T S_B w]\frac{d[w^T S_W w]}{dw} = 0 \Rightarrow$$
$$[w^T S_W w]2S_B w - [w^T S_B w]2S_W w = 0$$

Dividing by $w^T S_W w$ we have:

$$\left[\frac{w^T S_W w}{w^T S_W w}\right] S_B w - \left[\frac{w^T S_B w}{w^T S_W w}\right] S_W w = 0 \Rightarrow$$
$$S_B w - J S_W w = 0 \Rightarrow$$
$$S_W^{-1} S_B w - J w = 0$$

$$\Rightarrow \quad S_W^{-1} S_B w = J w$$

Generalized eigenvalue problem

Linear Discriminant Analysis (LDA)

- ✓ Solving the generalized eigenvalue problem we will have:

$$S_W^{-1}\,(\mu_1 - \mu_2) = w^*$$

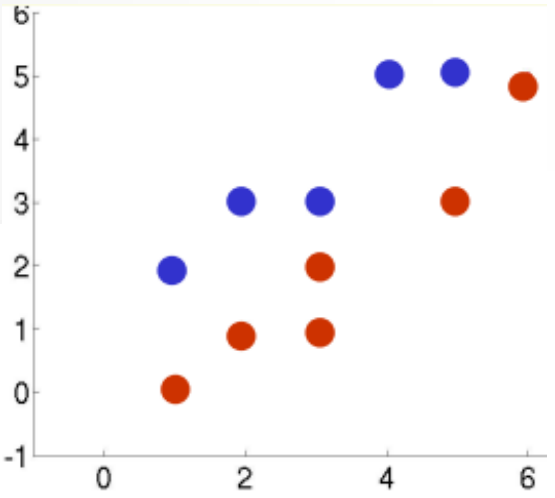
This is known as Fisher's linear discriminant (1936). Thus the projected data would be: $y = w^{*T} x$

- ➢ **Example:**

Compute the FLDA projection for the following 2D dataset:

$c1 = [(1,2), (2,3), (3,3), (4,5), (5,5)]$

$c2 = [(1,0), (2,1), (3,1), (3,2), (5,3), (6,5)]$



Linear Discriminant Analysis (LDA)

❑ Solution by hand:

✓ The class statistics are:

$$\mu_1 = mean(c1) = [3 \quad 3.6] \qquad \mu_2 = mean(c2) = [3.3 \quad 2.2]$$

$$S_1 = \begin{bmatrix} 10 & 8 \\ 8 & 7.2 \end{bmatrix} \qquad S_2 = \begin{bmatrix} 17.3 & 16 \\ 16 & 16 \end{bmatrix}$$

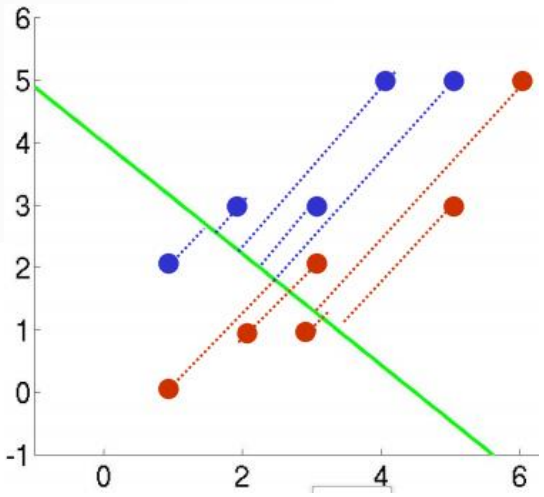
✓ Within class scatter:

$$S_W = S_1 + S_2 = \begin{bmatrix} 27.3 & 24 \\ 24 & 23.2 \end{bmatrix}$$

$$S_W^{-1} = \text{inv}(S_W) = \begin{bmatrix} 0.39 & -0.41 \\ -0.41 & 0.47 \end{bmatrix}$$

✓ Finally the optimal line direction W is:

$$S_W^{-1}(\mu_1 - \mu_2) = \begin{bmatrix} -0.79 \\ 0.89 \end{bmatrix}$$



✓ **Note: as long as the line has the right direction, its exact position does not matter**

Linear Discriminant Analysis (LDA)

➢ Multiclass problem:

✓ FLDA generalizes for C-class as well. In this case instead of one projection vector W we will seek for (C-1) projections $[w_1, w_2, \ldots, w_{C-1}]$.

$$y = w^T x$$

✓ Now we need to generalize the within-class and between-class scatter.

✓ The generalization of within-class scatter is as below:

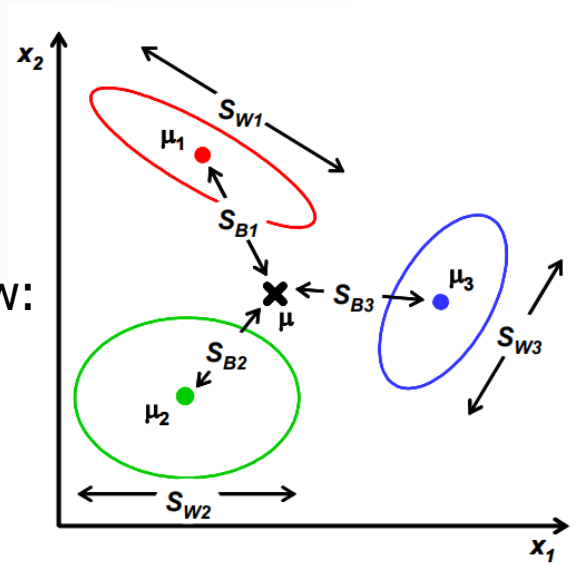
$$S_W = \sum_{i=1}^{C} S_i$$

$$\text{where } S_i = \sum_{x \in \omega_i} (x - \mu_i)(x - \mu_i)^T \text{ and } \mu_i = \frac{1}{N_i} \sum_{x \in \omega_i} x$$

✓ The generalization of between-class scatter is as below:

$$S_B = \sum_{i=1}^{C} N_i (\mu_i - \mu)(\mu_i - \mu)^T$$

$$\text{where } \mu = \frac{1}{N} \sum_{\forall x} x = \frac{1}{N} \sum_{x \in \omega_i} N_i \mu_i$$



Linear Discriminant Analysis (LDA)

- ✓ Similarly, we define the projected mean vector and scatter matrices as below:

$$\tilde{\mu}_i = \frac{1}{N_i}\sum_{y\in\omega_i} y \qquad \tilde{S}_W = \sum_{i=1}^{C}\sum_{y\in\omega_i}(y-\tilde{\mu}_i)(y-\tilde{\mu}_i)^T$$

$$\tilde{\mu} = \frac{1}{N}\sum_{\forall y} y \qquad \tilde{S}_B = \sum_{i=1}^{C} N_i(\tilde{\mu}_i-\tilde{\mu})(\tilde{\mu}_i-\tilde{\mu})^T$$

- ✓ We also will have:

$$\tilde{S}_W = W^T S_W W$$
$$\tilde{S}_B = W^T S_B W$$

- ✓ And finally we have the following objective function which goes to generalized eigenvalue problem.

$$J(W) = \frac{\left|\tilde{S}_B\right|}{\left|\tilde{S}_W\right|} = \frac{\left|W^T S_B W\right|}{\left|W^T S_W W\right|}$$

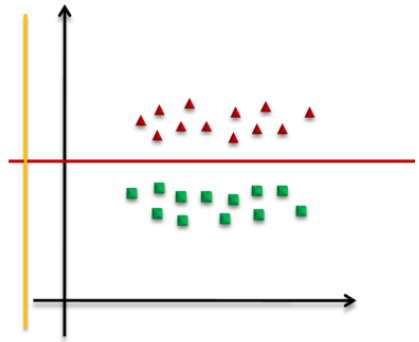
$$W^* = \left[w_1^* \mid w_2^* \mid \cdots \mid w_{C-1}^*\right] = \text{argmax}\left\{\frac{\left|W^T S_B W\right|}{\left|W^T S_W W\right|}\right\} \Rightarrow (S_B - \lambda_i S_W)w_i^* = 0$$

Linear Discriminant Analysis (LDA)

➢ PCA vs. FLDA:

- ✓ PCA perform dimensionality reduction while preserving as much of the variance in the high dimensional space as possible.
- ✓ LDA perform dimensionality reduction while preserving as much of the class discriminatory information as possible.

✓ Examples:

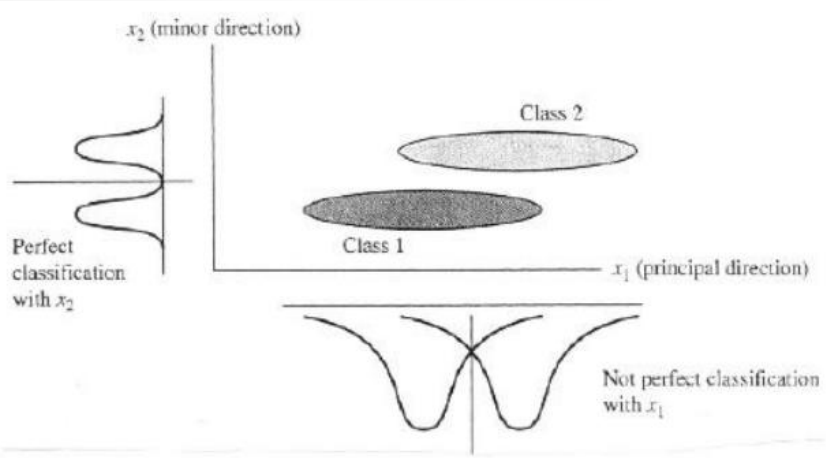


PCA

- higher variance
- bad for discriminability

LDA

- smaller variance
- good discriminability



Bayesian Classifier

- **Bayesian classifier:**
  - ✓ Bayesian theorem plays a critical role in probabilistic learning and classification.
  - ✓ It uses prior probability of each class given no information about an item.
  - ✓ Classification gives a posterior probability given a description of an item

- **Reminder:**
  - ✓ **Product rule:**

$$P(A \wedge B) = P(A \mid B)P(B) = P(B \mid A)P(A)$$

  - ✓ **Sum rule:**

$$P(A \vee B) = P(A) + P(B) - P(A \wedge B)$$

Bayesian Classifier

- **Bayesian theorem:**

$$P(c \mid X) = \frac{P(X \mid c)P(c)}{P(X)} \quad \Longrightarrow \quad Posterior = \frac{Likelihood \times Prior}{Evidence}$$

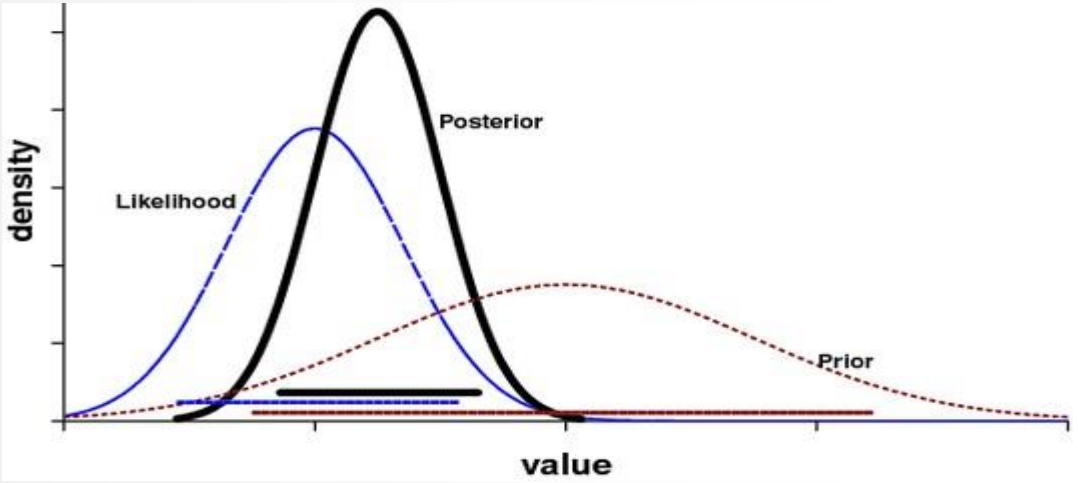
- ✓ Prior probability needs to be determined prior to observing the data.
- ✓ Likelihood probability is being constructed based on the observed data.
- ✓ Evidence can be built based on the likelihood functions:

$$p(X) = \sum_{i=1}^{n} p(X_i|c)p(c)$$

Bayesian Classifier

- ✓ Posterior probability is proportional to the multiplication of likelihood function and prior distribution.

Example distribution illustrations for prior, posterior and likelihood functions.



- ➢ **Maximum A Posterior (MAP)**
- ✓ Assign x to c^* if:

$$P(C = c^* \mid \mathbf{X} = \mathbf{x}) > P(C = c \mid \mathbf{X} = \mathbf{x}) \quad c \neq c^*,\ c = c_1, \cdots, c_L$$

Bayesian Classifier

- **Methods of Bayesian classification with MAP rule:**

1.

$$P(C = c_i \mid \mathbf{X} = \mathbf{x}) = \frac{P(\mathbf{X} = \mathbf{x} \mid C = c_i) P(C = c_i)}{P(\mathbf{X} = \mathbf{x})}$$

$$\propto P(\mathbf{X} = \mathbf{x} \mid C = c_i) P(C = c_i)$$

$$\text{for } i = 1, 2, \cdots, L$$

There is no need for calculating Evidence as Evidence is the same for all the posterior probabilities; thus, they can cancel out each other

2. Then apply the MAP rule

$$c_{MAP} \equiv \underset{c \in C}{\operatorname{argmax}} P(c \mid X)$$

$$= \underset{c \in C}{\operatorname{argmax}} \frac{P(X \mid c) P(c)}{P(X)}$$

$$= \underset{c \in C}{\operatorname{argmax}} P(X \mid c) P(c)$$

Bayesian Classifier

➢ Naïve Bayes classifier:

✓ Bayes classification:

$$P(C \mid \mathbf{X}) \propto P(\mathbf{X} \mid C)P(C) = P(X_1, \cdots, X_n \mid C)P(C)$$

Difficulty: learning the joint probability $P(X_1, \cdots, X_n \mid C)$

✓ Naïve Bayes classification:

✓ Assumption that all input attributes are conditionally independent!

$$\begin{aligned} P(X_1, X_2, \cdots, X_n \mid C) &= P(X_1 \mid X_2, \cdots, X_n, C)P(X_2, \cdots, X_n \mid C) \\ &= P(X_1 \mid C)P(X_2, \cdots, X_n \mid C) \\ &= P(X_1 \mid C)P(X_2 \mid C) \cdots P(X_n \mid C) \end{aligned}$$

Product of individual probabilities

It is called Naïve because it makes the assumption that the occurrence of a certain feature is independent of the occurrence of the other features!

✓ MAP classification rule for Naïve Bayes:

$$[P(x_1 \mid c^*) \cdots P(x_n \mid c^*)]P(c^*) > [P(x_1 \mid c) \cdots P(x_n \mid c)]P(c), \quad c \neq c^*, c = c_1, \cdots, c_L$$

Bayesian Classifier

- **All the classifiers have two phases:**

1. Training phase (learning phase)
2. Test phase

- **Naïve Bayes algorithm in discrete-valued features:**

✓ Training phase in Naïve Bayes classifier:

Given a training set S, for each target value of c_i $(c_i = c_1, \cdots, c_L)$

$\hat{P}(C = c_i) \leftarrow$ estimate $P(C = c_i)$ with examples in **S**;

We only need to create the probability tables

For every attribute value x_{jk} of each attribute X_j $(j = 1, \cdots, n; k = 1, \cdots, N_j)$

$\hat{P}(X_j = x_{jk} \mid C = c_i) \leftarrow$ estimate $P(X_j = x_{jk} \mid C = c_i)$ with examples in **S**;

✓ Test phase in Naïve Bayes classifier:

Given an unknown instance, we will look up tables to assign the label c^* to **X′**. This will be done as below:

$$[\hat{P}(a_1' \mid c^*) \cdots \hat{P}(a_n' \mid c^*)]\hat{P}(c^*) > [\hat{P}(a_1' \mid c) \cdots \hat{P}(a_n' \mid c)]\hat{P}(c), \quad c \neq c^*, c = c_1, \cdots, c_L$$

Bayesian Classifier

- **Famous example for discrete-valued features: Play tennis**

Observations

Features (variables)

Class labels

PlayTennis: training examples

Day	Outlook	Temperature	Humidity	Wind	PlayTennis
D1	Sunny	Hot	High	Weak	No
D2	Sunny	Hot	High	Strong	No
D3	Overcast	Hot	High	Weak	Yes
D4	Rain	Mild	High	Weak	Yes
D5	Rain	Cool	Normal	Weak	Yes
D6	Rain	Cool	Normal	Strong	No
D7	Overcast	Cool	Normal	Strong	Yes
D8	Sunny	Mild	High	Weak	No
D9	Sunny	Cool	Normal	Weak	Yes
D10	Rain	Mild	Normal	Weak	Yes
D11	Sunny	Mild	Normal	Strong	Yes
D12	Overcast	Mild	High	Strong	Yes
D13	Overcast	Hot	Normal	Weak	Yes
D14	Rain	Mild	High	Strong	No

1. Training phase:

PlayTennis: training examples

Day	Outlook	Temperature	Humidity	Wind	PlayTennis
D1	Sunny	Hot	High	Weak	No
D2	Sunny	Hot	High	Strong	No
D3	Overcast	Hot	High	Weak	Yes
D4	Rain	Mild	High	Weak	Yes
D5	Rain	Cool	Normal	Weak	Yes
D6	Rain	Cool	Normal	Strong	No
D7	Overcast	Cool	Normal	Strong	Yes
D8	Sunny	Mild	High	Weak	No
D9	Sunny	Cool	Normal	Weak	Yes
D10	Rain	Mild	Normal	Weak	Yes
D11	Sunny	Mild	Normal	Strong	Yes
D12	Overcast	Mild	High	Strong	Yes
D13	Overcast	Hot	Normal	Weak	Yes
D14	Rain	Mild	High	Strong	No

$P(\text{Play}=Yes) = 9/14$

$P(\text{Play}=No) = 5/14$

We have four variables, we calculate for each we calculate the conditional probability table

Outlook	Play=*Yes*	Play=*No*
Sunny	2/9	3/5
Overcast	4/9	0/5
Rain	3/9	2/5

Temperature	Play=*Yes*	Play=*No*
Hot	2/9	2/5
Mild	4/9	2/5
Cool	3/9	1/5

Humidity	Play=*Yes*	Play=No
High	3/9	4/5
Normal	6/9	1/5

Wind	Play=*Yes*	Play=*No*
Strong	3/9	3/5
Weak	6/9	2/5

2. Test phase:

- Given a new instance of variable values (i.e., test data), x′=(Outlook=*Sunny*, Temperature=*Cool*, Humidity=*High*, Wind=*Strong*)
- Given calculated Look up tables

P(Outlook=*Sunny*|Play=***Yes***) = 2/9
P(Temperature=*Cool*|Play=*Yes*) = 3/9
P(Huminity=*High*|Play=*Yes*) = 3/9
P(Wind=*Strong*|Play=*Yes*) = 3/9
P(Play=*Yes*) = 9/14

P(Outlook=*Sunny*|Play=***No***) = 3/5
P(Temperature=*Cool*|Play==*No*) = 1/5
P(Huminity=*High*|Play=*No*) = 4/5
P(Wind=*Strong*|Play=*No*) = 3/5
P(Play=*No*) = 5/14

Use the MAP rule to calculate Yes or No

P(*Yes*|x′): [P(*Sunny*|Yes)P(*Cool*|*Yes*)P(*High*|Yes)P(*Strong*|*Yes*)]P(Play=*Yes*) = 0.0053
P(*No*|x′): [P(*Sunny*|No) P(*Cool*|No)P(*High*|*No*)P(*Strong*|*No*)]P(Play=*No*) = 0.0206

Given the fact P(*Yes*|x′) < P(*No*|x′), we label x′ to be "*No*".

Bayesian Classifier

- **Naïve Bayes algorithm in continuous-valued features:**

✓ Conditional probability often modeled with the normal distribution

$$\hat{P}(x_j \mid c_i) = \frac{1}{\sqrt{2\pi}\sigma_{ji}} \exp\left(-\frac{(x_j - \mu_{ji})^2}{2\sigma_{ji}^2}\right)$$

μ_{ji} : mean (avearage) of feature values x_j of examples for which $c = c_i$

σ_{ji} : standard deviation of feature values x_j of examples for which $c = c_i$

✓ Learning Phase: for $\mathbf{X} = (X_1, \cdots, X_F)$, $C = c_1, \cdots, c_L$

Output: normal distributions $P(C = c_i)$ $i = 1, \cdots, L$

✓ Test Phase: Given an unknown instance $\mathbf{X}' = (a_1', \cdots, a_n')$

- ✓ **Instead of looking-up tables, calculate conditional probabilities** with all the normal distributions achieved in the learning phase
- ✓ Apply the MAP rule to assign a label (the same as done for the discrete case)

Bayesian Classifier

- **Example for continuous-valued features: Sex classification** (https://en.wikipedia.org/wiki/Naive_Bayes_classifier)

1. Training phase:

Sex	height (feet)	weight (lbs)	foot size(inches)
male	6	180	12
male	5.92 (5'11")	190	11
male	5.58 (5'7")	170	12
male	5.92 (5'11")	165	10
female	5	100	6
female	5.5 (5'6")	150	8
female	5.42 (5'5")	130	7
female	5.75 (5'9")	150	9

✓ Create prior probability:

$$P(male) = P(female) = 0.5$$

✓ Generate the parameters of the Gaussian distribution using training dataset:

Sex	mean (height)	variance (height)	mean (weight)	variance (weight)	mean (foot size)	variance (foot size)
male	5.855	3.5033*10-02	176.25	1.2292*10+02	11.25	9.1667*10-01
female	5.4175	9.7225*10-02	132.5	5.5833*10+02	7.5	1.6667

Bayesian Classifier

2. Test phase:

Sex	height (feet)	weight (lbs)	foot size(inches)
sample	6	130	8

P(*sex* | x′)?

$$\text{posterior (male)} = \frac{P(\text{male})\, p(\text{height} \mid \text{male})\, p(\text{weight} \mid \text{male})\, p(\text{foot size} \mid \text{male})}{evidence}$$

$$\text{posterior (female)} = \frac{P(\text{female})\, p(\text{height} \mid \text{female})\, p(\text{weight} \mid \text{female})\, p(\text{foot size} \mid \text{female})}{evidence}$$

P(*height* | male)= $\frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(\frac{-(6-\mu)^2}{2\sigma^2}\right) \approx 1.5789$

Replace μ *with the height mean for males* (5.855), *and* σ *with the height variance for males* ($3.5033 * 10^{-2}$)

Bayesian Classifier

2. Test phase:

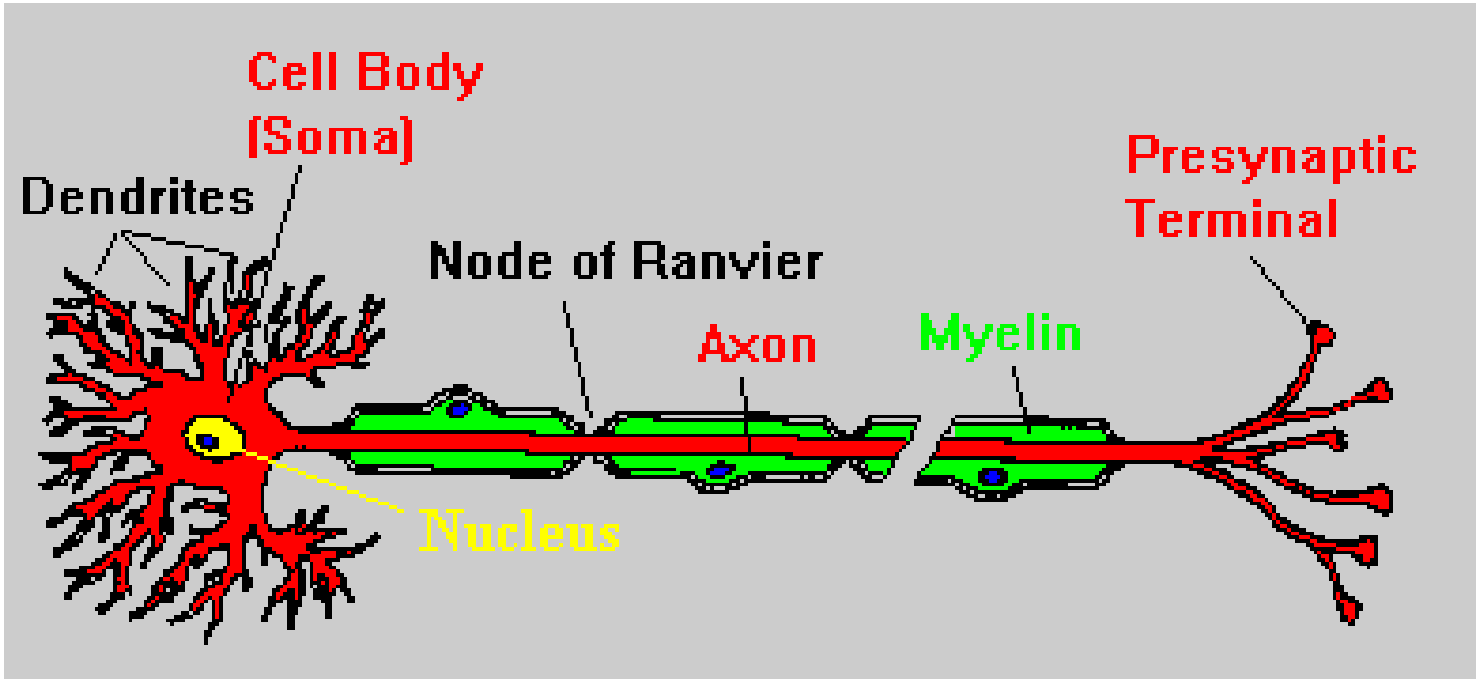
$$p(\text{weight} \mid \text{male}) = 5.9881 \cdot 10^{-6}$$
$$p(\text{foot size} \mid \text{male}) = 1.3112 \cdot 10^{-3}$$
$$\text{posterior numerator (male)} = \text{their product} = 6.1984 \cdot 10^{-9}$$
$$P(\text{female}) = 0.5$$
$$p(\text{height} \mid \text{female}) = 2.2346 \cdot 10^{-1}$$
$$p(\text{weight} \mid \text{female}) = 1.6789 \cdot 10^{-2}$$
$$p(\text{foot size} \mid \text{female}) = 2.8669 \cdot 10^{-1}$$
$$\text{posterior numerator (female)} = \text{their product} = 5.3778 \cdot 10^{-4}$$

- ✓ Since the posterior probability is greater for females, we predict the test sample is female!

- ✓ Naïve Bayes advantages:
  - ✓ Training is very easy and fast; just requires considering each attribute in each class separately.
  - ✓ Test is straightforward; just looking up tables or calculating conditional probabilities with normal distributions.
  - ✓ Working well sometimes for data violating the assumption (the independence assumption)!

Neural Network

➢ The Biological Neuron



Reminder: The human brain is made of about 100 billions of such neurons.

Neural Network

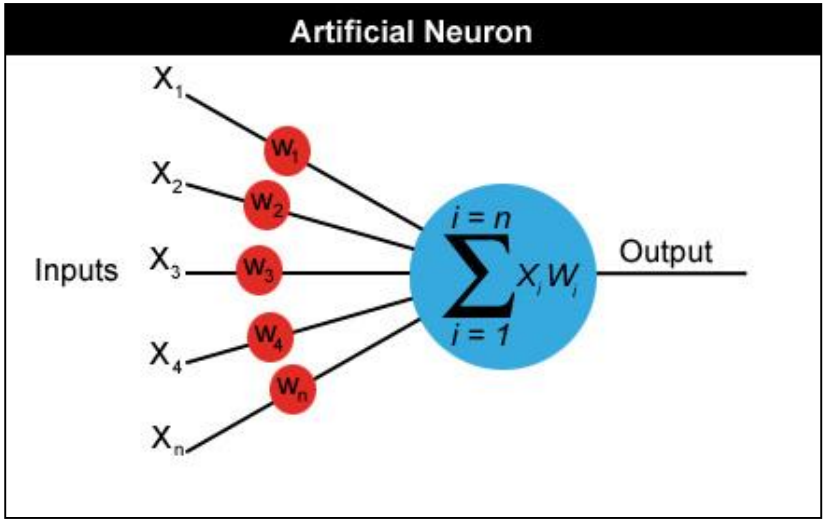
➢ Characteristics of biological neurons

- ✓ Massive connectivity
- ✓ Nonlinear, parallel, and robust
- ✓ Adaptive to surrounding
- ✓ Learn and generalize from known examples

Artificial neural networks mimic some of the properties of the biological neural networks!

Neural Network

- ✓ Neural networks (NN) are made up of many artificial neurons.
- ✓ Each input into the neuron has its own weight associated with it.
- ✓ A neuron can have any number of inputs from one to n, where n is the total number of inputs.
- ✓ The inputs may be represented therefore as x1, x2, x3… xn.
- ✓ The corresponding weights for the inputs as w1, w2, w3… wn.
- ✓ Output is a weighted some of the inputs which are fed into an activation function



Neural Network

- **Activation Functions**

1) **Threshold Function**

$f(v) = 1$ if $v \geq 0$

$= 0$ otherwise

2) **Piecewise-Linear Function**

$f(v) = 1$ if $v \geq ½$

$= v$ if $½ > v > -½$

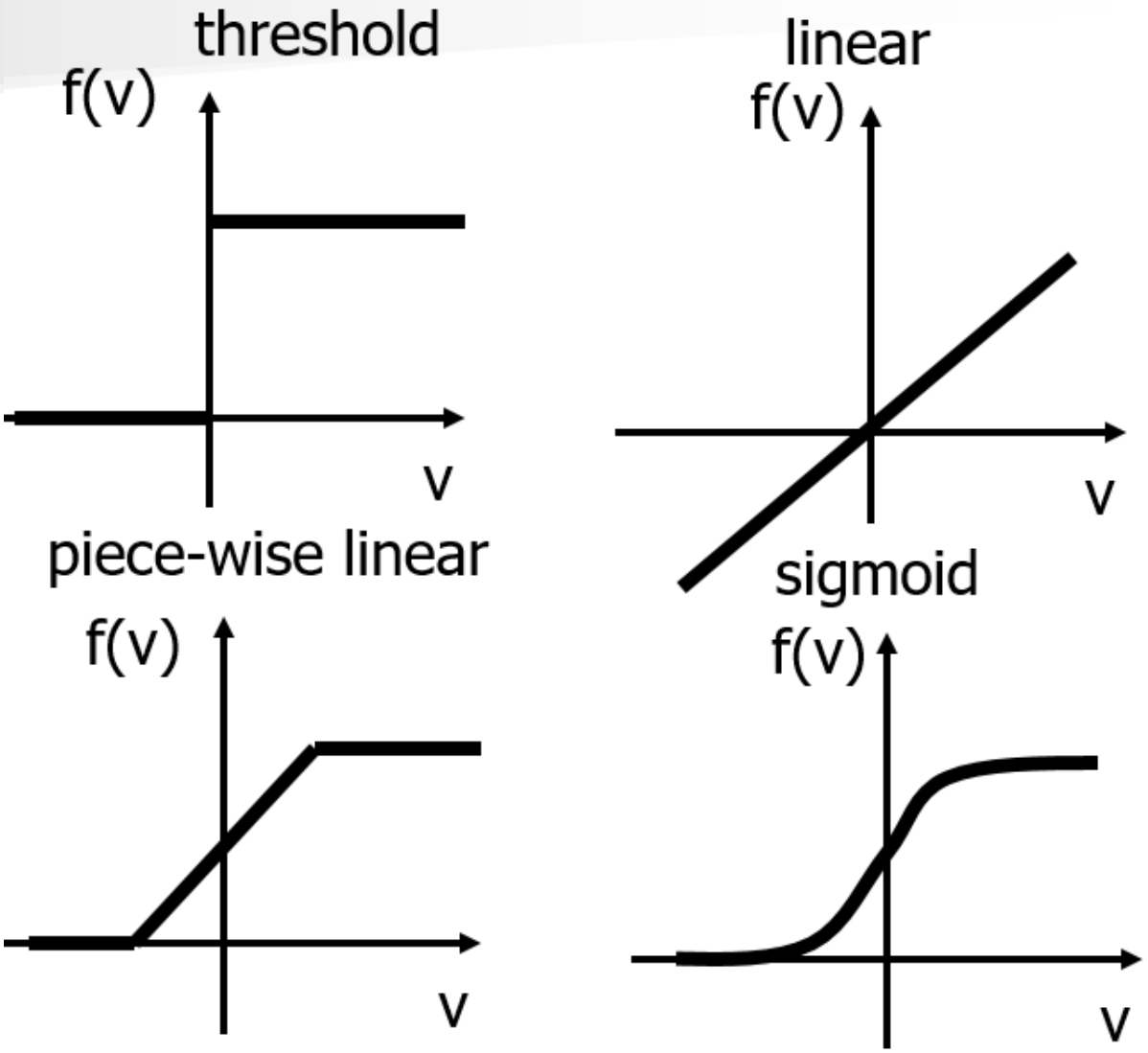
$= 0$ otherwise

3) **Sigmoid Function**

$f(v) = 1/\{1 + exp(-av)\}$

etc..

Neural Network



Neural Network

- **Neural network is characterized by:**

1) Architecture
2) Learning (update scheme of weights and/or outputs)

- **Architecture:**

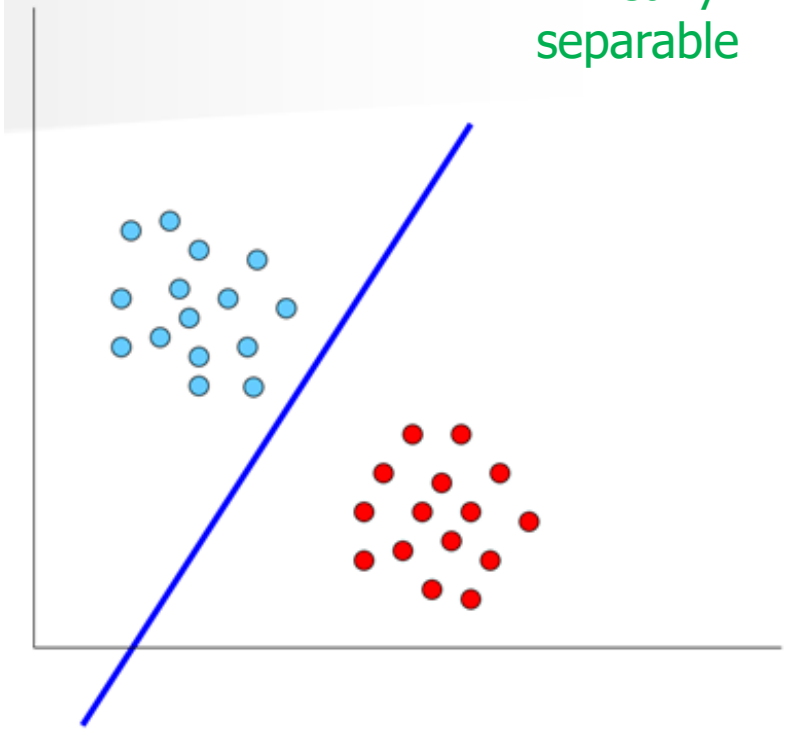
✓ **Layered** (single/multiple): Feed forward network; the neurons in each layer feed their output forward to the next layer until it get to the final output layer of NN.
✓ **Recurrent** : At least one feedback loop
✓ **Competitive** : p – dimensional array of neurons with a set of nodes supplying input to each element of the array (nodes compete for the right to respond to a subset of the input data).

- **Learning:**

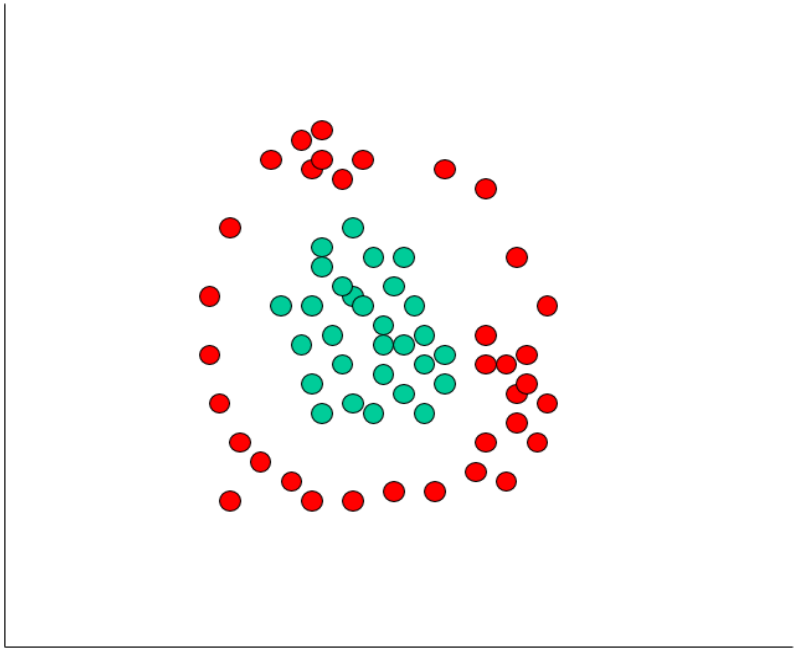
✓ **Supervised**: the class labels are determined
✓ **Unsupervised** (Self-Organized): the class labels are not determined
✓ **Reinforcement** : Trial and error, no class label but can learn from the situation

Neural Network

Linearly separable



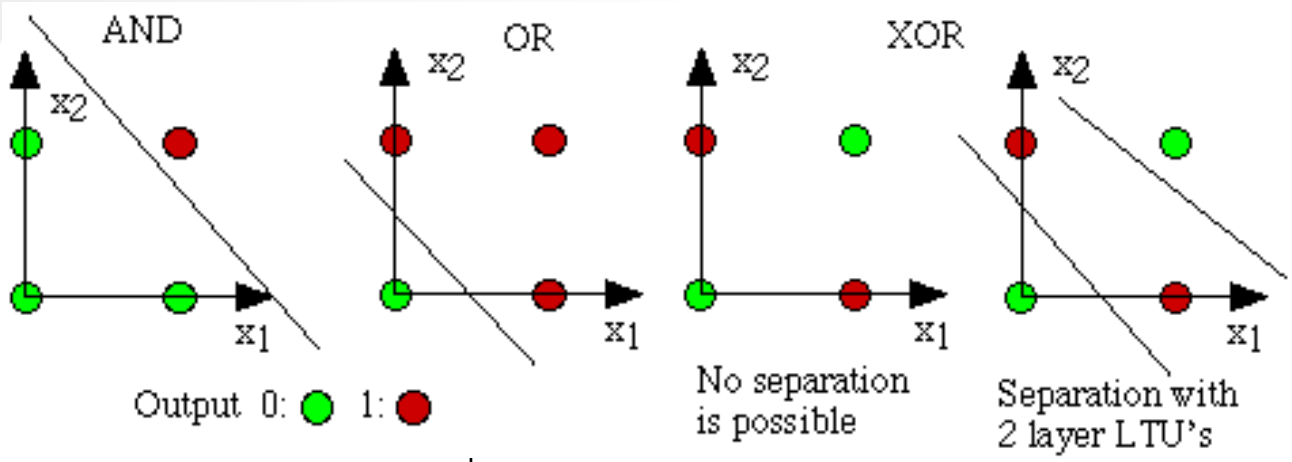
?



Neural Network

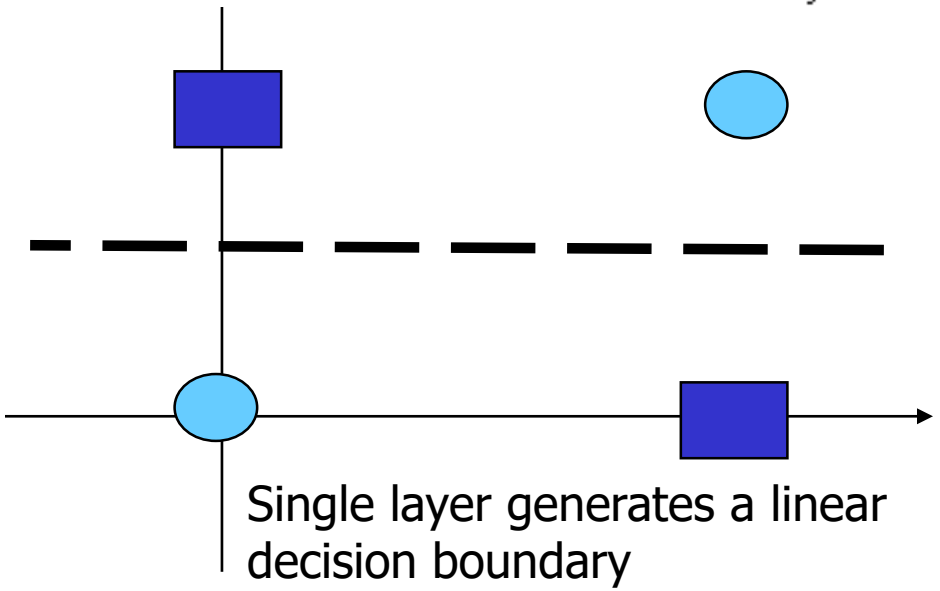
- ✓ **Linearly separable**

OR & AND are linearly separable Boolean operators



XOR is not linearly separable!!

Perceptron does not work here

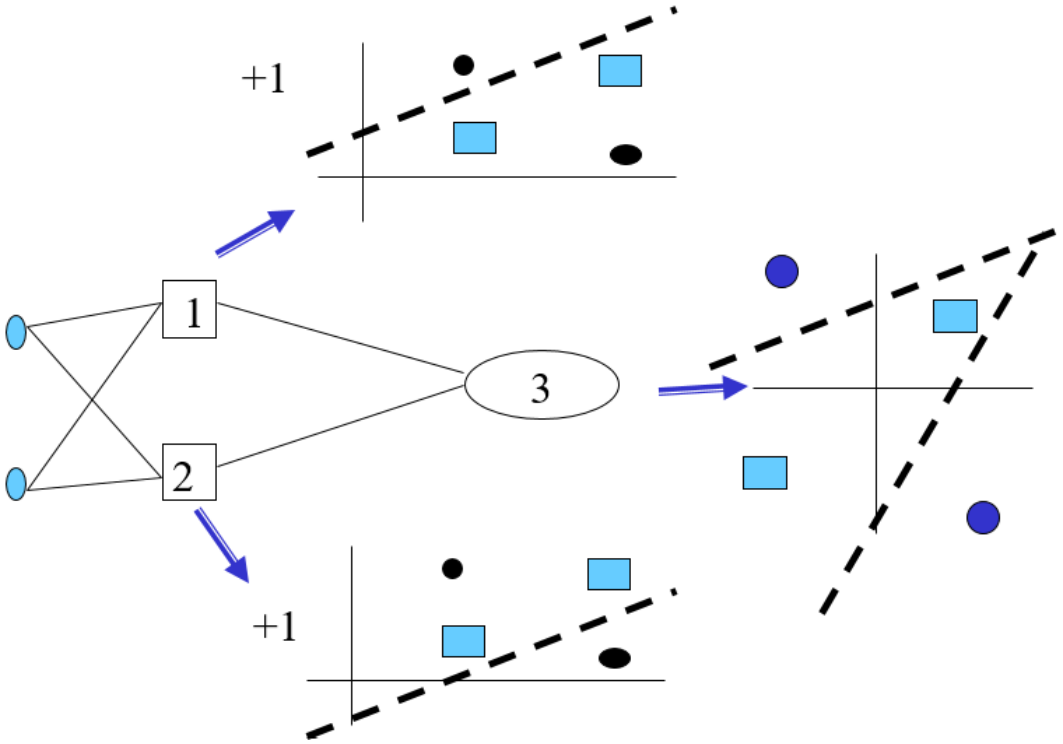


Neural Network

Solution for XOR problem: Adding hidden layer allows more target functions to be represented!

- ✓ Hidden units allow a network to learn non-linear functions.

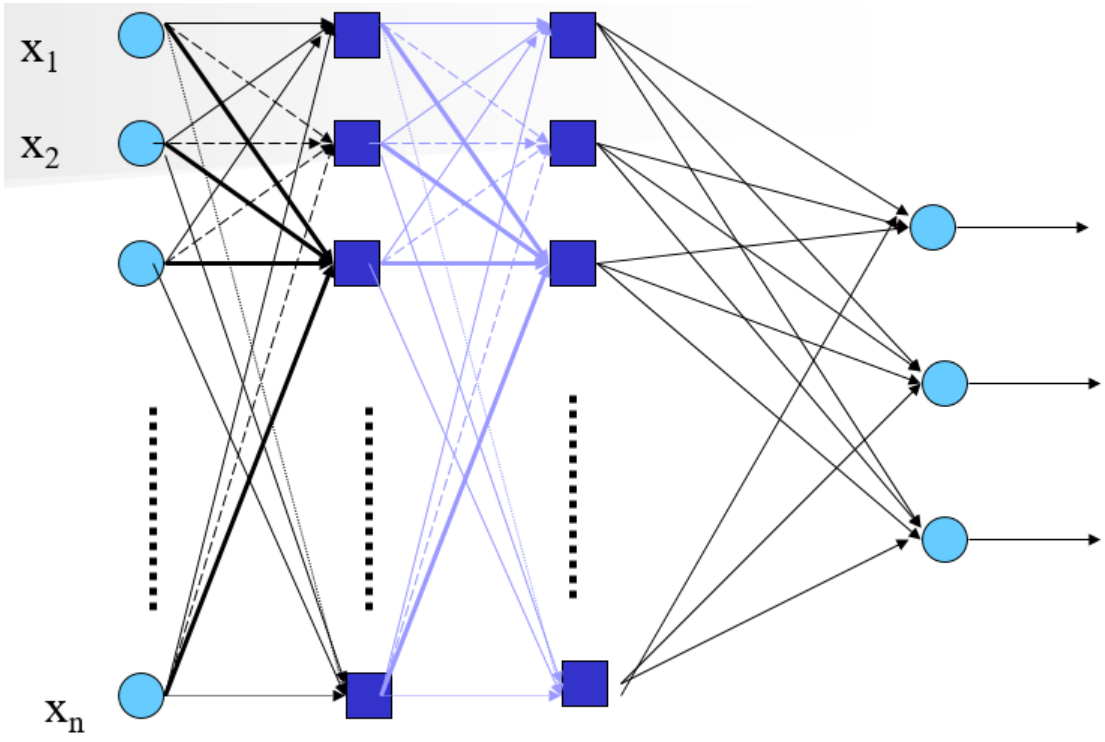
Minsky & Papert (1969) offered solution to XOR problem by combining perceptron unit responses using a second layer of units



Neural Network

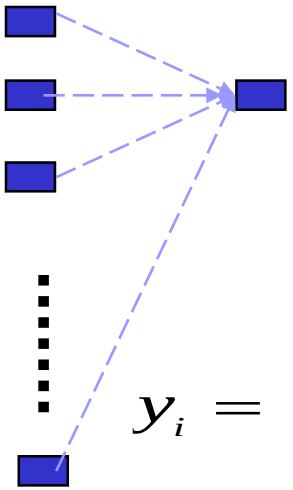
- **Multilayer perceptron (MLP):**
  - ✓ MLP is a class of feed-forward neural network.
  - ✓ Each layer is fully connected to the layer after.

An example of three-layer network



Neural Network

- **Properties of an MLP architecture:**
- ✓ Consists at of least three layered nodes (input layer, hidden layer, and output layer).
- ✓ No connections within a layer.
- ✓ Each node in an MLP is a perceptron (neuron) with an activation function
- ✓ No direct connections between input and output layers
- ✓ Fully connected between layers
- ✓ Number of output units need not equal number of input units
- ✓ Number of hidden units per layer can be more or less than input or output units

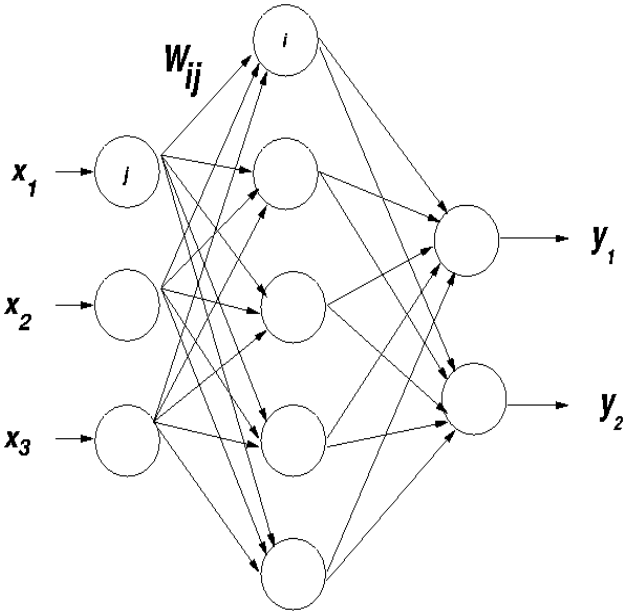


$$y_i = f\left(\sum_{j=1}^{m} w_{ij} x_j + b_i\right)$$

Neural Network

- ✓ Suppose we have a training set $X=\{(x_1,y_1),....,(x_n,y_n)\}$, where $x \in R^p$ and $y \in R^q$.

There is an unknown functional relationship between x and y (e.g., y = F(x))



- ✓ Our objective is to learn ***F,*** given X.

When an input vector is given to an MLP it computes a function. The function ***F**** which the MLP computes has the weights and biases of each nodes as a parameter. Let ***W*** be a vector which contains all the weights and biases associated with the MLP as its elements**,** thus the MLP computes the function ***F*(W , x).***

Our objective would be to find such a *W* which minimizes the sum squared error as below:

$$E = \tfrac{1}{2} \sum_i (y_i - F^*(W , x_i))^2$$

where y is the given class label (in the training dataset)

Neural Network

- ✓ The solution exists when we have at least one parameter vector $\boldsymbol{w}_{opt} = (w_{1,opt},\ldots,w_{N,opt})^T$ exists, such that:

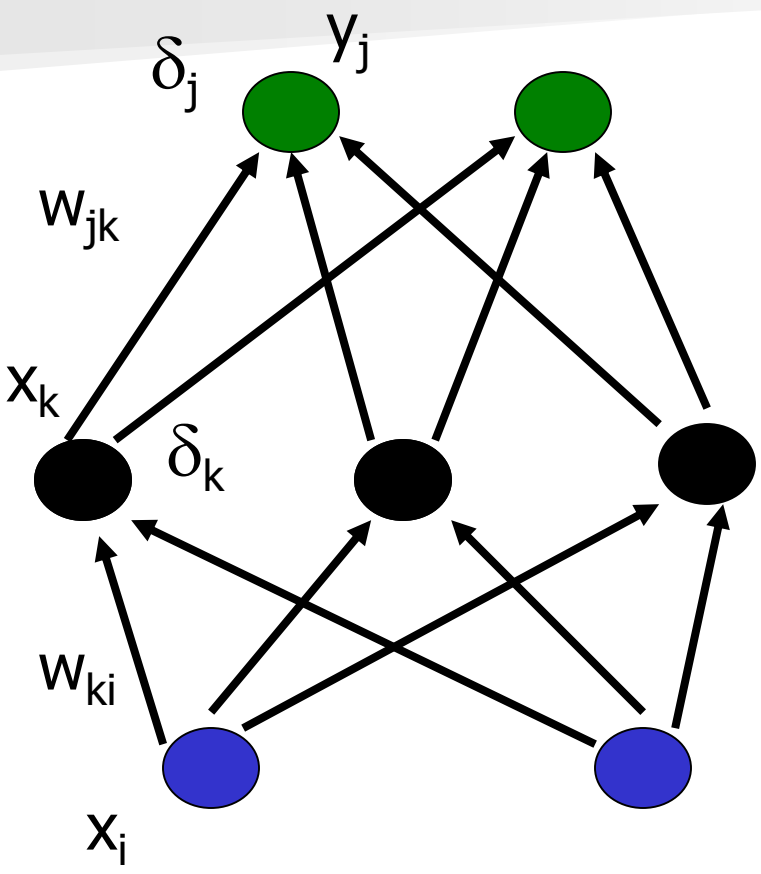
$$\frac{\partial J(w_{opt})}{\partial w_i} = 0 \qquad 1 \le i \le N$$

- ➢ **Backpropagation learning algorithm (BP):** Forward activity backward error

- ✓ **Forward pass phase:** where the output is calculated.
- ✓ **Backward pass phase:** computes 'error signal', *propagates* the error *backwards* through network starting at output units (According to the error the weights are updated)

Neural Network

- ✓ **Backpropagation conecpt**



Backward step:
propagate errors from output to hidden layer

Forward step:
Propagate activation from input to output layer

Neural Network

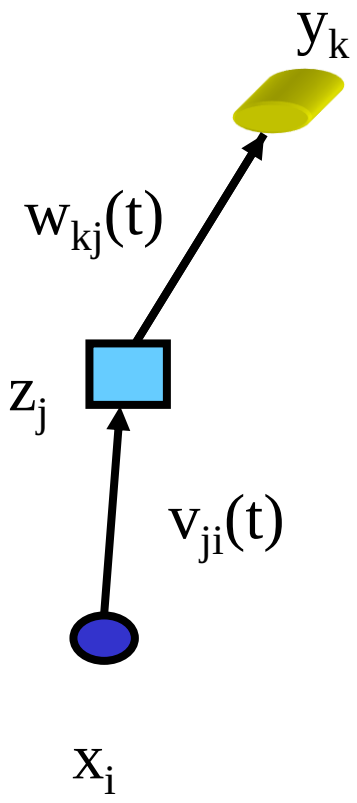
- **Example for a three-layer network**

✓ Forward Pass:

1. Compute values for hidden units

$$u_j(t) = \sum_i v_{ji}(t) x_i(t)$$

$$z_j = g(u_j(t))$$



2. Compute values for output units

$$a_k(t) = \sum_j w_{kj}(t) z_j$$

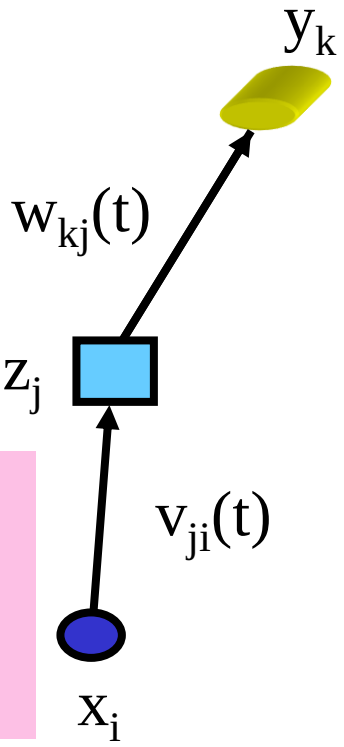
$$y_k = g(a_k(t))$$

Neural Network

✓ Backward Pass:

1. Compute errors:

$$E(t) = \frac{1}{2}\sum_{k=1}(d_k(t) - y_k(t))^2$$



2. Compute weights:

$$w_{ij}(t+1) = w_{ij}(t) - a\frac{\partial E(t)}{\partial w_{ij}(t)}$$

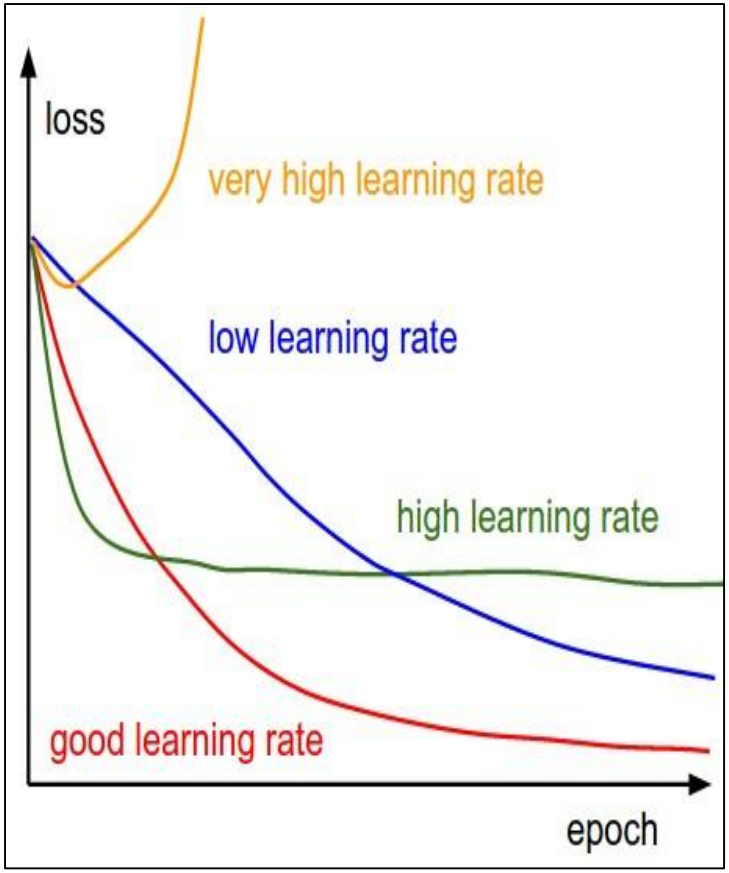
Gradient Descent Learning Update

Where ***w(0)*** is any initial parameter vector and α is a positive values sequence of learning rate.

This optimization procedure may lead to a **local minima** of the cost function *J*.

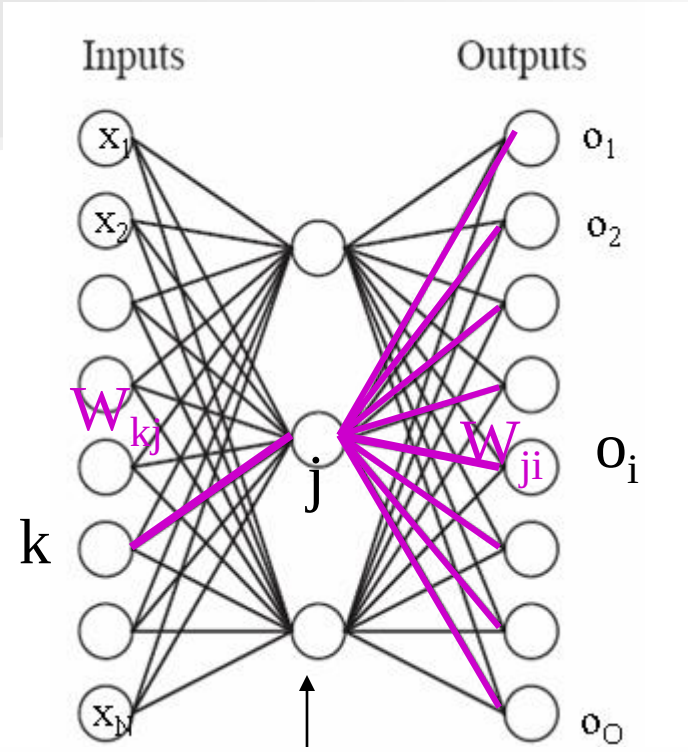
Neural Network

- **Learning rate: Relatively small (.01 - .5 common)**



Neural Network

➢ BP Learning Algorithm for MLP:



Perceptron update:

$$W_j \leftarrow W_j + \alpha \times Err \times g'(in) \times x_j$$

$Err_i = y_i - o_i$

Output layer weight update (**similar to perceptron**)

$$W_{j,i} \leftarrow W_{j,i} + \alpha \times a_j \times \Delta_i$$

$$\Delta_i = Err_i \times g'(in_i)$$

Hidden layer: back-propagate the error from the output layer:

$$W_{k,j} \leftarrow W_{k,j} + \alpha \times a_k \times \Delta_j \ .$$

How to compute the errors for the hidden units?

$$\Delta_j = g'(in_j) \underbrace{\sum_i W_{j,i} \Delta_i}_{} \ .$$

Err_j → "Error" for hidden node j

Neural Network

- **BP algorithm:**
  - ✓ **Forward Propagation:**

Step 1: Initialize weights at random, choose a learning rate η

Step 2: Do forward pass through net (with fixed weights) to produce output(s)

-i.e., in Forward Direction, layer by layer:

- Inputs applied
- Multiplied by weights
- Summed
- Apply the activation function (g)
- Output passed to each neuron in next layer

-Repeat above until network output(s) produced

Neural Network

- **BP algorithm:**
  - ✓ **Back-propogation of error (weight update):**

Step 3:
Compute error for each output ($\Delta_i = Err_i \times g'(in_i)$)
Layer-by-layer compute error for each hidden layer (by back-propogating error $\Delta_j = g'(in_j) \sum_i W_{j,i} \Delta_i$)

Step 4: Update all the weights by gradient descent, go back to step 2.

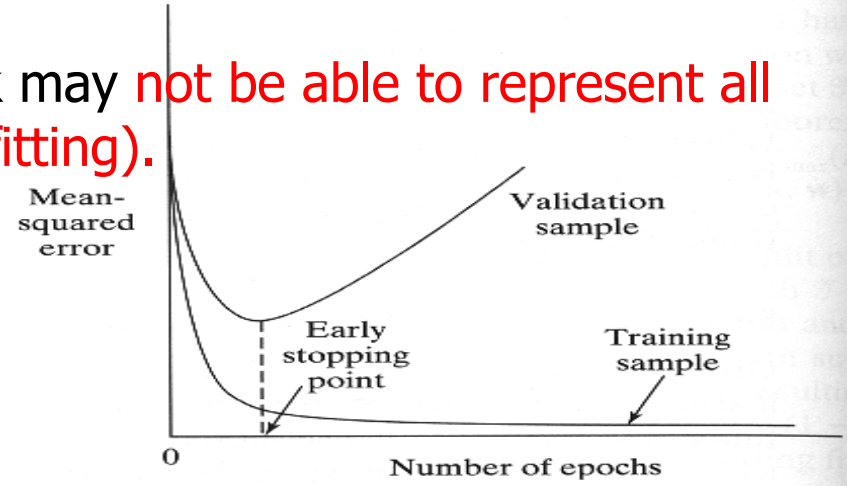
- **Stopping criteria:**
  - ✓ **Average squared error change:** Back-prop is considered to have converged when the absolute rate of change in the average squared error per epoch is sufficiently small (in the range [0.1, 0.01]).

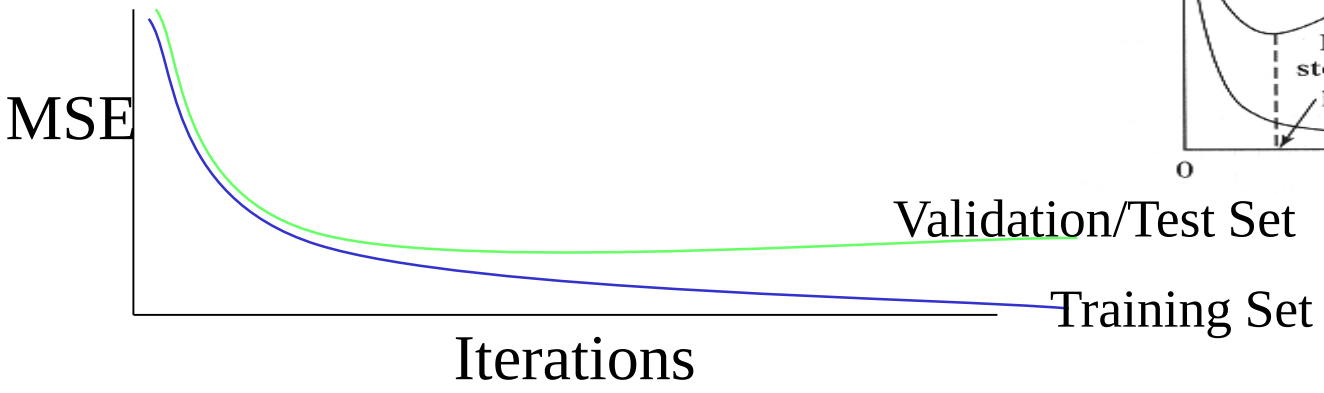
Neural Network

- ✓ **Generalization based criterion:** After each iteration (epoch) the NN is tested for generalization. If the generalization performance is adequate then stop (A network is said to generalize well if it produces correct output (or nearly so) for an input data point never used to train the network).

- ✓ **How many hidden units?**
  Given too many hidden units, a neural net will simply memorize the input patterns (overfitting).

  Given too few hidden units, the network may not be able to represent all of the necessary generalizations (underfitting).





Neural Network

➢ Cross-validation:

- ✓ Often we use a validation set (separate from the training or test set) for deciding on optimum parameters such as stopping criteria, number of iterations, number of layers, etc.
- ✓ These validation set usually can be obtained through a cross-validation (CV) procedure (e.g., 10-fold cross validation)