Lecture 16: Machine learning

IST5573

統計方法 Statistical methods

2016/12/28

Machine learning

- Closely related to
  - Computational statistics
  - Mathematical optimization
  - Data mining
  - Supervised / unsupervised learning
- We will be studying
  - Support vector machine
  - Neural networks
  - Classification and regression tree
  - K-nearest neighbor

Support vector machine

These slides are courtesy of Jinwei Gu 2008/10/16

http://slideplayer.com/slide/4043415/

Support vector machine (SVM)

- A classifier derived from statistical learning theory by Vapnik, et al. in 1992
- SVM became famous when, using images as input, it gave accuracy comparable to neural-network with hand-designed features in a handwriting recognition task
- Currently, SVM is widely used in object detection & recognition, content-based image retrieval, text recognition, biometrics, speech recognition, etc.

Discriminant function

- The classifier is said to assign a feature vector $\boldsymbol{X}$ to class π_i if

$$g_i(\boldsymbol{x}) > g_j(\boldsymbol{x}) \qquad \text{for all } j \neq i$$

- For two-class case, $g(\boldsymbol{x}) \equiv g_1(\boldsymbol{x}) - g_2(\boldsymbol{x})$
  Decide π_1 if $g(x) > 0$; otherwise decide π_2
- An example we've learned before:

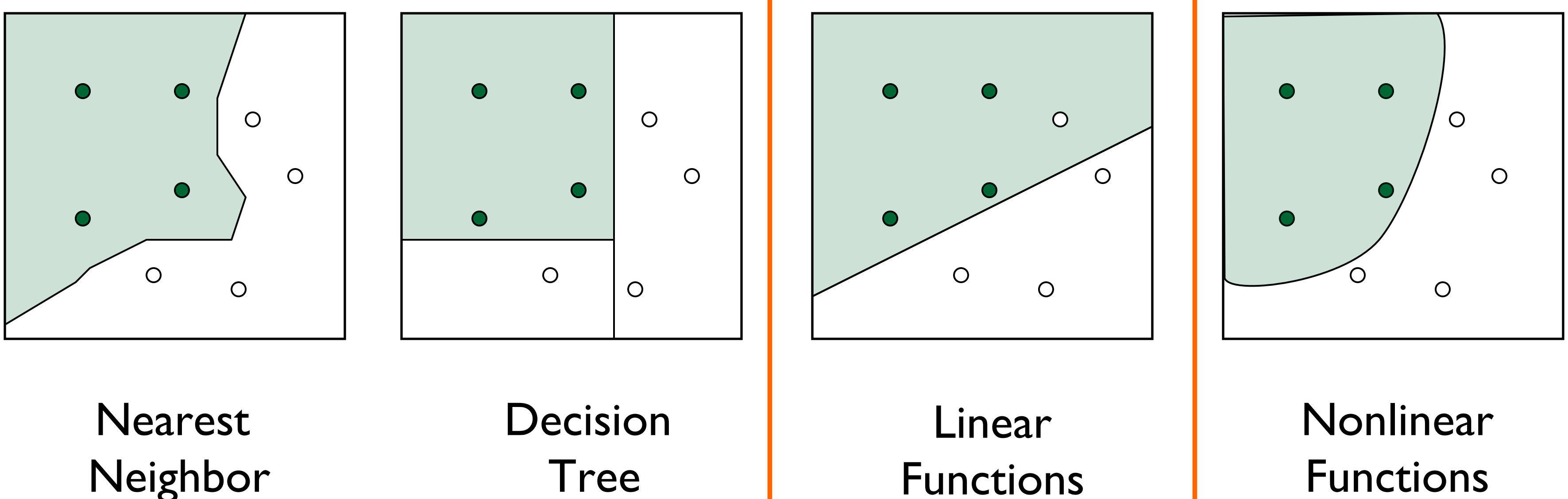
  LDA, QDA for equal priors and misclassification cost:

$$g(\boldsymbol{x}) \equiv f_1(\boldsymbol{x}) - f_2(\boldsymbol{x})$$

where $f_i(\cdot)$: probability density functions for π_i

Discriminant function

- It can be arbitrary functions of x, such as:



$$g(\mathbf{x}) = \mathbf{w}^T \mathbf{x} + b$$

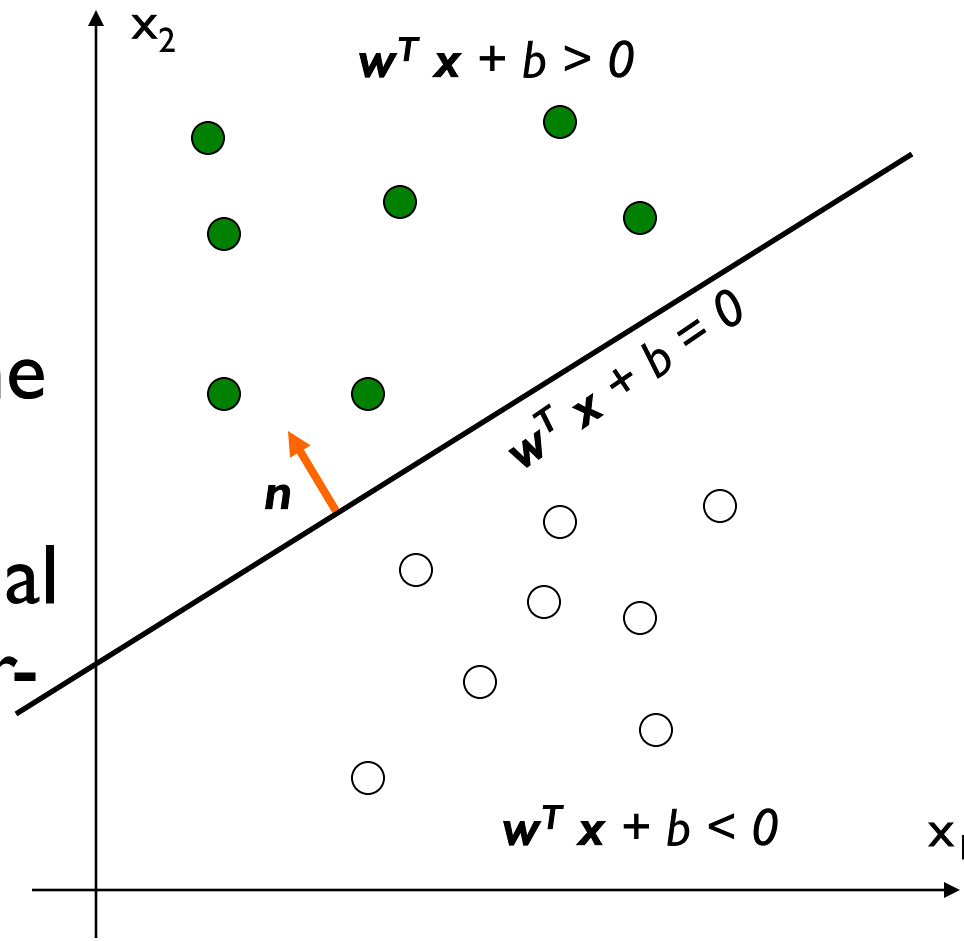
Linear discriminant function

- $g(\boldsymbol{x})$ is a linear function:

  $$g(\boldsymbol{x}) = \boldsymbol{w}^T \boldsymbol{x} + b$$

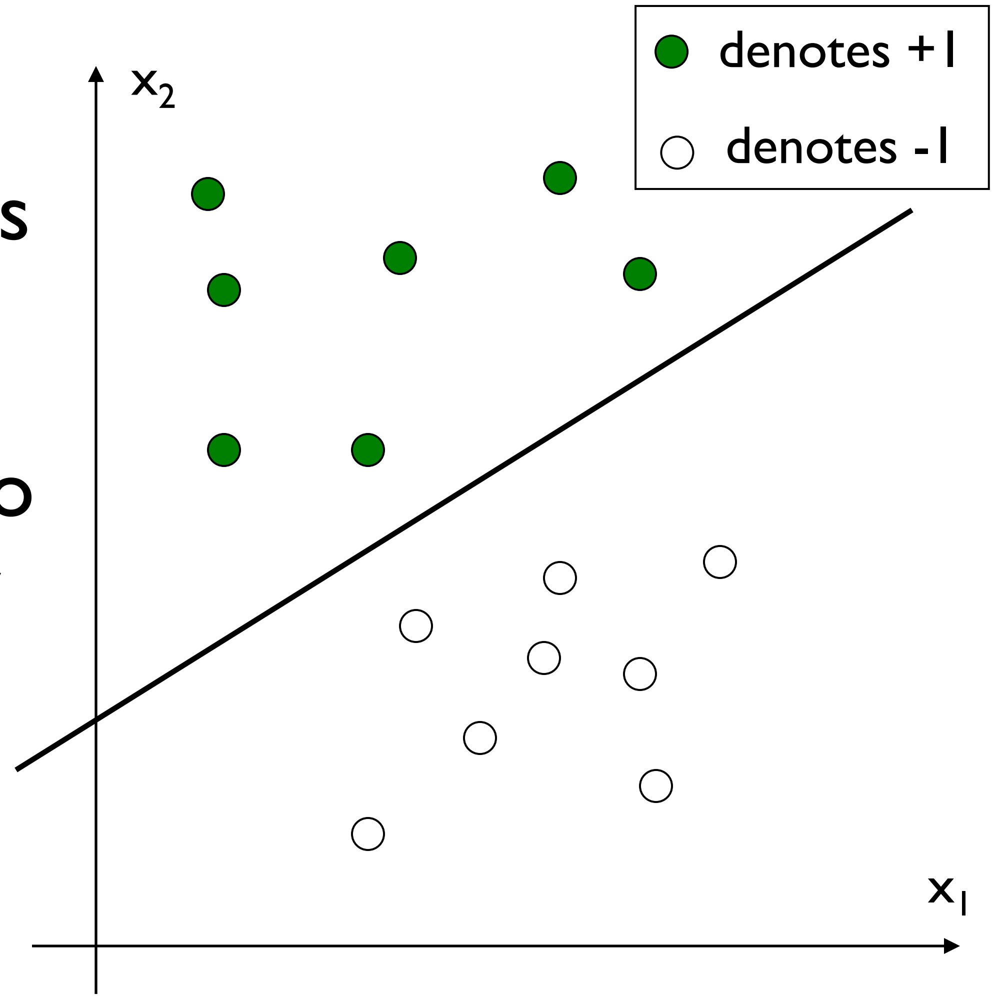
- A hyper-plane in the feature space
- (Unit-length) normal vector of the hyper-plane:

  $$\boldsymbol{n} = \frac{\boldsymbol{w}}{\|\boldsymbol{w}\|}$$



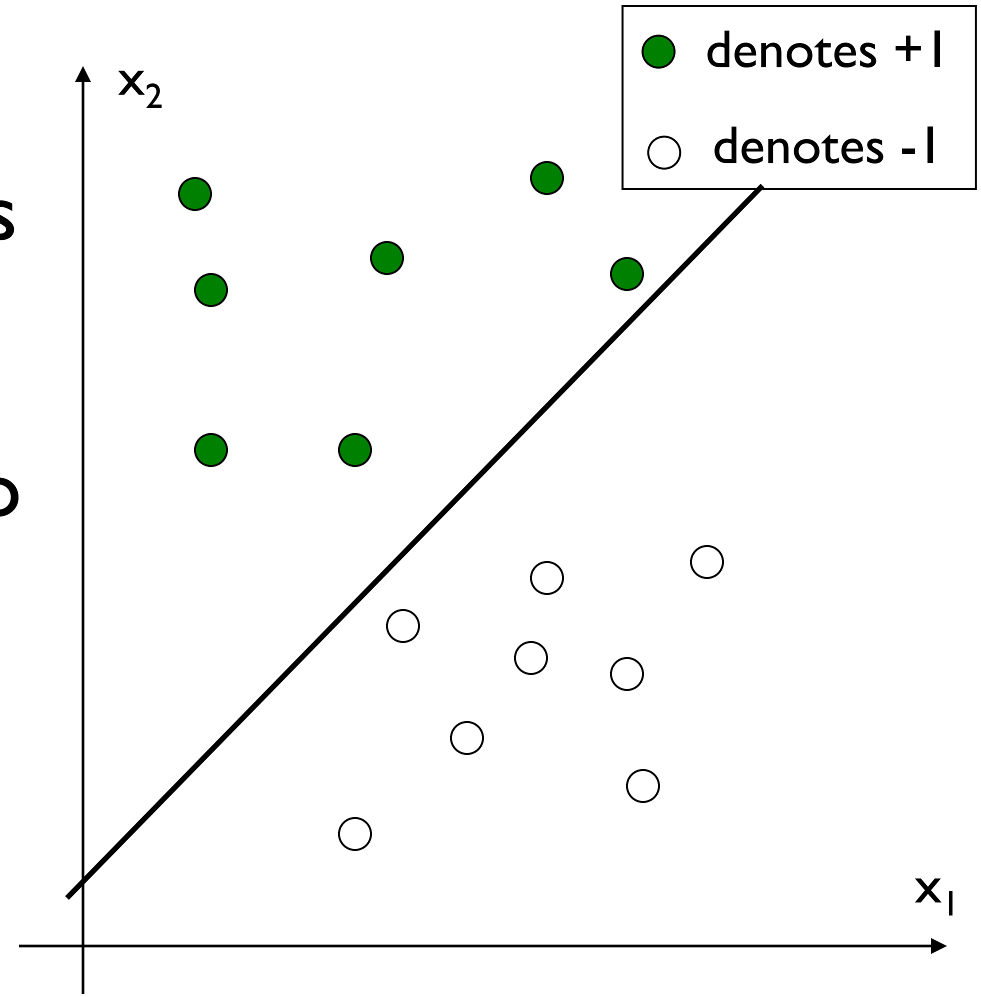
Linear discriminant function

- How would you classify these points using a linear discriminant function in order to minimize the error rate?
- Infinite number of answers!



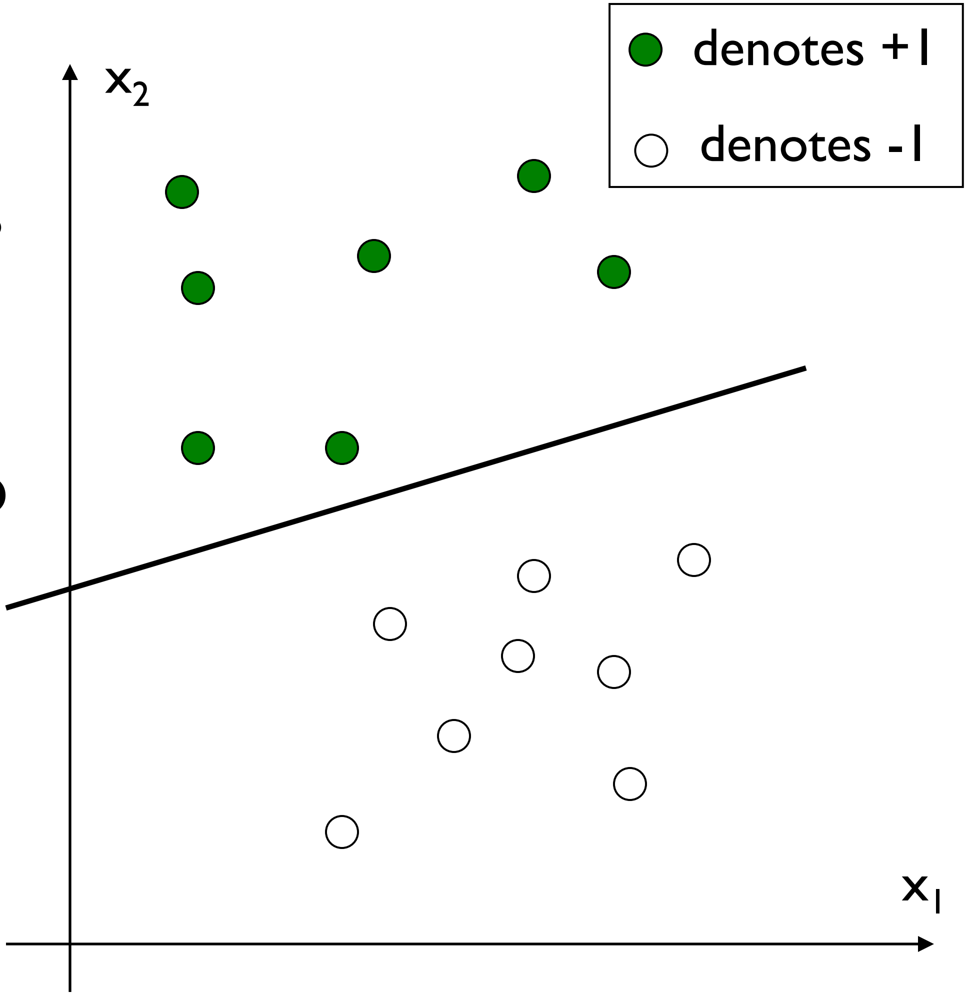
Linear discriminant function

- How would you classify these points using a linear discriminant function in order to minimize the error rate?
- Infinite number of answers!



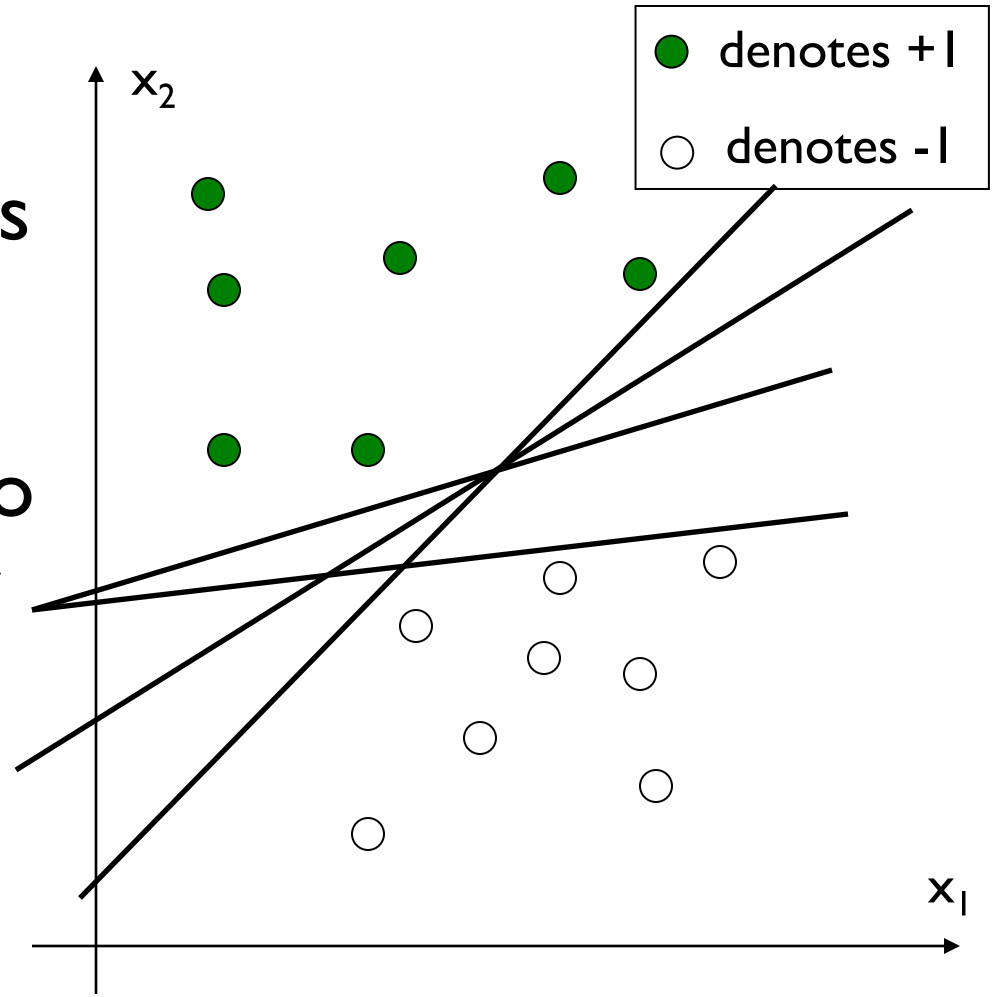
Linear discriminant function

- How would you classify these points using a linear discriminant function in order to minimize the error rate?
- Infinite number of answers!



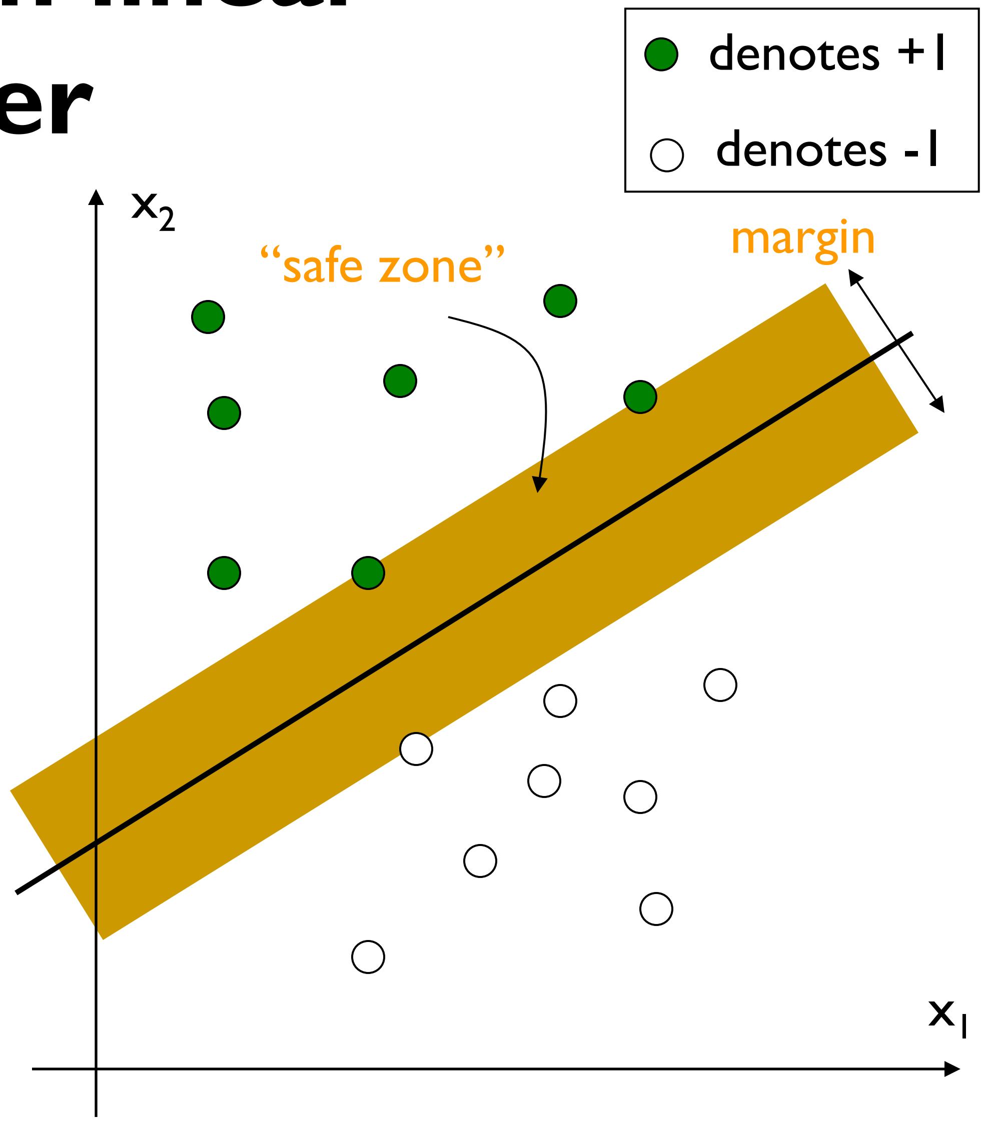
Linear discriminant function

- How would you classify these points using a linear discriminant function in order to minimize the error rate?
- Infinite number of answers!
- Which one is the best?

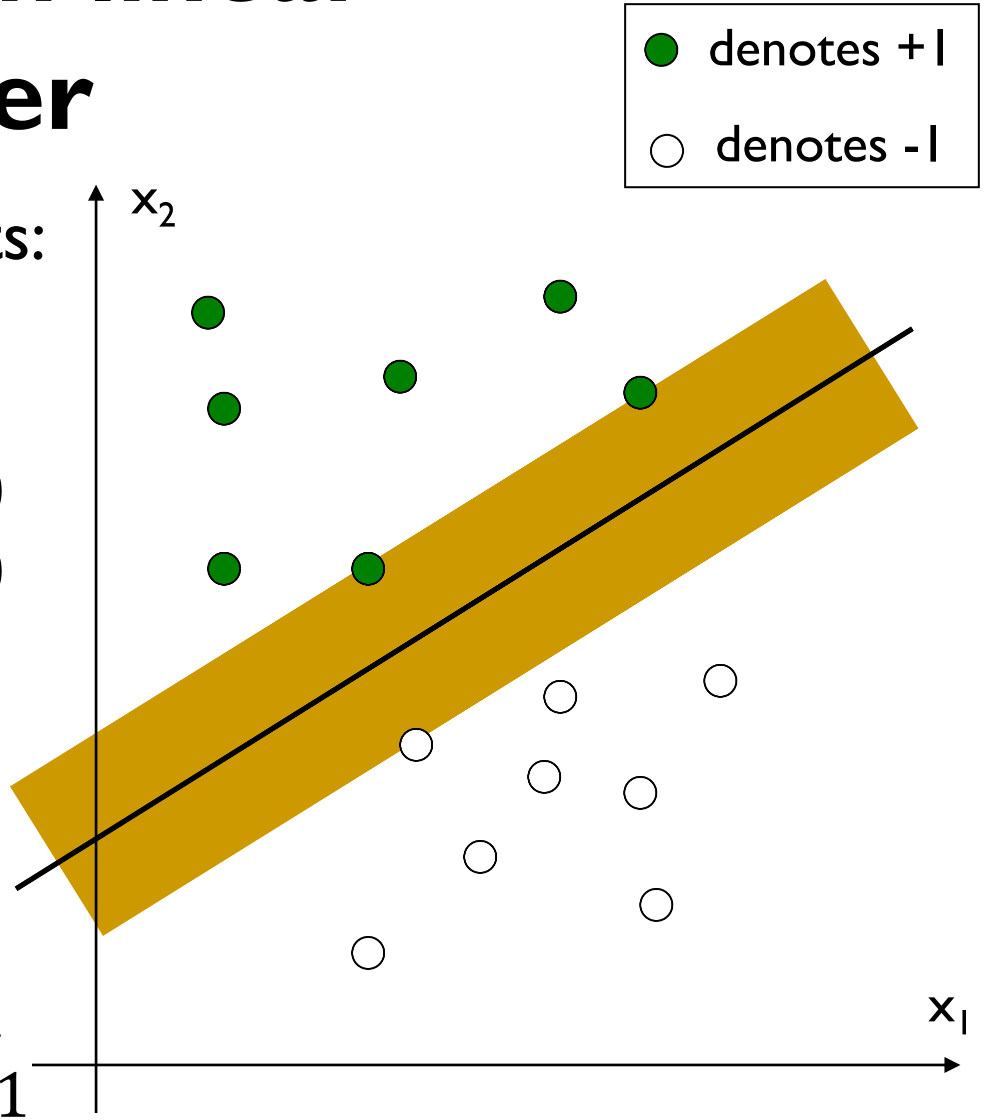


Large margin linear classifier

- The linear discriminant function (classifier) with the maximum margin is the best
- Margin is defined as the width that the boundary could be increased by before hitting a data point
- Why it is the best?
  - Robust to outliners and thus strong generalization ability



Large margin linear classifier



- Given a set of data points: $\{(\boldsymbol{x}_i, y_i)\}, i = 1, \cdots, n$, where

  For $y_i = +1, \boldsymbol{w}^T \boldsymbol{x}_i + b > 0$
  For $y_i = -1, \boldsymbol{w}^T \boldsymbol{x}_i + b < 0$

- With a scale transformation on both $\boldsymbol{w}$ and b, the above is equivalent to

  For $y_i = +1, \boldsymbol{w}^T \boldsymbol{x}_i + b > 1$
  For $y_i = -1, \boldsymbol{w}^T \boldsymbol{x}_i + b < -1$

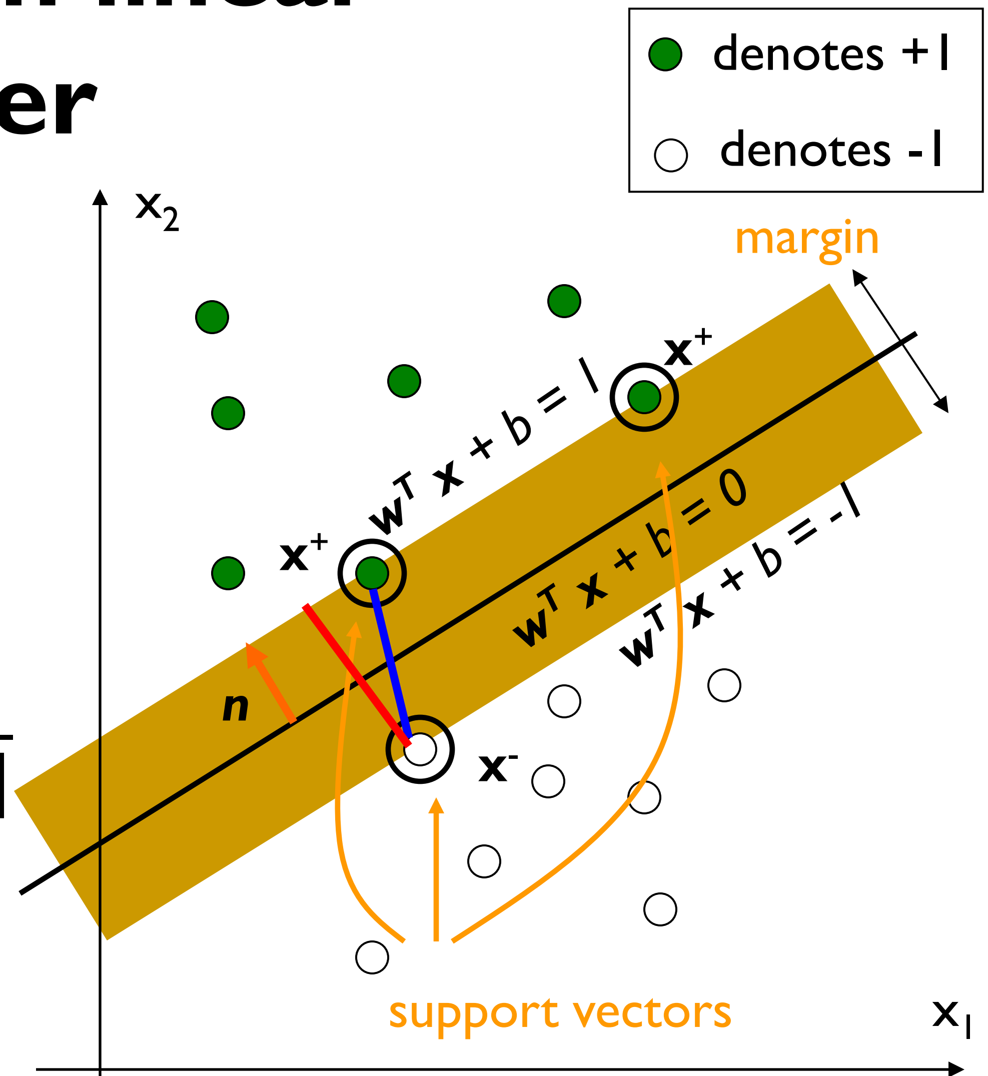
Large margin linear classifier

- We know that

$$\boldsymbol{w}^T\boldsymbol{x}^+ + b = 1$$
$$\boldsymbol{w}^T\boldsymbol{x}^- + b = -1$$

- The margin width is:

$$M = (\boldsymbol{x}^+ - \boldsymbol{x}^-)\cdot \boldsymbol{n}$$
$$= (\boldsymbol{x}^+ - \boldsymbol{x}^-)\cdot \frac{\boldsymbol{w}}{\|\boldsymbol{w}\|} = \frac{2}{\|\boldsymbol{w}\|}$$



The margin is the projection of $(\boldsymbol{x}^+ - \boldsymbol{x}^-)$ along the direction of $\boldsymbol{w}$

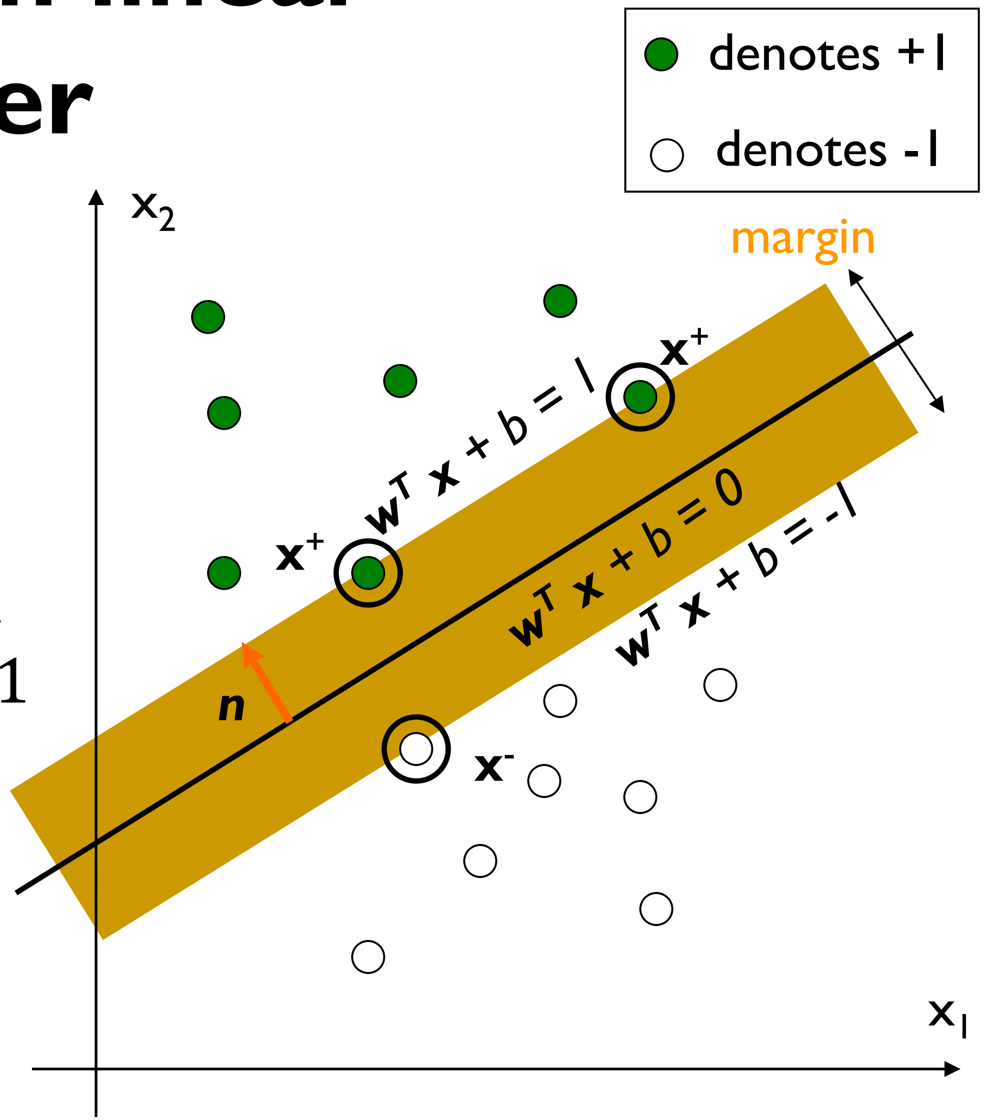
Large margin linear classifier

- Formulation:

$$\text{maximize } \frac{2}{\|\boldsymbol{w}\|}$$

such that

$$\text{For } y_i = +1, \boldsymbol{w}^T \boldsymbol{x}_i + b > 1$$
$$\text{For } y_i = -1, \boldsymbol{w}^T \boldsymbol{x}_i + b < -1$$



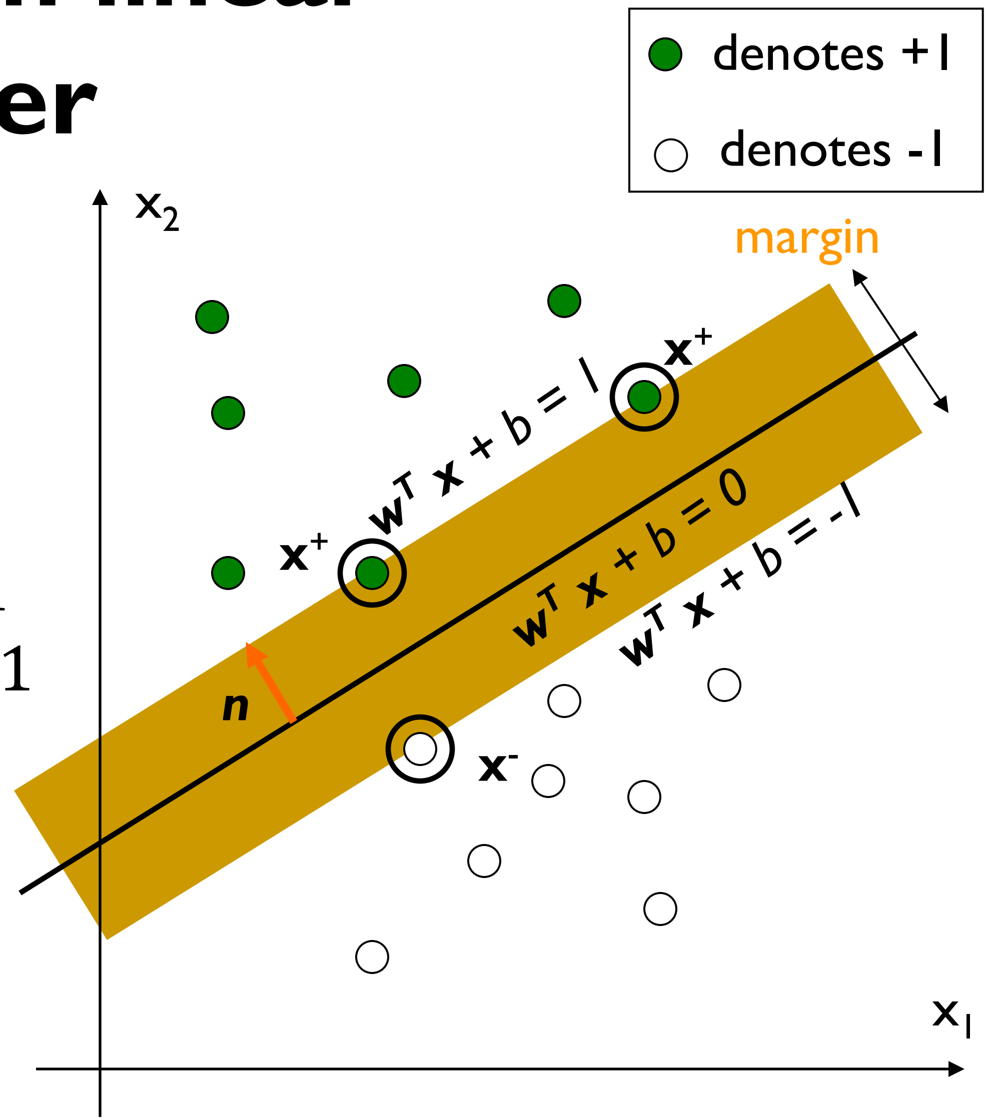
Large margin linear classifier

- Formulation:

$$\text{minimize } \frac{1}{2}\|\boldsymbol{w}\|^2$$

such that

$$\text{For } y_i = +1, \boldsymbol{w}^T\boldsymbol{x}_i + b > 1$$
$$\text{For } y_i = -1, \boldsymbol{w}^T\boldsymbol{x}_i + b < -1$$



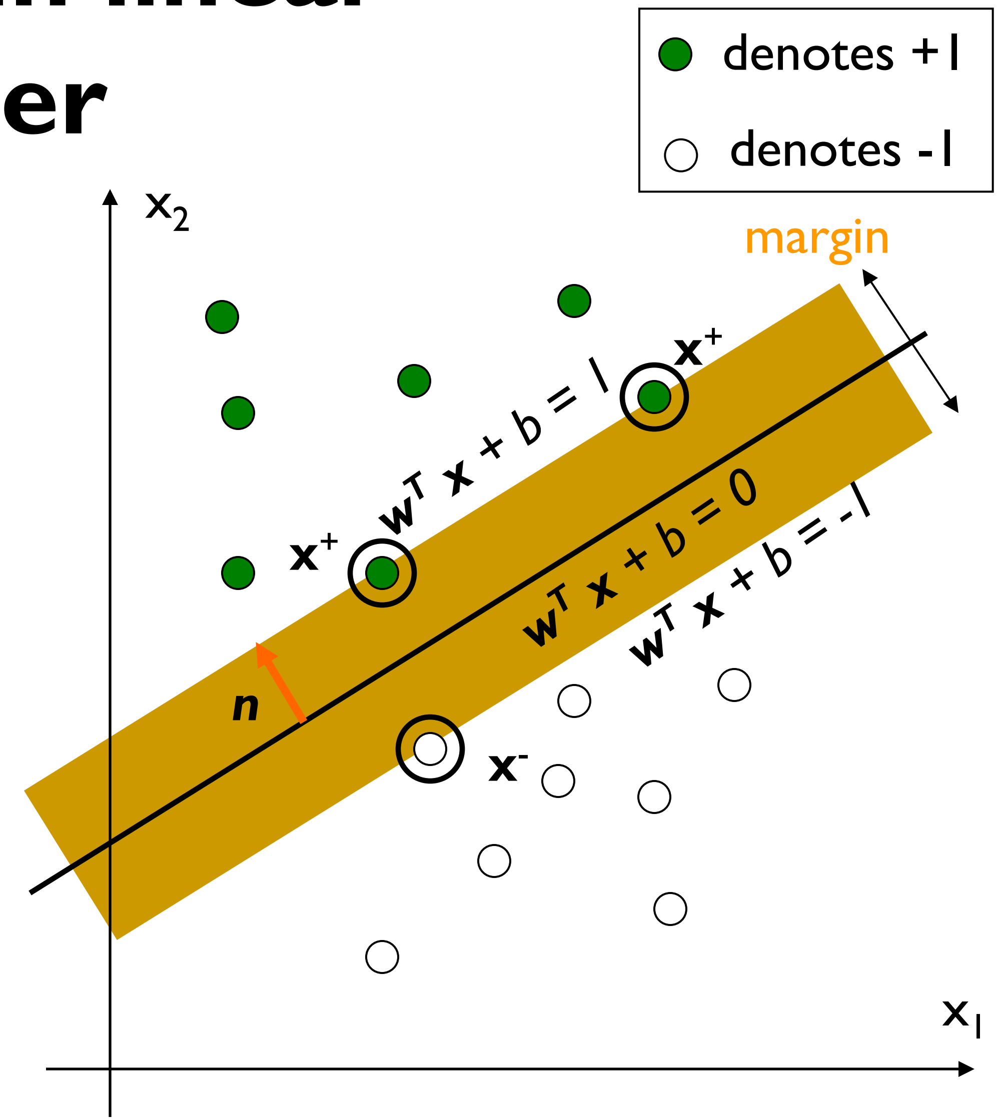
Large margin linear classifier

- Formulation:

$$\text{minimize } \frac{1}{2}\|\boldsymbol{w}\|^2$$

such that

$$y_i(\boldsymbol{w}^T\boldsymbol{x}_i + b) \geq 1$$



Solving the optimization problem

Quadratic programming with linear constraints

$$\text{minimize} \quad \frac{1}{2}\|\mathbf{w}\|^2$$

$$\text{s.t.} \quad y_i(\mathbf{w}^T\mathbf{x}_i + b) \ge 1$$

Lagrangian function

$$\text{minimize} \quad L_p(\mathbf{w}, b, \alpha_i) = \frac{1}{2}\|\mathbf{w}\|^2 - \sum_{i=1}^{n} \alpha_i \left(y_i(\mathbf{w}^T\mathbf{x}_i + b) - 1 \right)$$

$$\text{s.t.} \quad \alpha_i \ge 0$$

Solving the optimization problem

$$\text{minimize} \quad L_p(\mathbf{w}, b, \alpha_i) = \frac{1}{2}\|\mathbf{w}\|^2 - \sum_{i=1}^{n} \alpha_i \left(y_i(\mathbf{w}^T \mathbf{x}_i + b) - 1 \right)$$

$$\text{s.t.} \quad \alpha_i \geq 0$$

$$\frac{\partial L_p}{\partial \mathbf{w}} = 0 \quad \Rightarrow \quad \mathbf{w} = \sum_{i=1}^{n} \alpha_i y_i \mathbf{x}_i$$

$$\frac{\partial L_p}{\partial b} = 0 \quad \Rightarrow \quad \sum_{i=1}^{n} \alpha_i y_i = 0$$

Solving the optimization problem

$$\text{minimize } \; L_p(\mathbf{w}, b, \alpha_i) = \frac{1}{2}\|\mathbf{w}\|^2 - \sum_{i=1}^{n} \alpha_i \left(y_i (\mathbf{w}^T \mathbf{x}_i + b) - 1 \right)$$

s.t. $\alpha_i \geq 0$

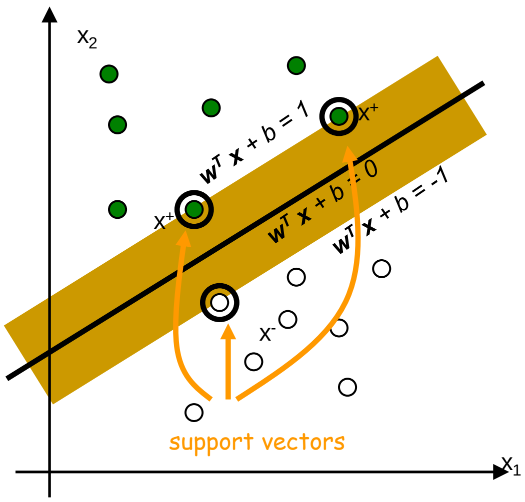
Lagrangian dual problem

$$\text{maximize } \; \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j \mathbf{x}_i^T \mathbf{x}_j$$

s.t. $\alpha_i \geq 0$, and $\sum_{i=1}^{n} \alpha_i y_i = 0$

Solving the optimization problem

- From KKT condition, we know:
  $$\alpha_i(y_i(\boldsymbol{w}^T\boldsymbol{x}_i + b) - 1) = 0$$
- Thus, only support vectors have $\alpha_i \neq 0$
- The solution has the form:
  $$\boldsymbol{w} = \sum_{i=1}^{n} \alpha_i y_i \boldsymbol{x}_i = \sum_{i\in\mathrm{SV}} \alpha_i y_i \boldsymbol{x}_i$$
  get b from $y_i(\boldsymbol{w}^T\boldsymbol{x}_i + b) - 1 = 0$, where x_i is support vector



Solving the optimization problem

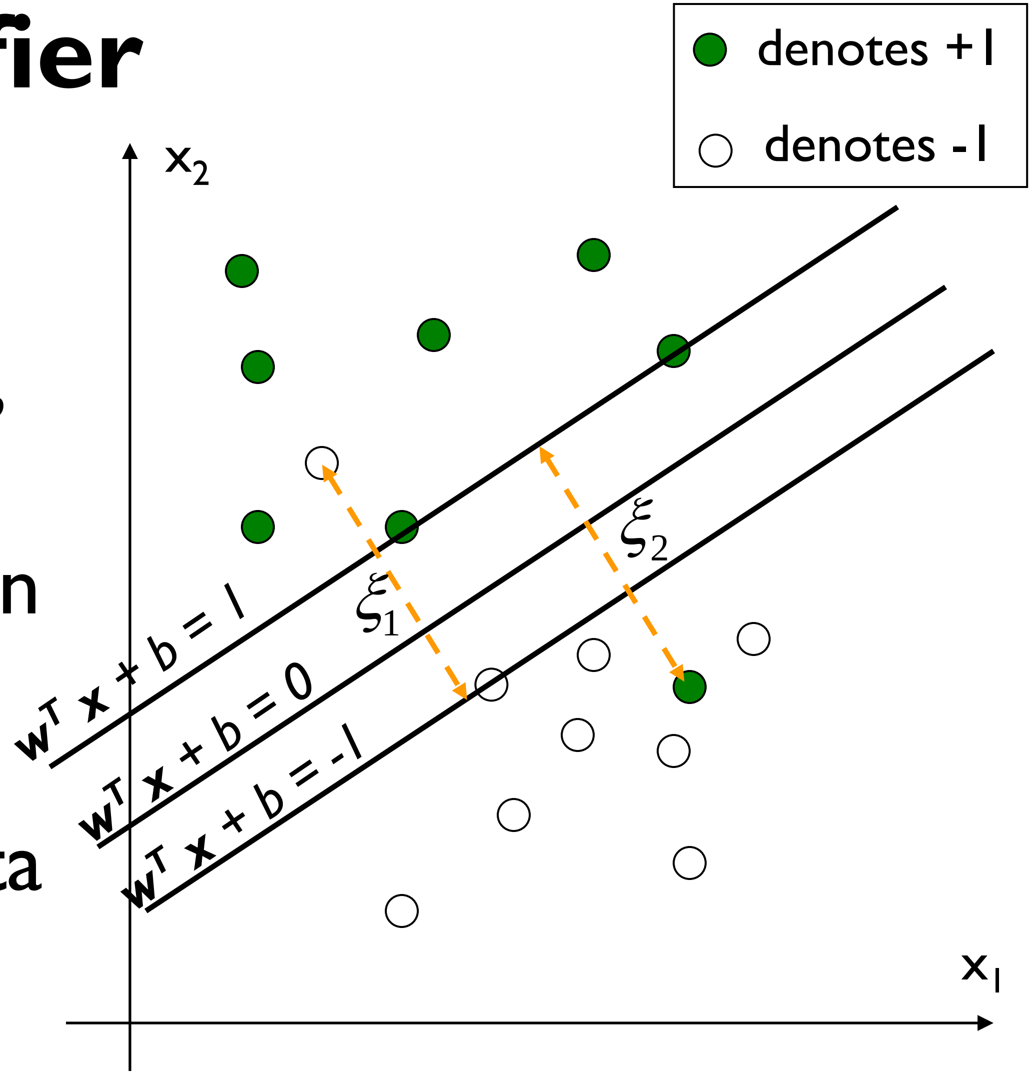
- The linear discriminant function is:

$$g(\boldsymbol{x}) = \boldsymbol{w}^T\boldsymbol{x} + b = \sum_{i \in \mathrm{SV}} \alpha_i \boldsymbol{x}_i{}^T \boldsymbol{x} + b$$

- Notice it relies on a dot product between the test point $\boldsymbol{x}$ and the support vectors $\boldsymbol{x}_i$
- Also keep in mind that solving the optimization problem involved computing the dot products $\boldsymbol{x}_i{}^T\boldsymbol{x}_i$ between all pairs of training points

Soft margin linear classifier

- What if data is not linear separable? (noisy data, outliers, etc.)
- Slack variables ξ_i can be added to allow mis-classification of difficult or noisy data points, resulting margins are called soft.



Soft margin linear classifier

- Formulation:

$$\text{minimize } \frac{1}{2}\|\boldsymbol{w}\|^2 + C\sum_{i=1}^{n}\xi_i$$

  such that

$$y_i(\boldsymbol{w}^T\boldsymbol{x}_i + b) \geq 1 - \xi_i$$
$$\xi_i \geq 0$$

- Parameter C can be viewed as a way to control over-fitting.

Soft margin linear classifier

- Formulation: (Lagrangian dual problem)

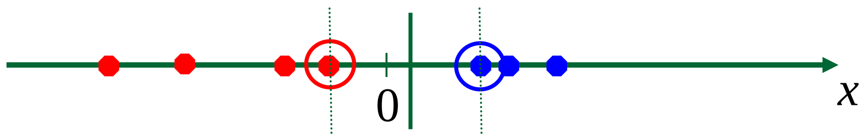
$$\text{maximize} \sum_{i=1}^{n} \alpha_i - \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j \boldsymbol{x}_i{}^T \boldsymbol{x}_j$$

such that

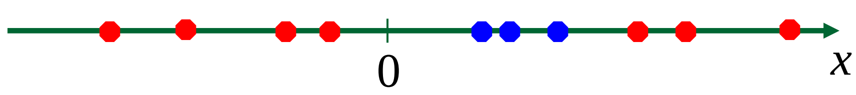
$$0 \le \alpha_i \le C$$
$$\sum\nolimits_{i=1}^{n} \alpha_i y_i = 0$$

Non-linear SVMs

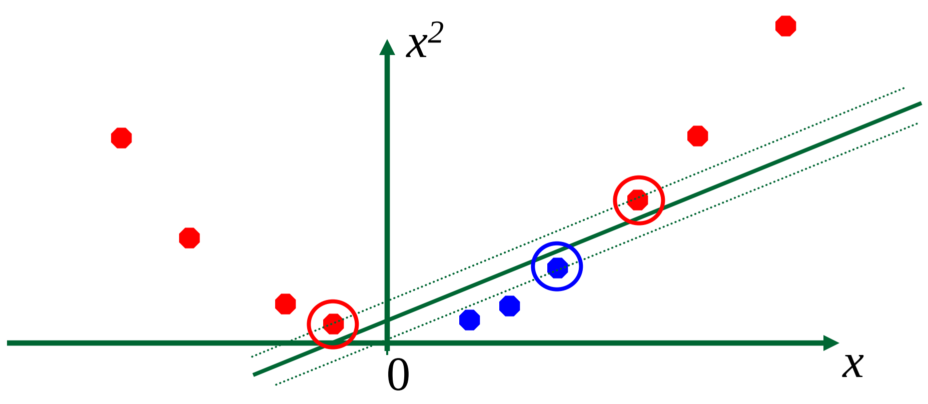
- Datasets that are linearly separable with noise work out great:



- But what are we going to do if the dataset is just too hard?

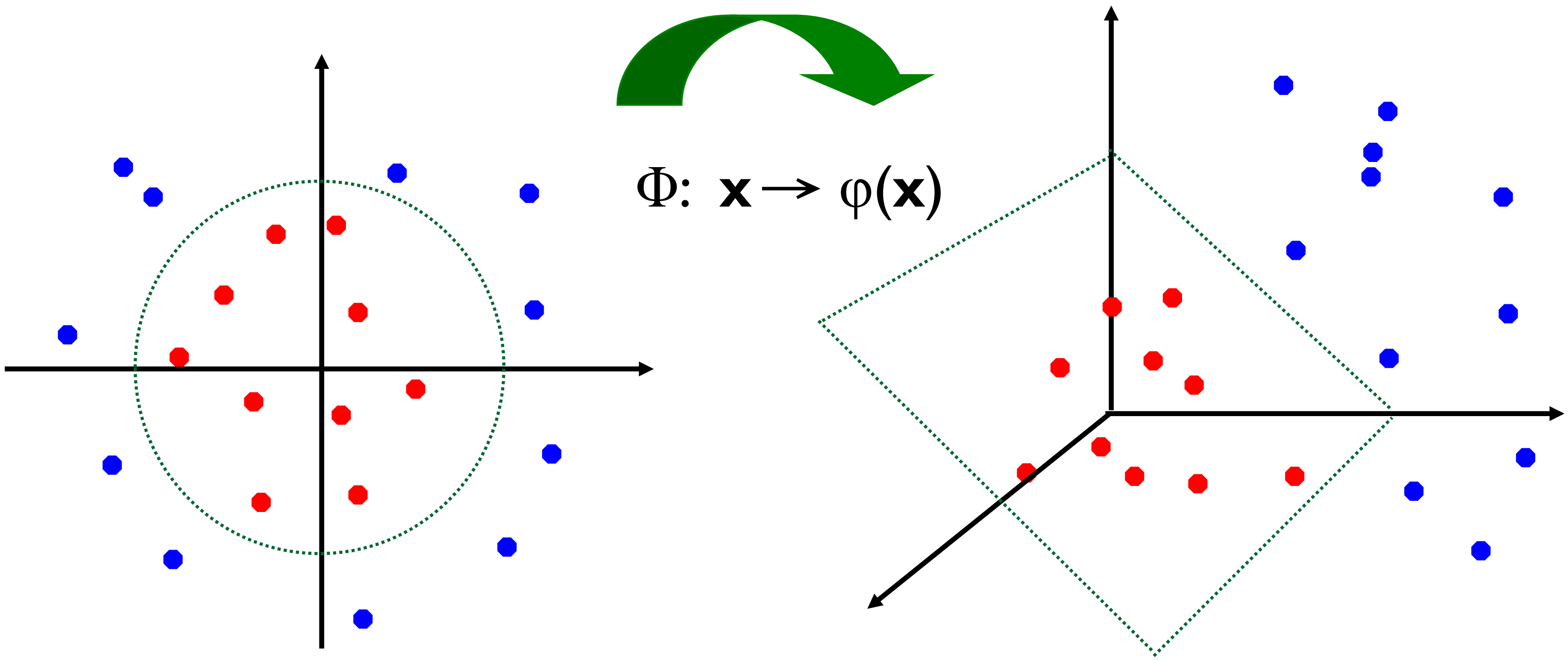


- How about… mapping data to a higher-dimensional space:



Non-linear SVMs: feature space

- General idea: the original input space can be mapped to some higher-dimensional feature space where the training set is separable:



Nonlinear SVMs: the kernel trick

- With this mapping, our discriminant function is now:

$$g(\boldsymbol{x}) = \boldsymbol{w}^T \varphi(\boldsymbol{x}) + b = \sum_{i \in \mathrm{SV}} \alpha_i \boxed{\varphi(\boldsymbol{x}_i)^T \varphi(\boldsymbol{x})} + b$$

- No need to know this mapping explicitly, because we only use the dot product of feature vectors in both the training and test.
- A kernel function is defined as a function that corresponds to a dot product of two feature vectors in some expanded feature space:

$$K(\boldsymbol{x}_i, \boldsymbol{x}_j) \equiv \varphi(\boldsymbol{x}_i)^T \varphi(\boldsymbol{x}_j)$$

Nonlinear SVMs: the kernel trick

- An example:

2-dimensional vectors $\mathbf{x}=[x_1 \;\; x_2]$;

let $K(\mathrm{x_i},\mathrm{x_j})=(1 + \mathrm{x_i}^T\mathrm{x_j})^2$,

Need to show that $K(\mathrm{x_i},\mathrm{x_j}) = \varphi(\mathrm{x_i})^T\varphi(\mathrm{x_j})$:

$$K(\mathrm{x_i},\mathrm{x_j})=(1 + \mathrm{x_i}^T\mathrm{x_j})^2$$

$$=1+ x_{i1}^2x_{j1}^2 + 2\, x_{i1}x_{j1}\, x_{i2}x_{j2}+ x_{i2}^2x_{j2}^2 + 2x_{i1}x_{j1} + 2x_{i2}x_{j2}$$

$$=[1 \;\; x_{i1}^2 \;\; \sqrt{2}x_{i1}x_{i2} \;\; x_{i2}^2 \;\; \sqrt{2}x_{i1} \;\; \sqrt{2}x_{i2}]^T [1 \;\; x_{j1}^2 \;\; \sqrt{2}x_{j1}x_{j2} \;\; x_{j2}^2 \;\; \sqrt{2}x_{j1} \;\; \sqrt{2}x_{j2}]$$

$$= \varphi(\mathrm{x_i})^T\varphi(\mathrm{x_j}), \quad \text{where } \varphi(\mathrm{x}) = [1 \;\; x_1^2 \;\; \sqrt{2}\, x_1x_2 \;\; x_2^2 \;\; \sqrt{2}x_1 \;\; \sqrt{2}x_2]$$

Nonlinear SVMs: the kernel trick

- Examples of commonly-used kernel functions:
  - Linear kernel: $K(\boldsymbol{x}_i, \boldsymbol{x}_j) = \boldsymbol{x}_i{}^T \boldsymbol{x}_j$
  - Polynomial kernel: $K(\boldsymbol{x}_i, \boldsymbol{x}_j) = (1 + \boldsymbol{x}_i{}^T \boldsymbol{x}_j)^{\,p}$
  - Gaussian kernel: $K(\boldsymbol{x}_i, \boldsymbol{x}_j) = \exp(-\frac{\|\boldsymbol{x}_i - \boldsymbol{x}_j\|^2}{2\sigma^2})$
  - Sigmoid: $K(\boldsymbol{x}_i, \boldsymbol{x}_j) = \tanh(\beta_0 \boldsymbol{x}_i{}^T \boldsymbol{x}_j + \beta_1)$
- In general, functions that satisfy Mercer's condition can be kernel functions.

Nonlinear SVM: optimization

- Formulation: (Lagrangian dual problem)

$$\text{maximize } \sum_{i=1}^{n} \alpha_i - \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j K(\boldsymbol{x}_i, \boldsymbol{x}_j)$$

such that

$$0 \le \alpha_i \le C$$

$$\sum\nolimits_{i=1}^{n} \alpha_i y_i = 0$$

- The solution of the discriminant function is

$$g(\boldsymbol{x}) = \sum\nolimits_{i \in \mathrm{SV}} \alpha_i \, K(\boldsymbol{x}_i, \boldsymbol{x}) + b$$

- The optimization technique is the same.

SVM: algorithm

1. Choose a kernel function
2. Choose a value for C
3. Solve the quadratic programming problem (many software packages available)
4. Construct the discriminant function from the support vectors

Some issues

- Choice of kernel
  - Gaussian or polynomial kernel is default
  - if ineffective, more elaborate kernels are needed
  - domain experts can give assistance in formulating appropriate similarity measures
- Choice of kernel parameters
  - e.g., σ in Gaussian kernel
  - σ is the distance between closest points with different classifications
  - In the absence of reliable criteria, applications rely on the use of a validation set or cross-validation to set such parameters.

Some issues

- Optimization criterion – Hard margin v.s. Soft margin
  - a lengthy series of experiments in which various parameters are tested
- Scaling before applying SVM is very important.
  - The main advantage of scaling is to avoid attributes in greater numeric ranges dominating those in smaller numeric ranges.
  - Another advantage is to avoid numerical difficulties during the calculation.
  - Of course we have to use the same method to scale both training and testing data.

Artificial neural networks

These slides are courtesy of 虞台文
http://aimm02.cse.ttu.edu.tw/class_2004_2/ANNs/Lecture1---Introduction%20to%20ANN.ppt

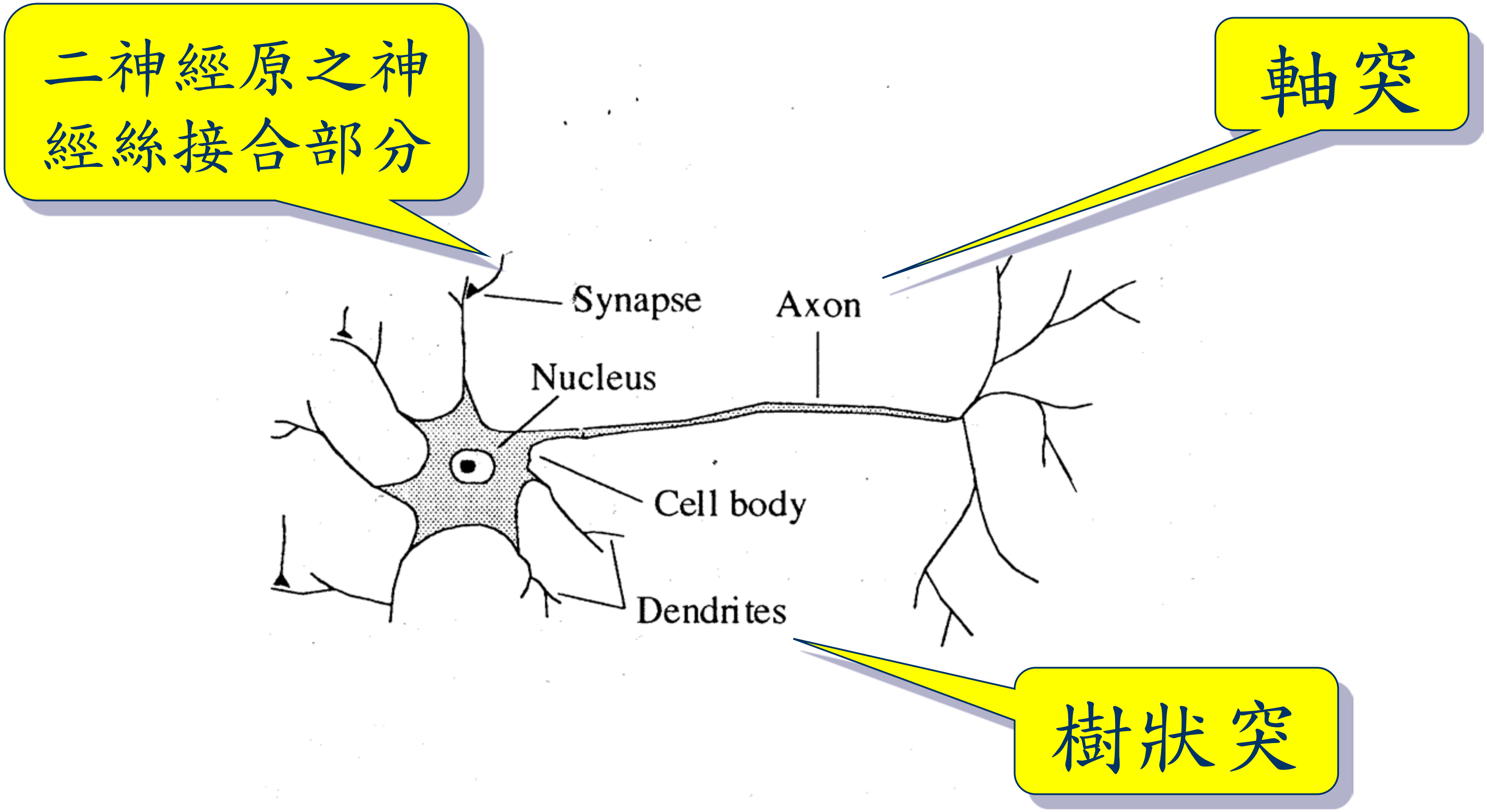
Artificial neural networks (ANNs)

- To simulate human brain behavior
- A new generation of information processing system

Configuration of ANNs

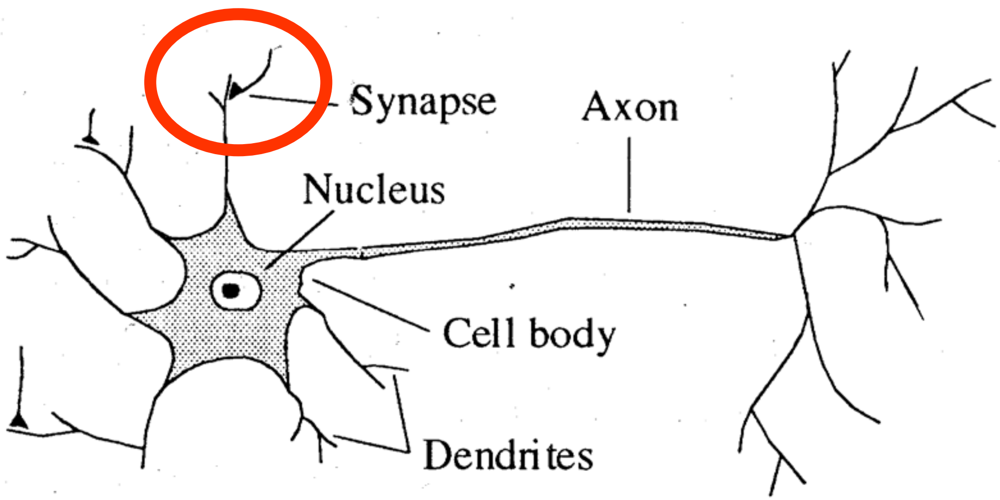
- An ANN consists of a large number of interconnected processing elements called neurons.
  - A human brain consists of ~10^{11} neurons of many different types.
- How ANN works?
  - Collective behavior

The biologic neuron

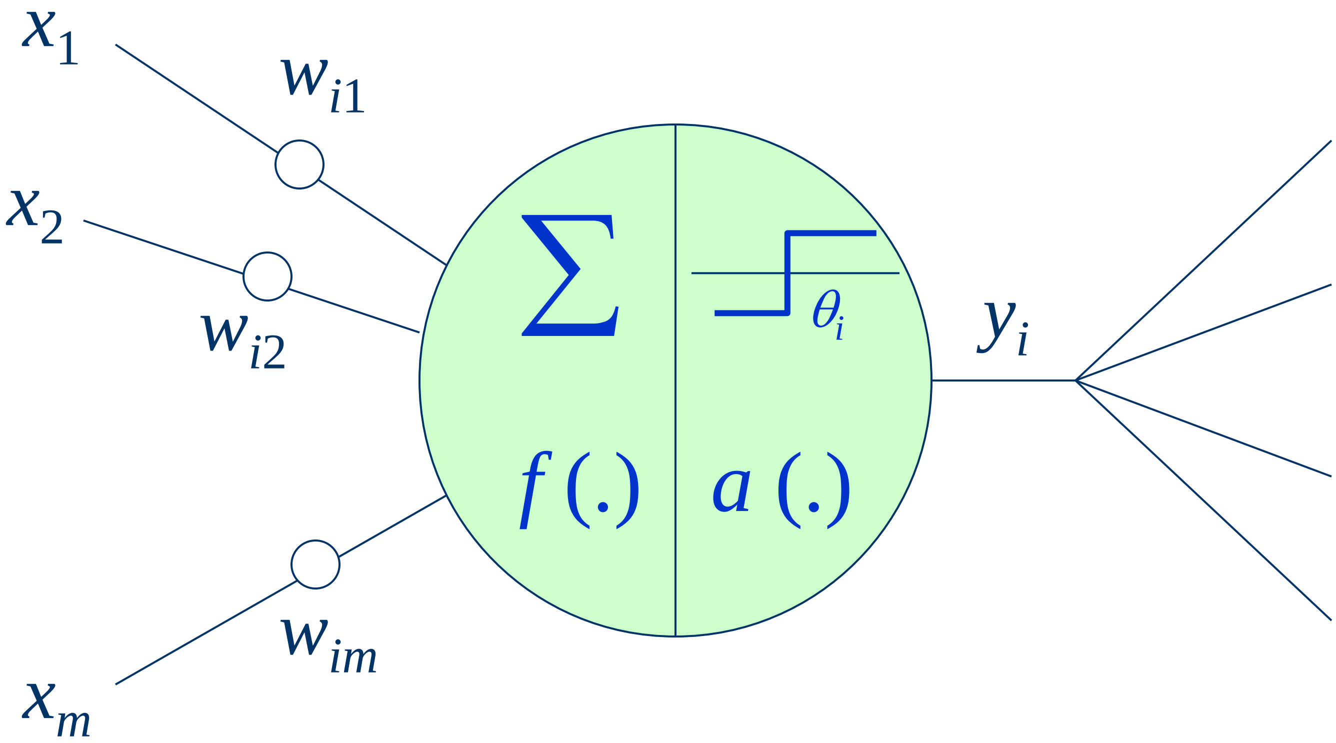


The biologic neuron

Excitatory or Inhibitory

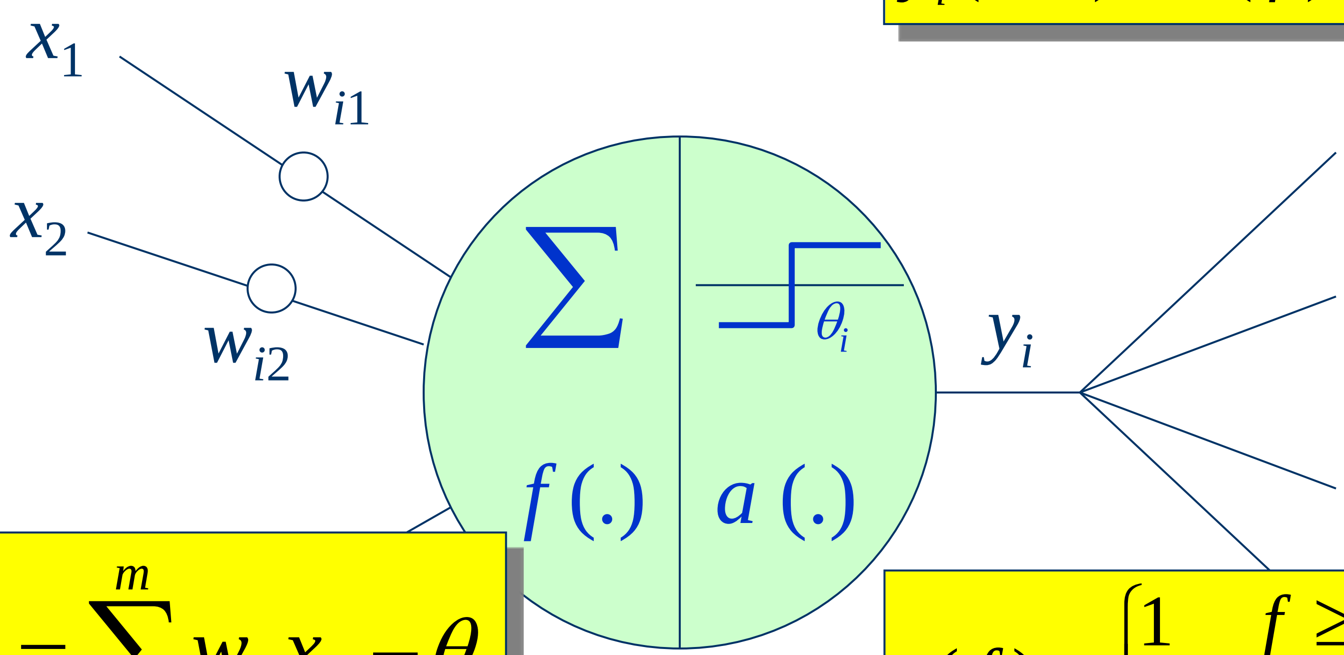


The artificial neuron



The artificial neuron

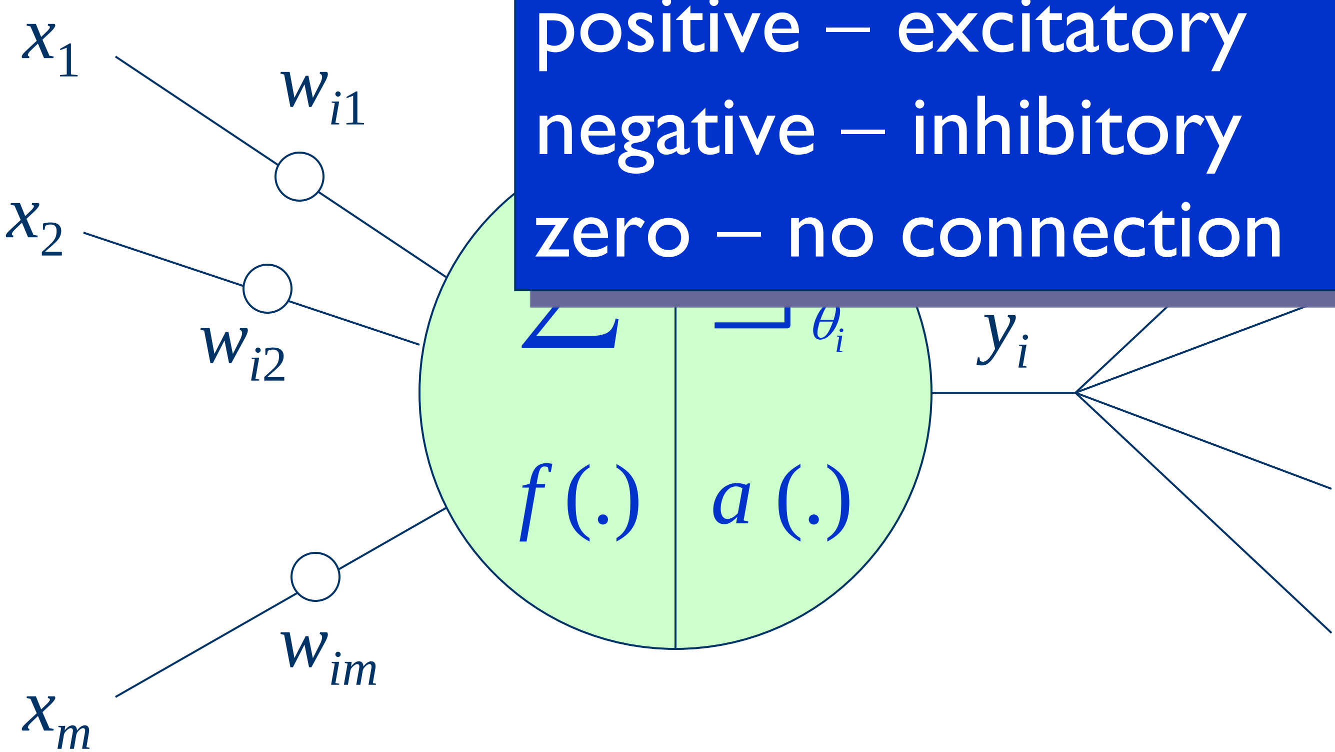
$$y_i(t+1) = a(f)$$



$$f(\mu_i) = \sum_{j=1}^{m} w_{ij} x_j - \theta_i$$

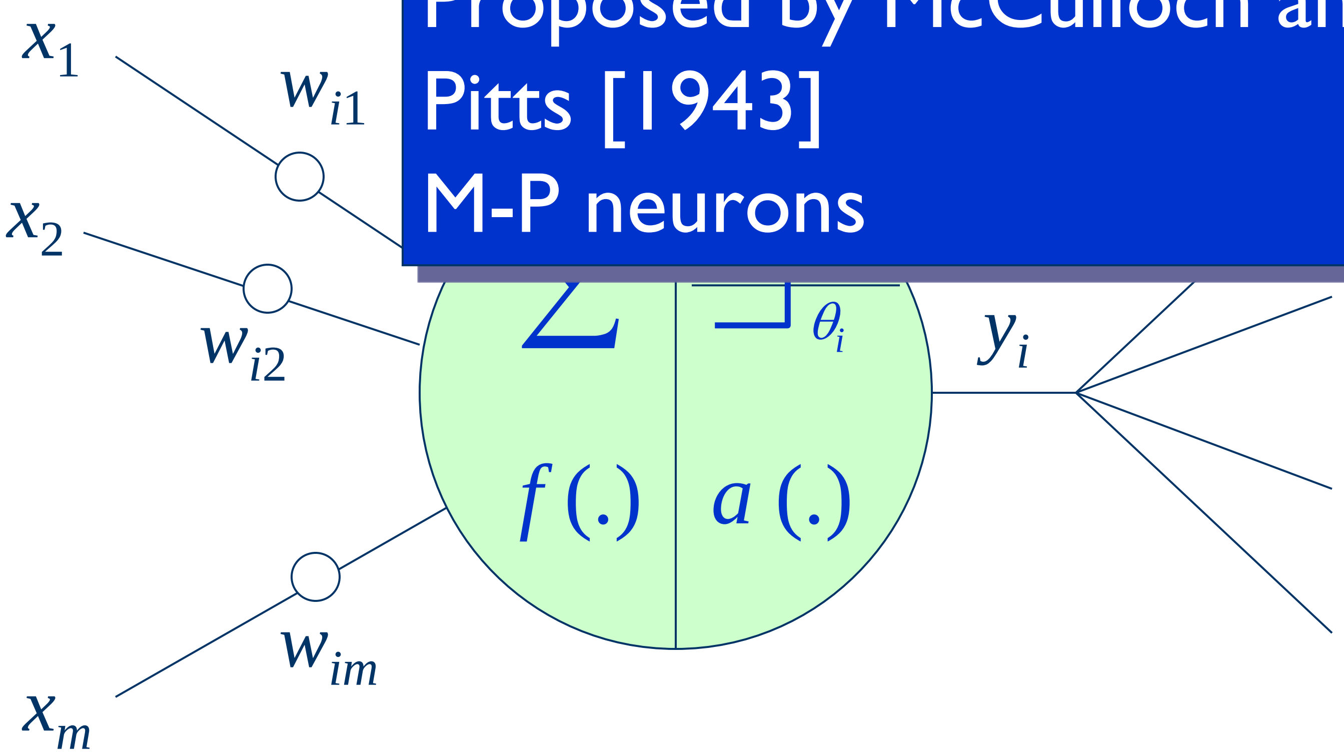
$$a(f) = \begin{cases} 1 & f \geq 0 \\ 0 & otherwise \end{cases}$$

The artificial neuron



w_{ij}
positive – excitatory
negative – inhibitory
zero – no connection

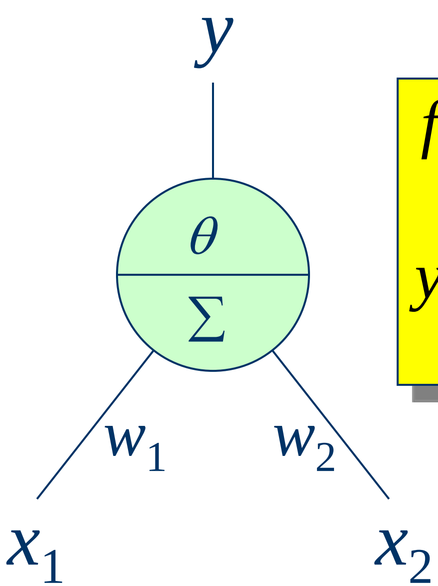
The artificial neuron



Proposed by McCulloch and Pitts [1943]
M-P neurons

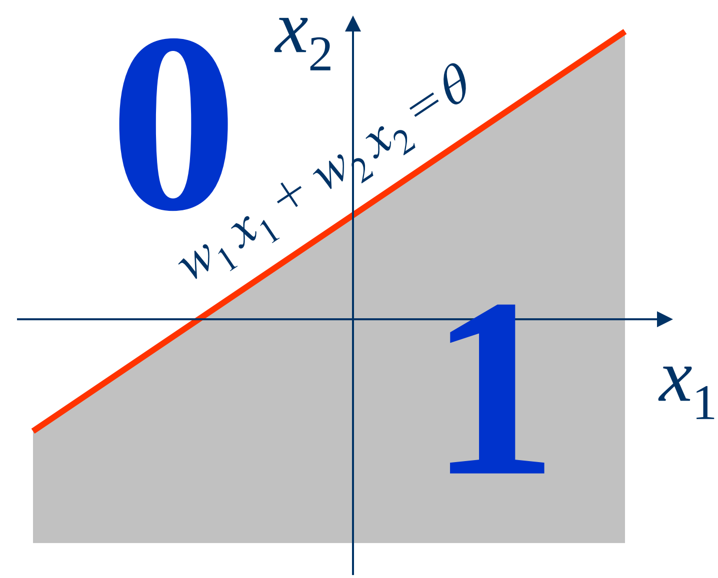
What can be done by M-P neurons?

- A hard limiter.
- A binary threshold unit.
- Hyperspace separation.



$$f(\mu) = w_1 x_1 + w_2 x_2 - \theta$$

$$y = \begin{cases} 1 & f(\mu) \geq 0 \\ 0 & otherwise \end{cases}$$



What ANNs will be?

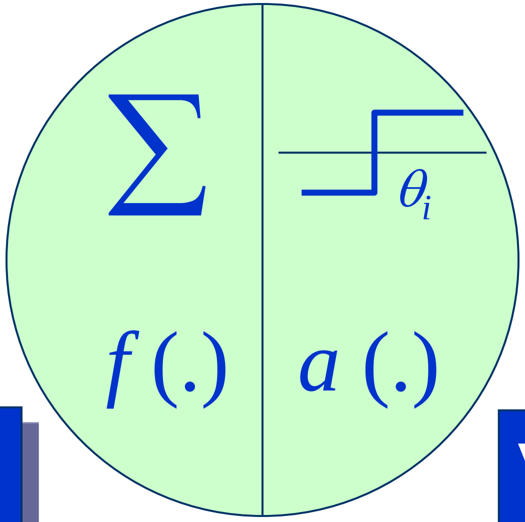
- ANN -- A neurally inspired mathematical model.
- Consists a large number of highly interconnected PEs.
- Its connections (weights) holds knowledge.
- The response of PE depends only on local information.
- Its collective behavior demonstrates the computation power.
- With learning, recalling and, generalization capability.

Three basic entities of ANN models

- Models of neurons
- Models of synaptic interconnections and structures
- Training or learning rules

Neuron models

Extensions of M-P neurons



What integration functions we may have?

What activation functions we may have?

Integration functions

M-P neuron

$$f_i \equiv net_i = \sum_{j=1}^{m} w_{ij} x_j - \theta_i$$

Quadratic Function

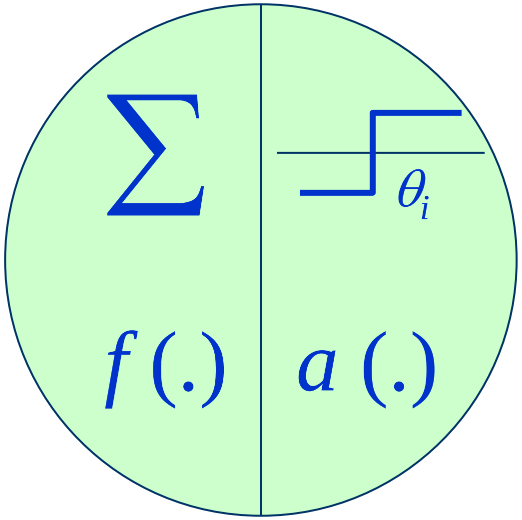
$$f_i = \sum_{j=1}^{m} w_{ij} x_j^2 - \theta_i$$

Spherical Function

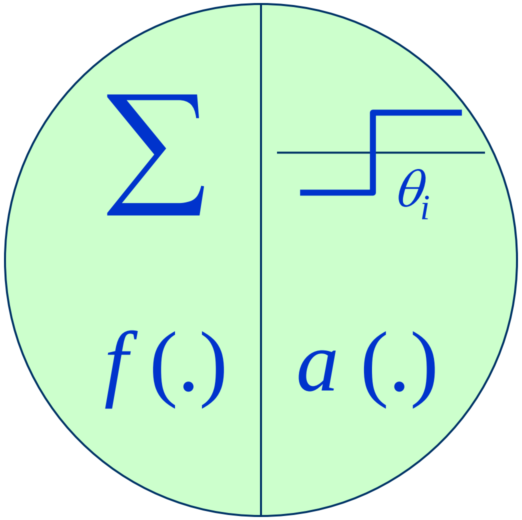
$$f_i = \sum_{j=1}^{m} (x_j - w_{ij})^2 - \theta_i$$

Polynomial Function

$$f_i = \sum_{j=1}^{m} \sum_{k=1}^{m} \left(w_{ijk} x_j x_k + x_j^{\alpha_j} + x_k^{\alpha_k} \right) - \theta_i$$

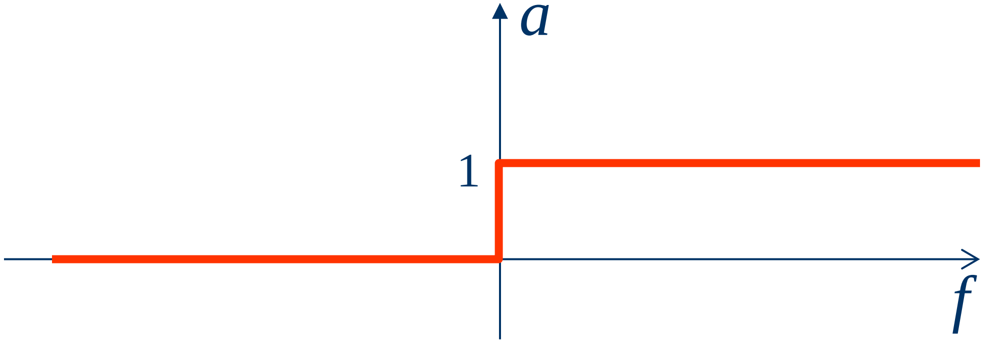


Activation functions



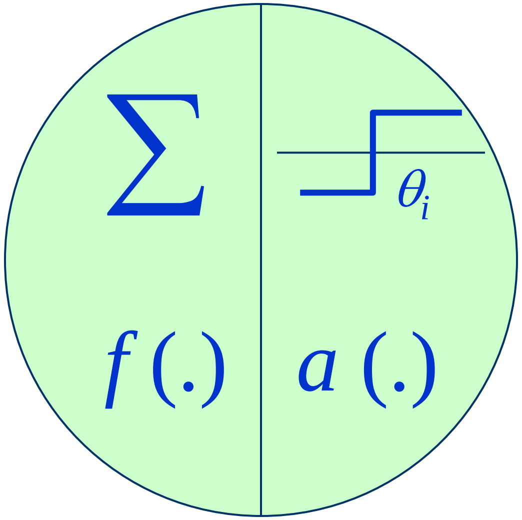
M-P neuron: (Step function)

$$a(f) = \begin{cases} 1 & f \geq 0 \\ 0 & otherwise \end{cases}$$

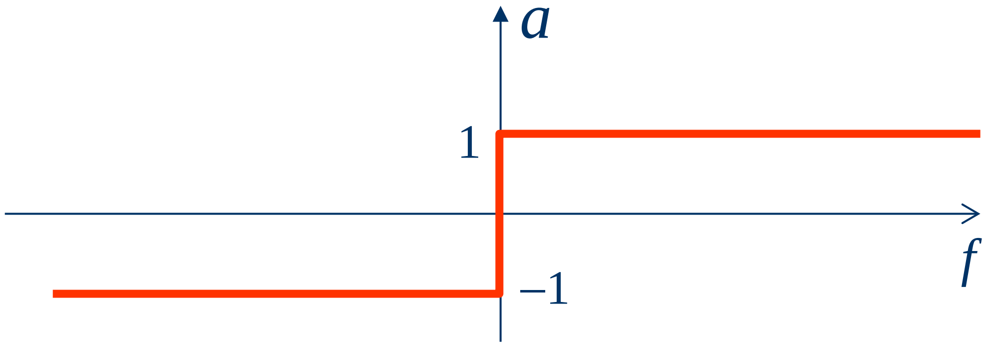


Activation functions

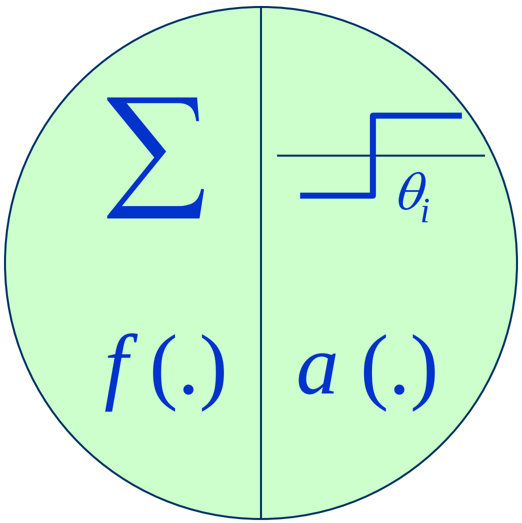
Hard Limiter (Threshold function)



$$a(f) = \mathrm{sgn}(f) = \begin{cases} 1 & f \geq 0 \\ -1 & f < 0 \end{cases}$$

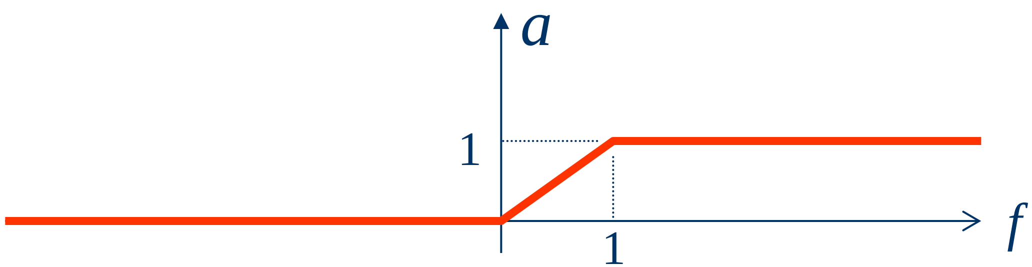


Activation functions

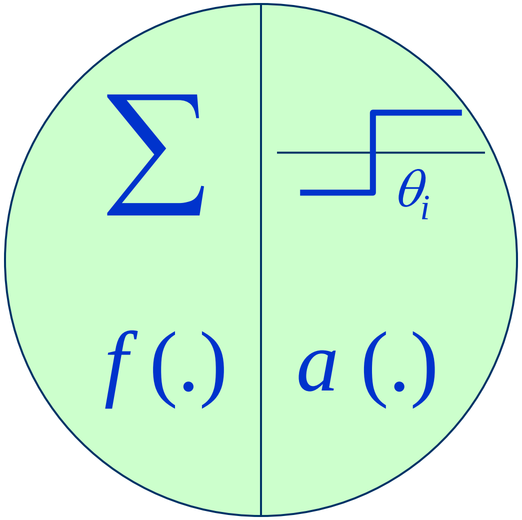


Ramp function:

$$a(f) = \begin{cases} 1 & f > 1 \\ f & 0 \le f \le 1 \\ 0 & f < 0 \end{cases}$$

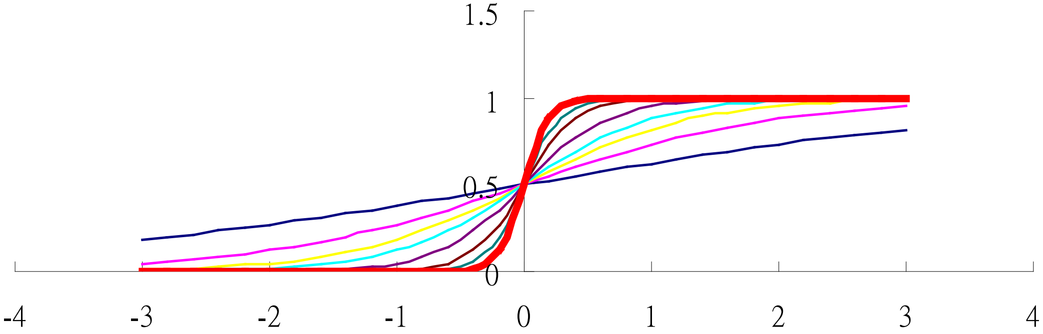


Activation functions

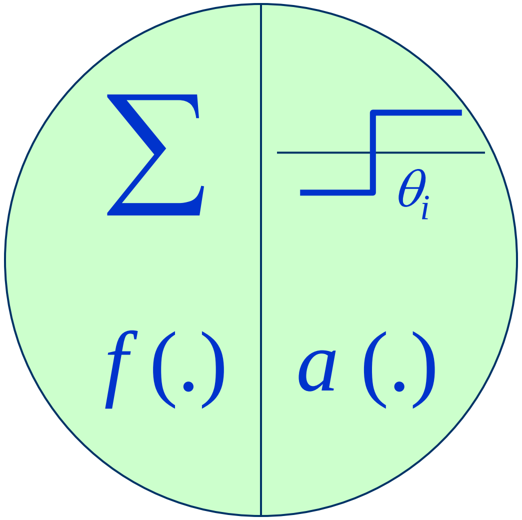


Unipolar sigmoid function:

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

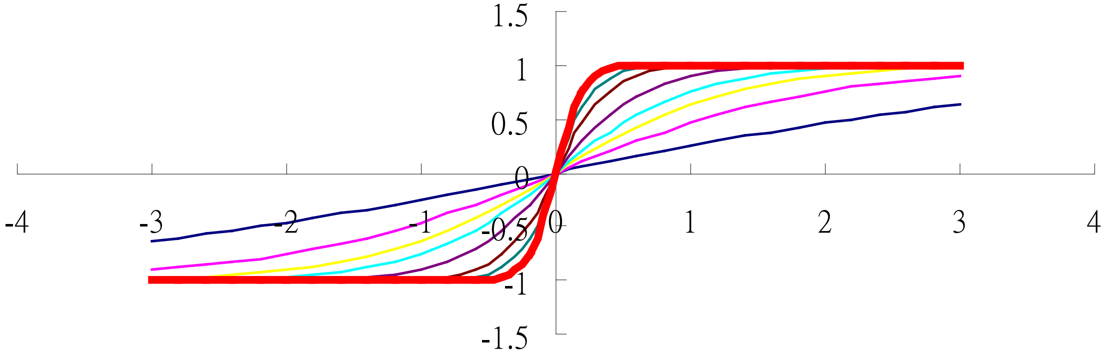


Activation functions

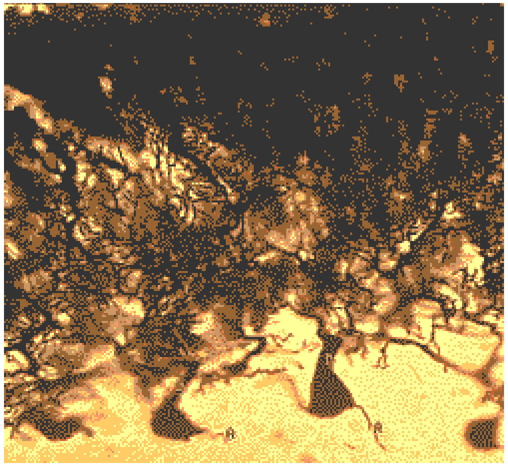


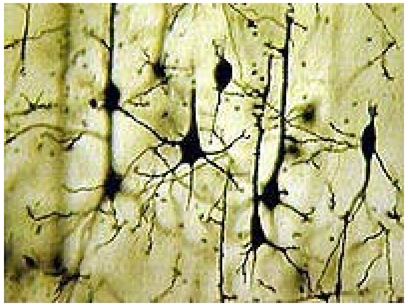
Bipolar sigmoid function:

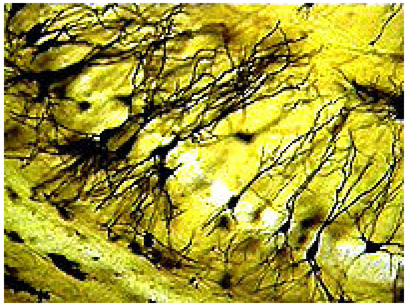
$$a(f) = \frac{2}{1+e^{-\lambda f}} - 1$$



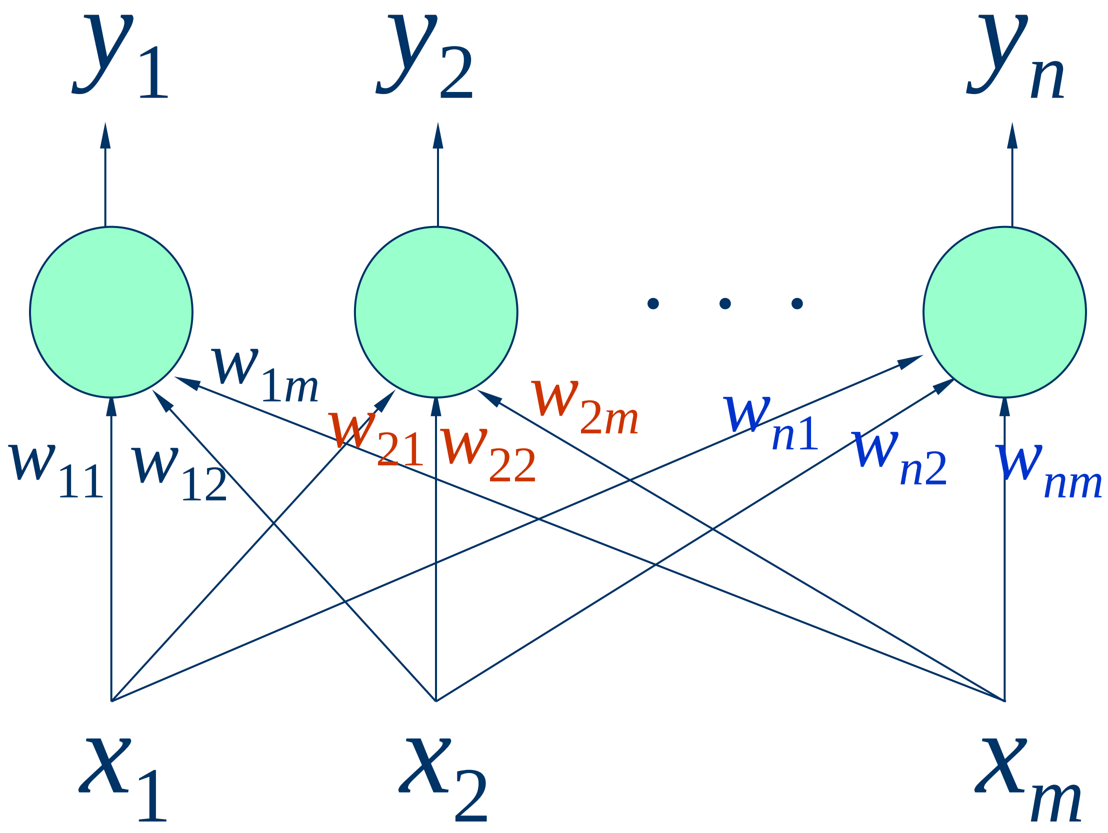
ANN structure (connections)



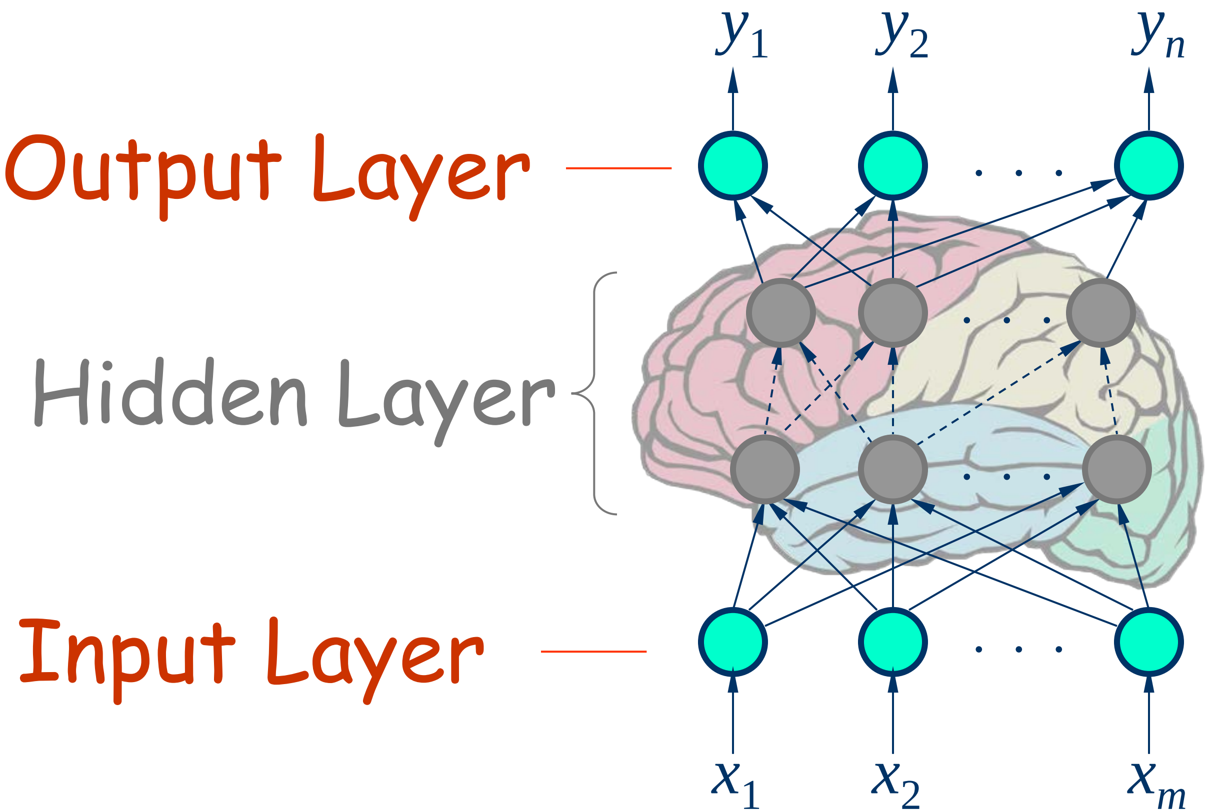




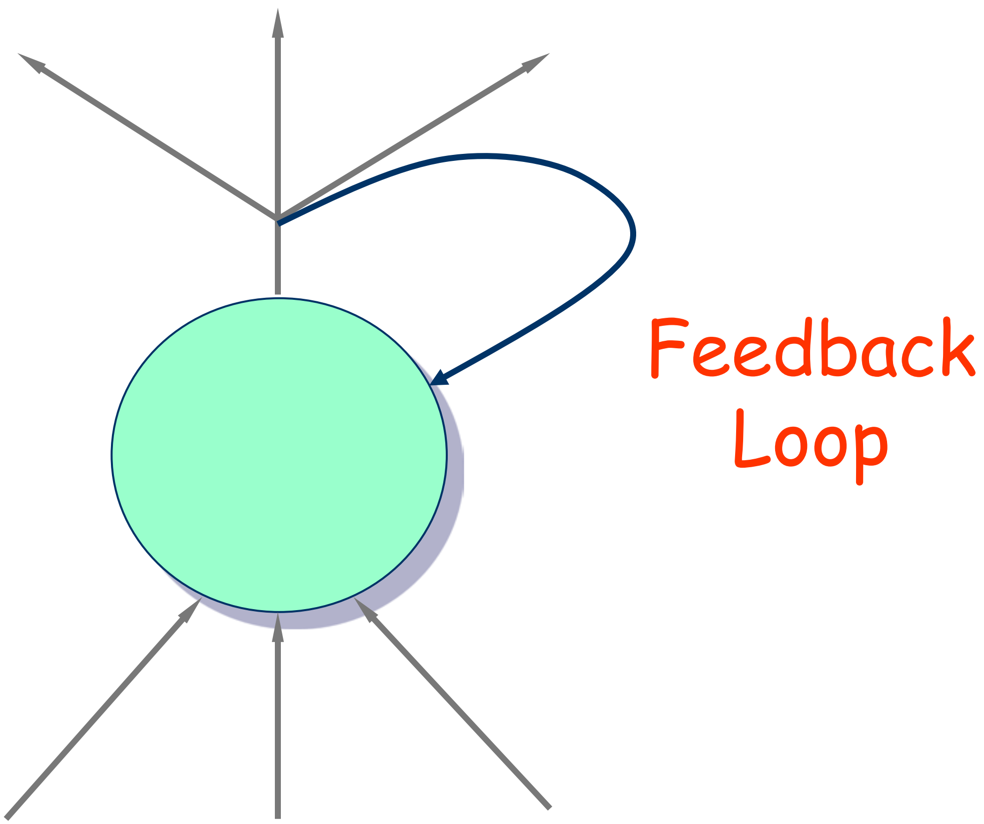
Single-layer feedforward networks



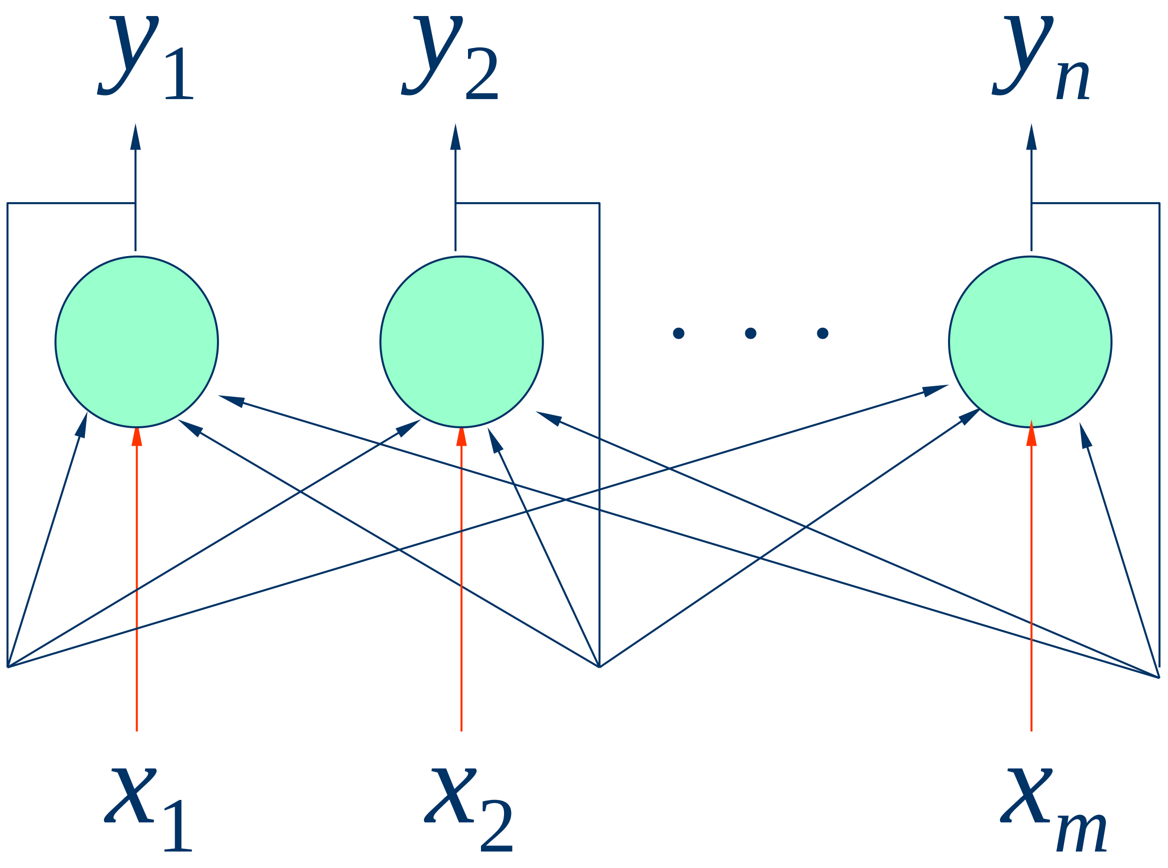
Multilayer feedforward networks



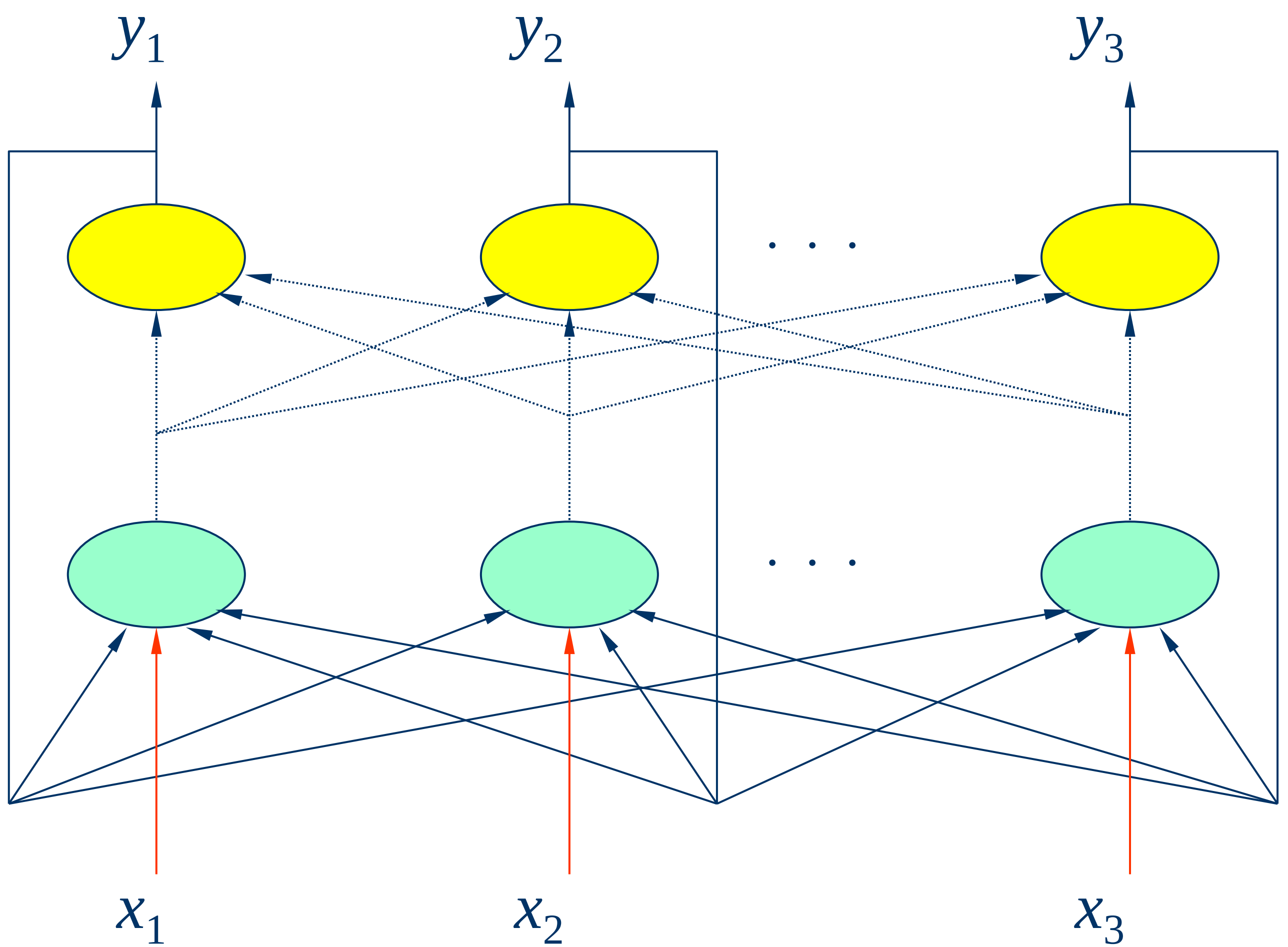
Single node with feedback to itself



Single-layer recurrent networks



Multilayer recurrent networks



Learning

- Consider an ANN with *n* neurons and each with *m* adaptive weights.
- Weight matrix:

$$\mathbf{W} = \begin{bmatrix} \mathbf{w}_1^T \\ \mathbf{w}_2^T \\ \vdots \\ \mathbf{w}_n^T \end{bmatrix} = \begin{bmatrix} w_{11} & w_{12} & \cdots & w_{1m} \\ w_{21} & w_{22} & \cdots & w_{2m} \\ \vdots & \vdots & \ddots & \vdots \\ w_{n1} & w_{n2} & \cdots & w_{nm} \end{bmatrix}$$

- Note: θ_i can be viewed as a weight with $x_i = 1$

Learning How?

- Consider an ANN with *n* neurons and each with *m* adaptive weights.
- W

To "learn" the weight matrix.

$$\mathbf{W} = \begin{bmatrix} \mathbf{w}_1^T \\ \mathbf{w}_2^T \\ \vdots \\ \mathbf{w}_n^T \end{bmatrix} = \begin{bmatrix} w_{11} & w_{12} & \cdots & w_{1m} \\ w_{21} & w_{22} & \cdots & w_{2m} \\ \vdots & \vdots & \ddots & \vdots \\ w_{n1} & w_{n2} & \cdots & w_{nm} \end{bmatrix}$$

- Note: θ_i can be viewed as a weight with $x_i = 1$

Learning rules

- Supervised learning
- Reinforcement learning
- Unsupervised learning

Example: 部落格分類

	A	B	C	D	E	F	G
1	blogid	commentscount	articlescount	usagedays	subscribercount	cateid	
2	1	15648	2756	2638	161	0	
3	2	18876	3075	3545	2254	0	
4	3	24614	1465	229	417	0	
5	4	18426	583	1494	101	0	
6	5	15854	1390	2639	757	0	
7	6	14624	1856	2285	487	0	
8	7	7608	1422	2374	1375	0	
9	8	2866	789	2500	657	0	
10	9	2973	1118	1591	1134	0	
11	10	8320	366	979	3650	0	
12	11	11339	1822	1453	99	0	
13	12	7269	1374	2707	805	0	
14	13	12138	2318	2566	50	0	
15	14	2657	565	1791	916	0	
16	15	1654	1795	2519	385	0	

(RMD_example 16.1)

Variable	Description
blogid	部落格 ID
commentscount	累積留言數
articlescount	總發表文章數
usagedays	使用痞客邦天數
subscribercount	訂閱數
cateid	部落格分類編號：0=美食情報，1=休閒旅遊，2=職場甘苦

部落格分類

- Support vector machine
  - R: library e1071, function svm
  - RMD_example 16.2
- Artificial neural networks
  - R: library neuralnet, function neuralnet
  - RMD_example 16.3