UCB - CS189
Introduction to Machine Learning
Fall 2015
Lecture 3: Support Vector Machines

Isabelle Guyon

ChaLearn

Come to my office hours…

Wed 2:30-4:30 Soda 329

Last time

- We represent patterns as vectors **x** in a space of d dimensions.
- A "discriminant function" f(**x**) is a function such that f(**x**) > 0 for one class and f(**x**) < 0 for the other. f(**x**)=0 is the equation of the decision boundary.
- Given a weight vector **w**, f(**x**)=**w.x** is a linear discriminant function. The corresponding decision boundary **w.x**=0 is a hyperplane (a subspace of dimension (d-1).
- Feature transforms **x** -> Φ(**x**) permit to built non-linear decision boundaries, while using discriminant linear in **w** (NOT in x).

Come to my office hours…
Wed 2:30-4:30 Soda 329

Today

- With linear threshold units ("neurons") we can build:
  - Linear discriminant
  - Kernel methods
  - Neural networks

DUAL

PARAMETRIC

NON PARAMETRIC

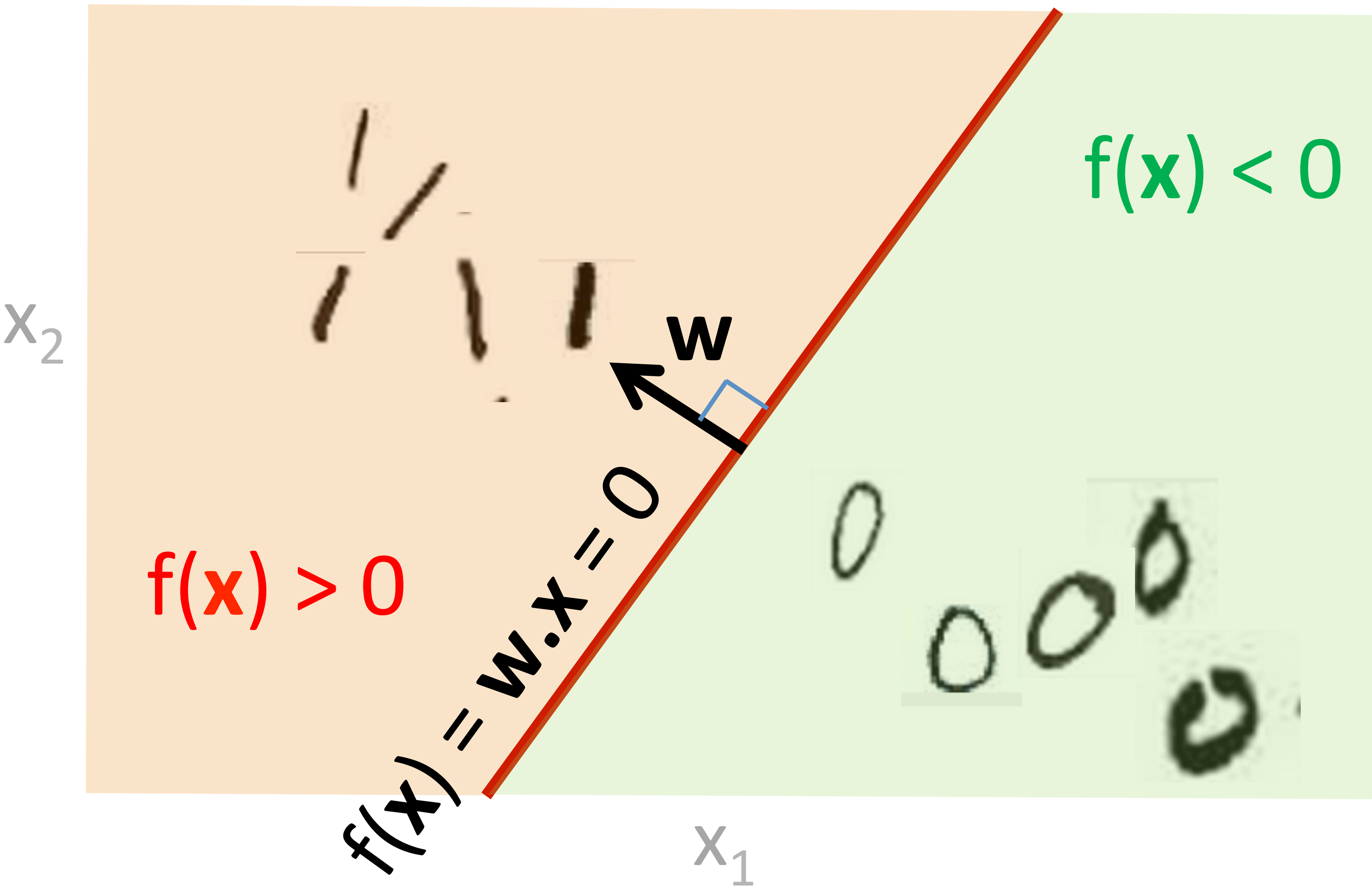
- The architectural hyper-parameters may include:
  - The choice of basis functions ϕ (features)
  - The kernel
  - The number of hidden units.

Control the COMPLEXITY

- "Complex" models are prone to overfitting.

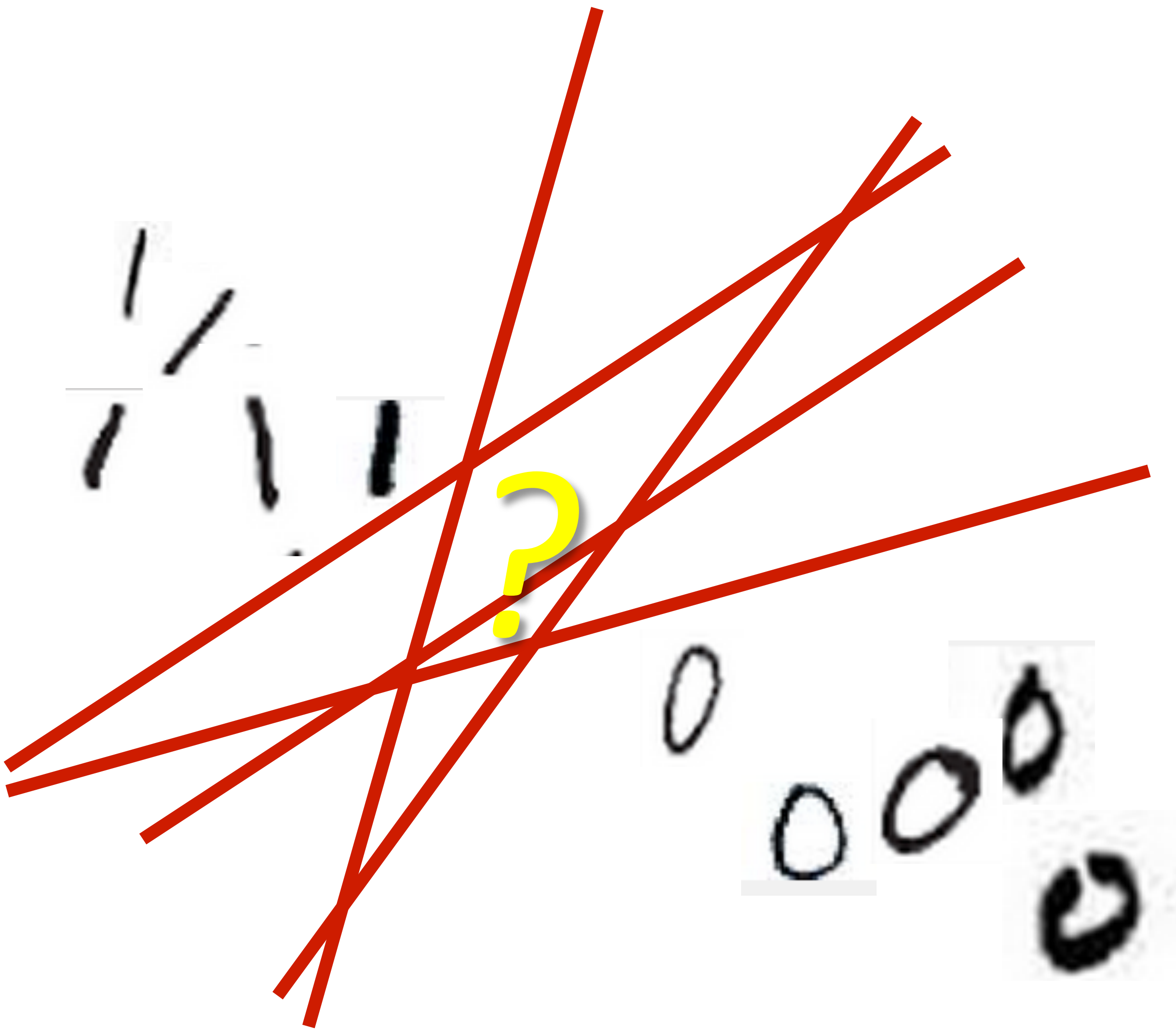
LAST TIME

Separating hyperplane

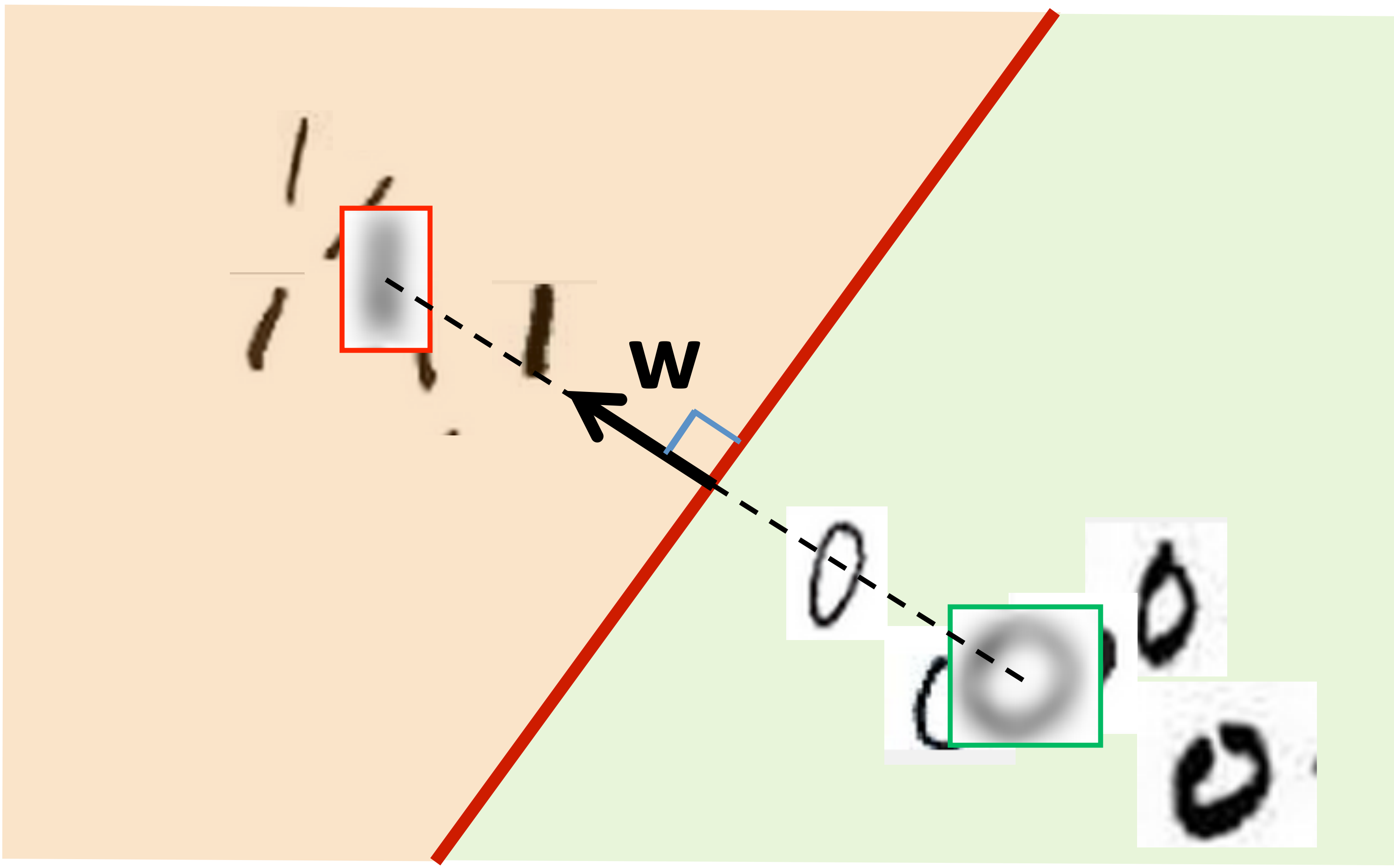


LAST TIME

Separating hyperplane

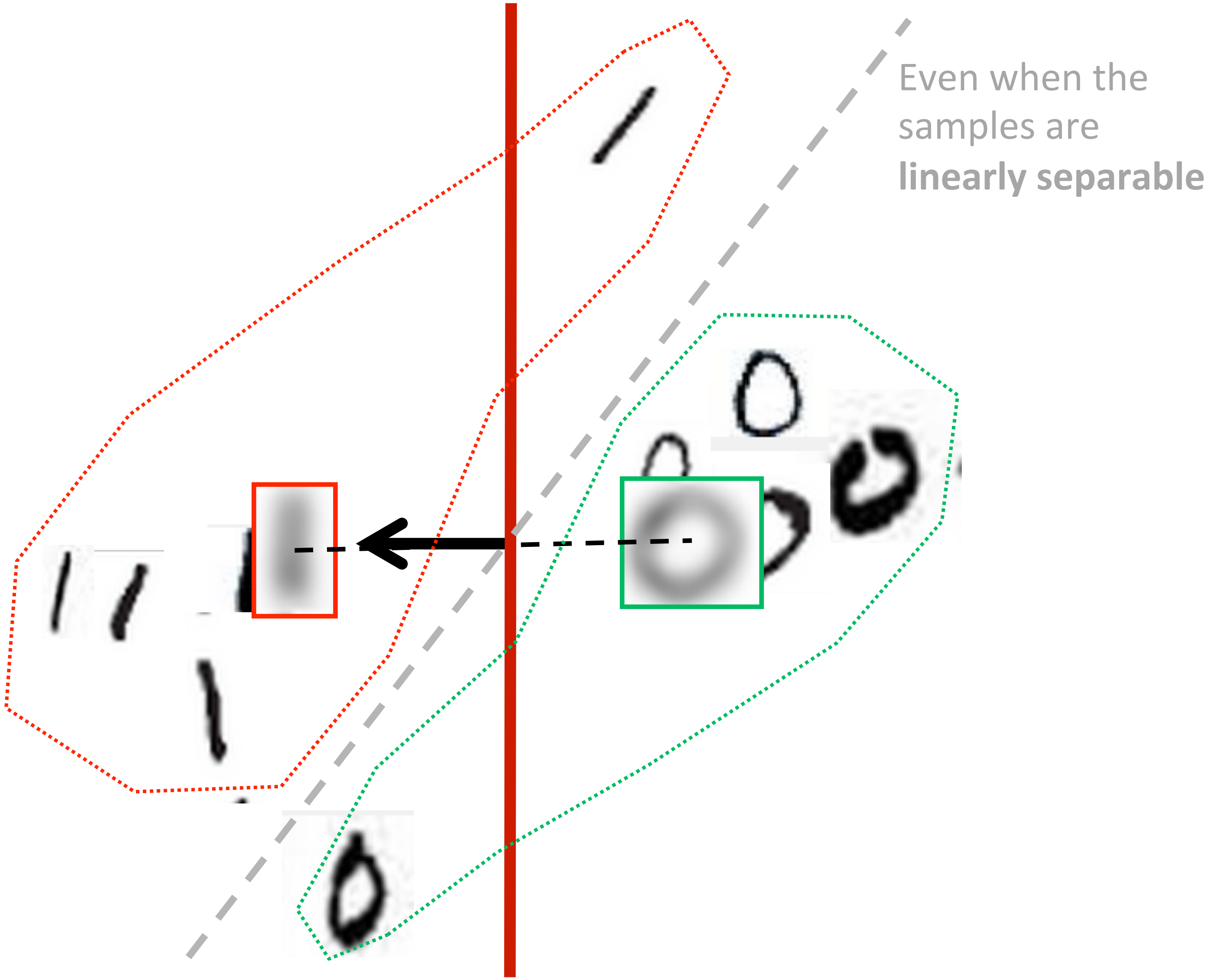


Centroid method

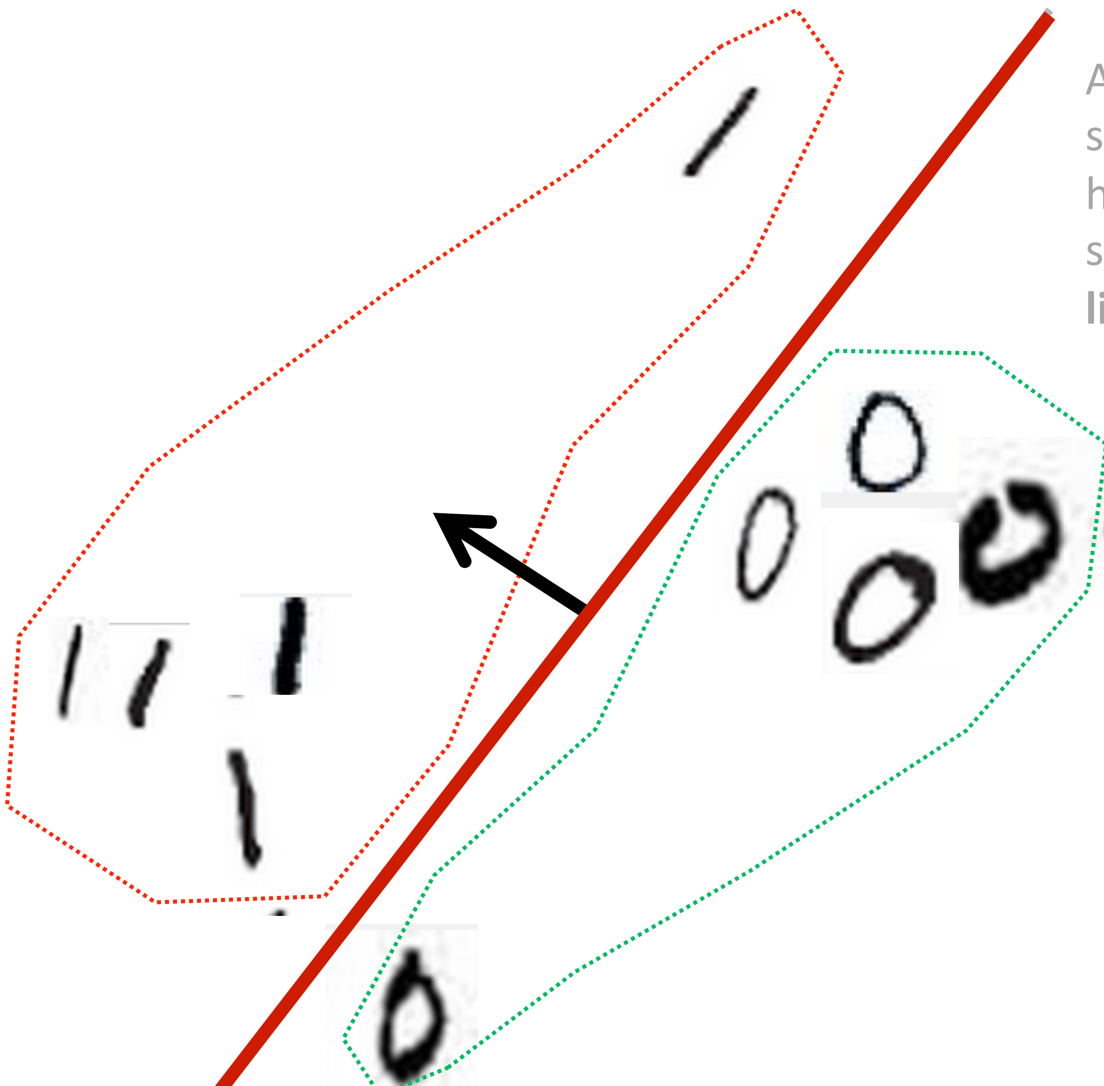


LAST TIME

The centroid methods can fail!

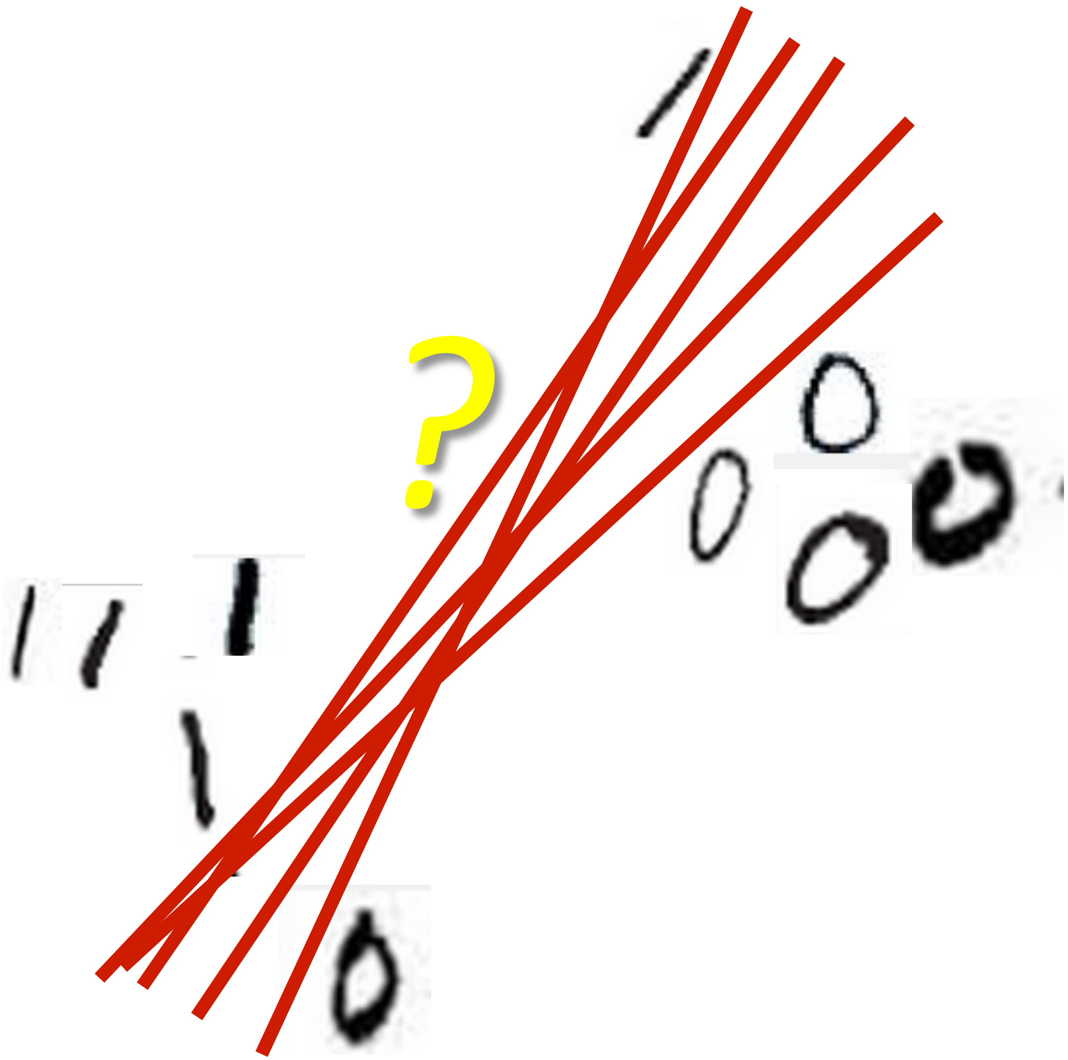


The Perceptron algorithm

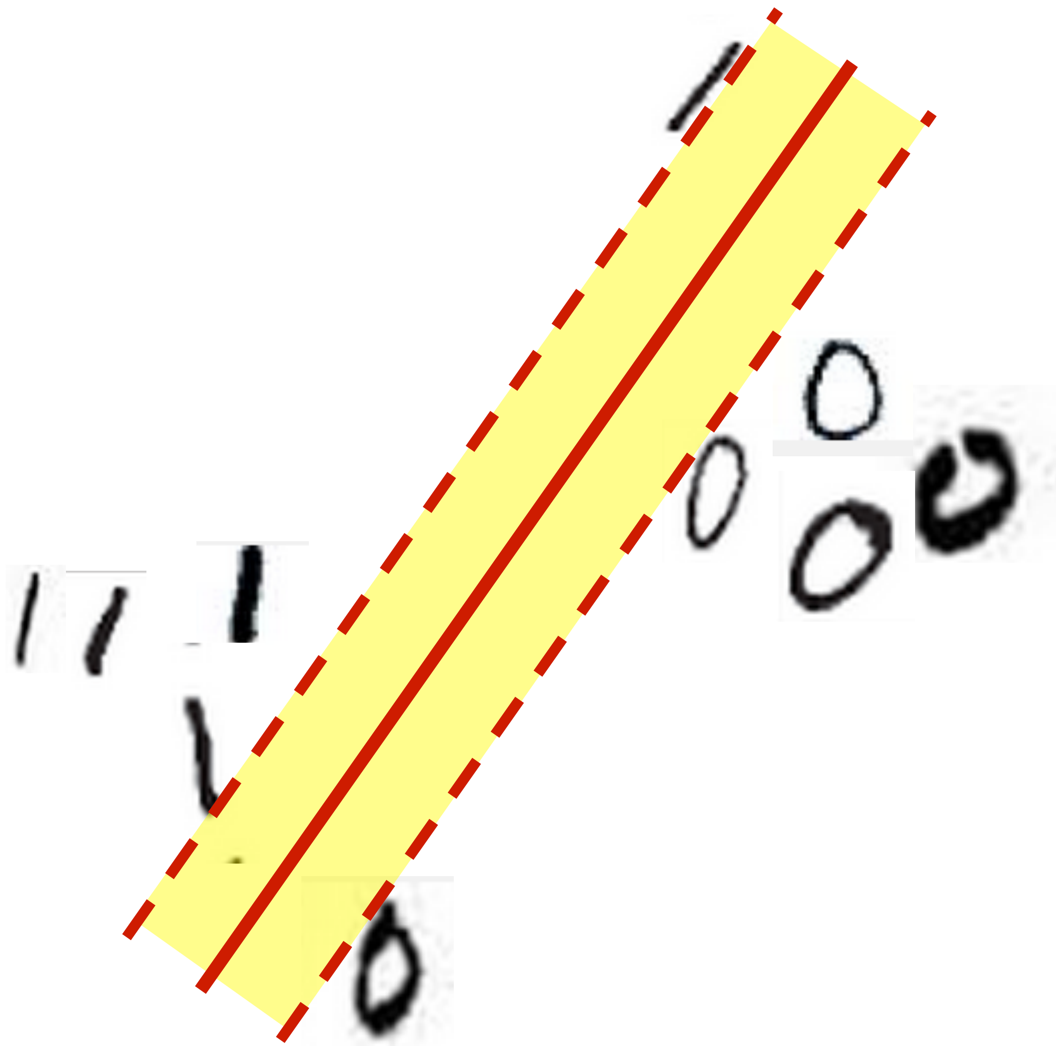


Always find ONE separating hyperplane if the samples are **linearly separable**

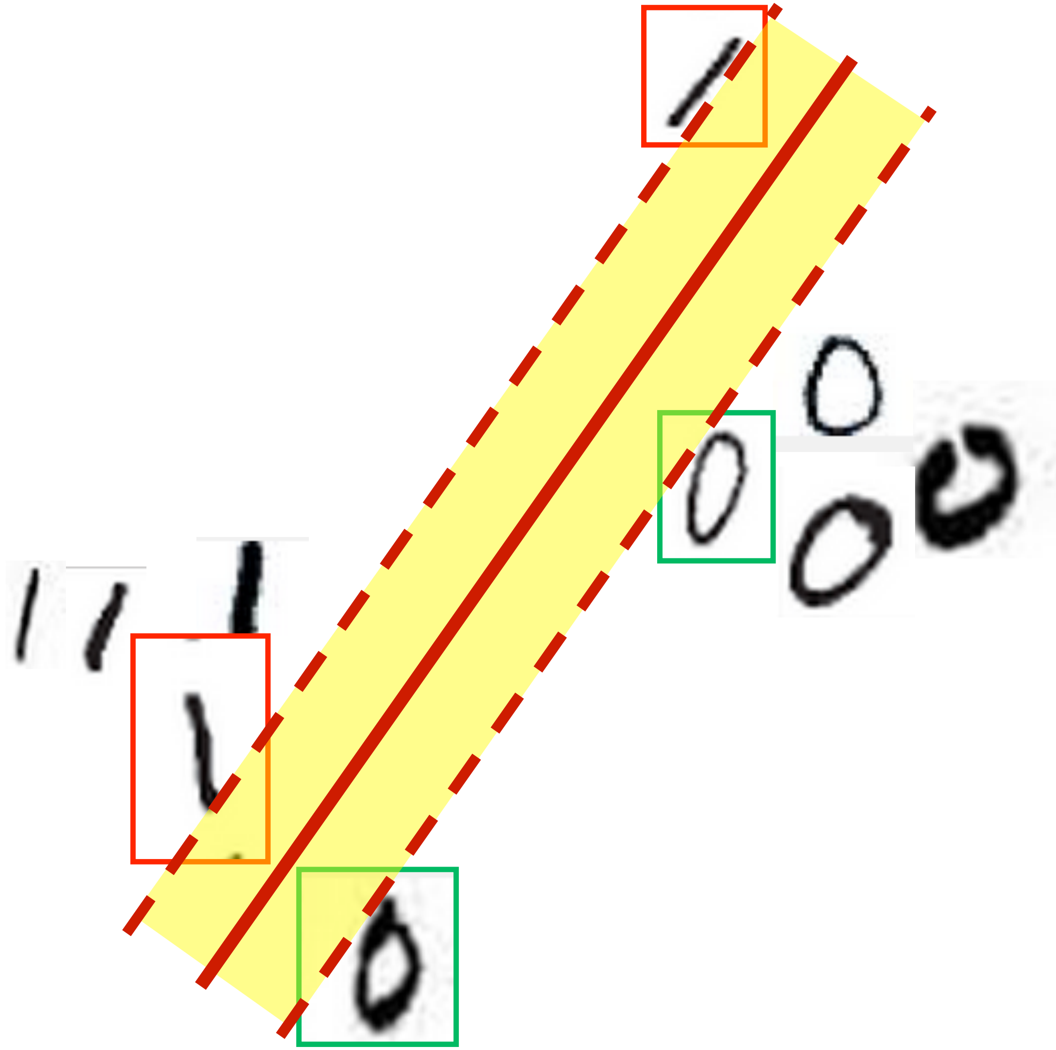
But which one is best?



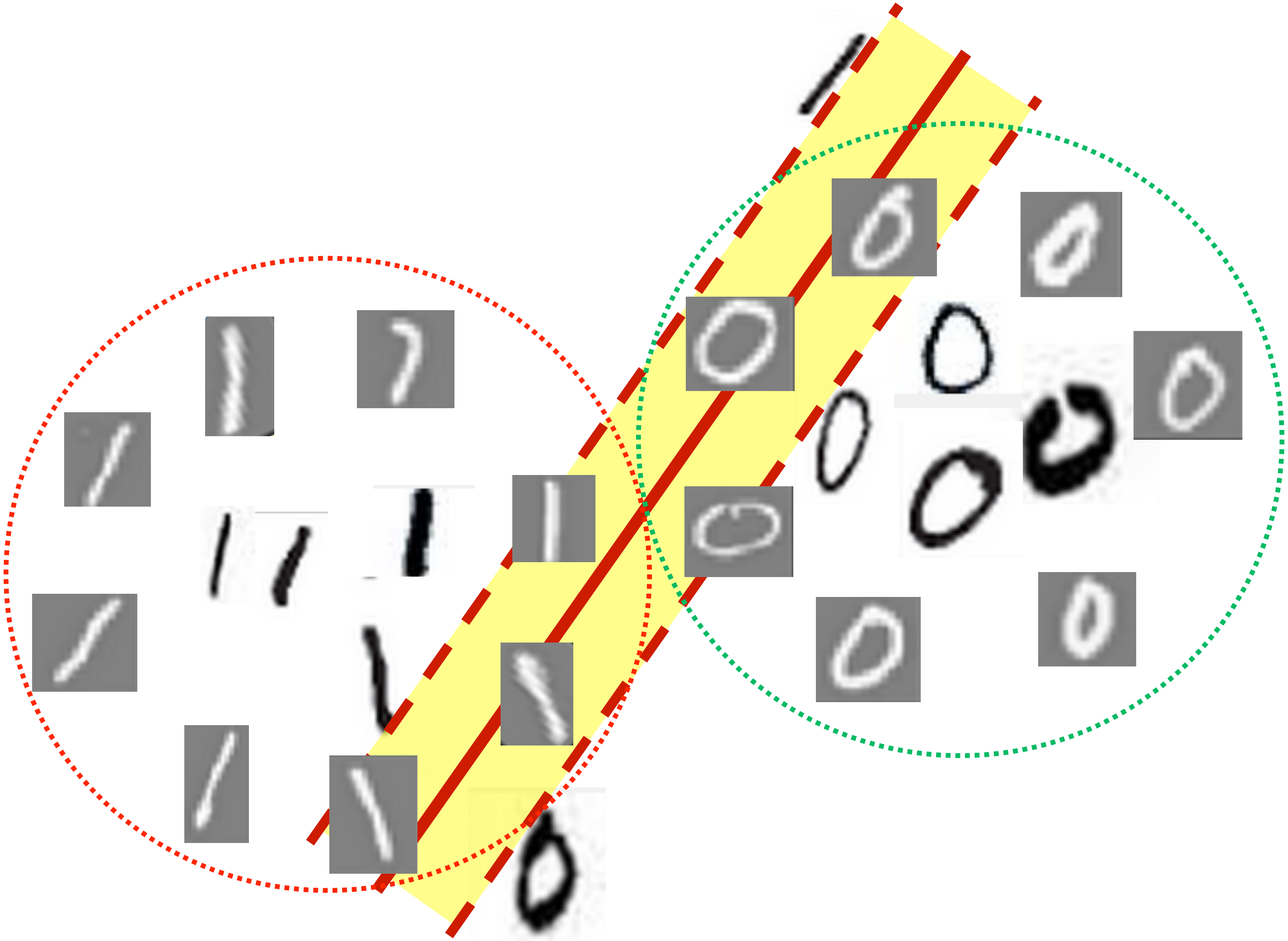
Safety margin



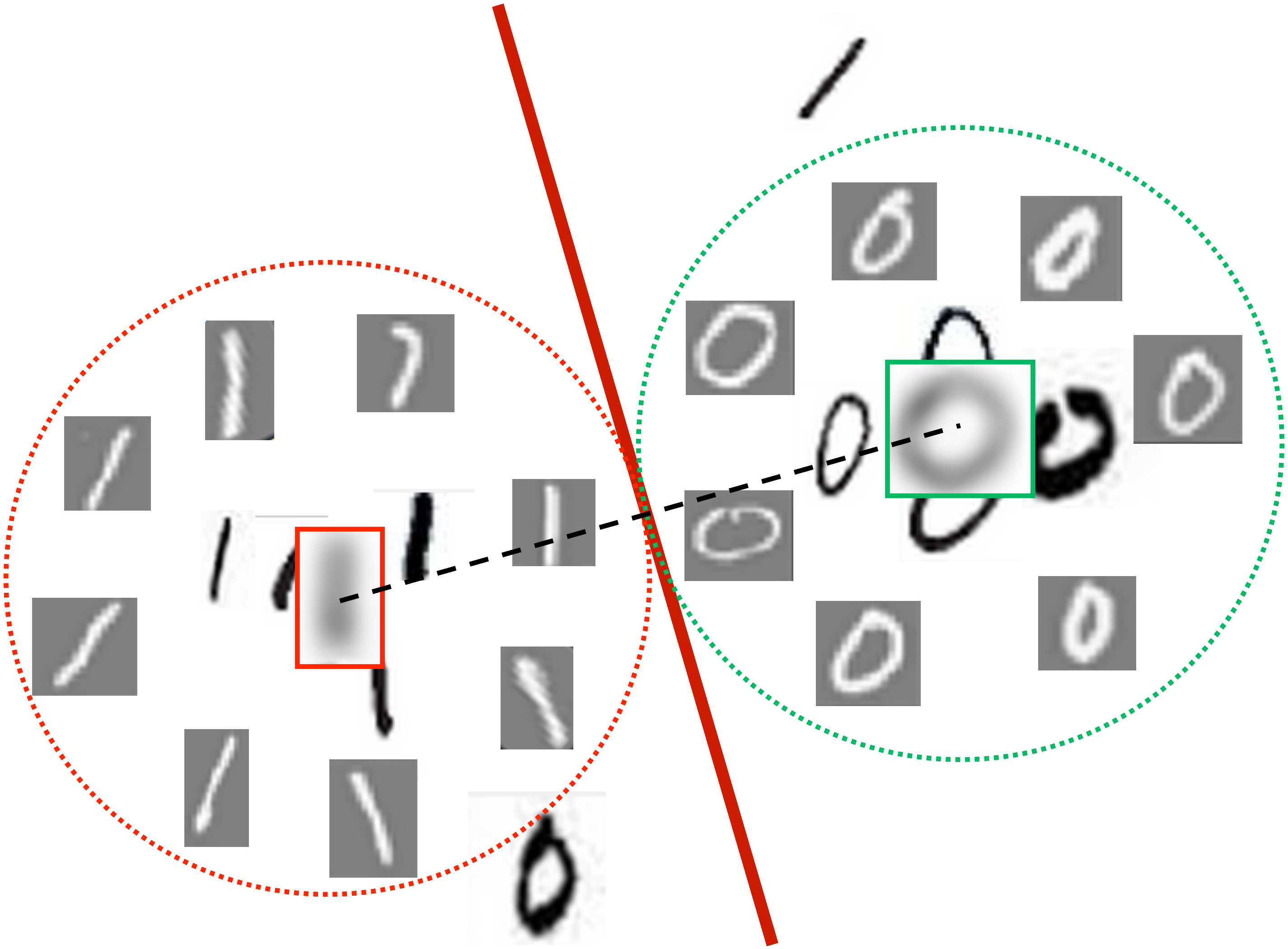
Support Vector Classifier



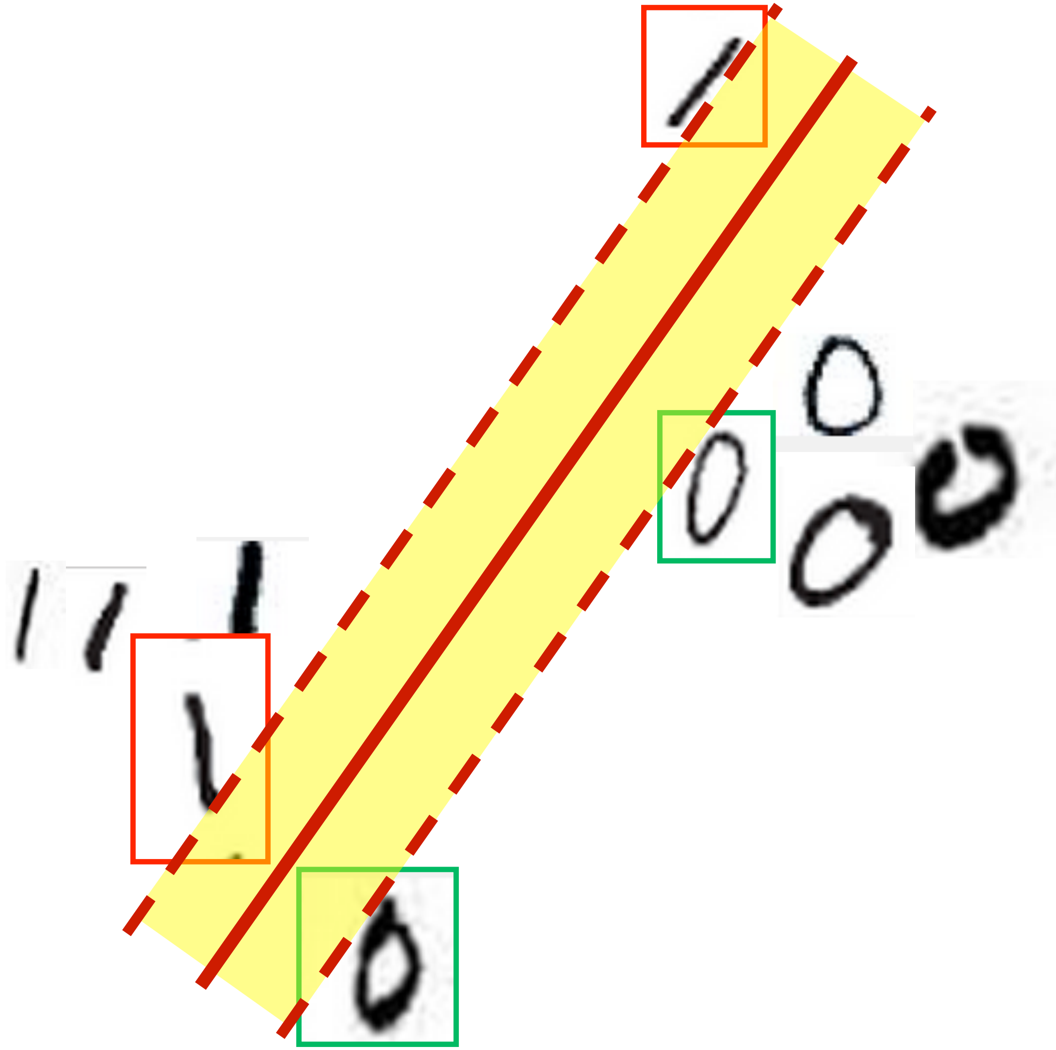
New test examples



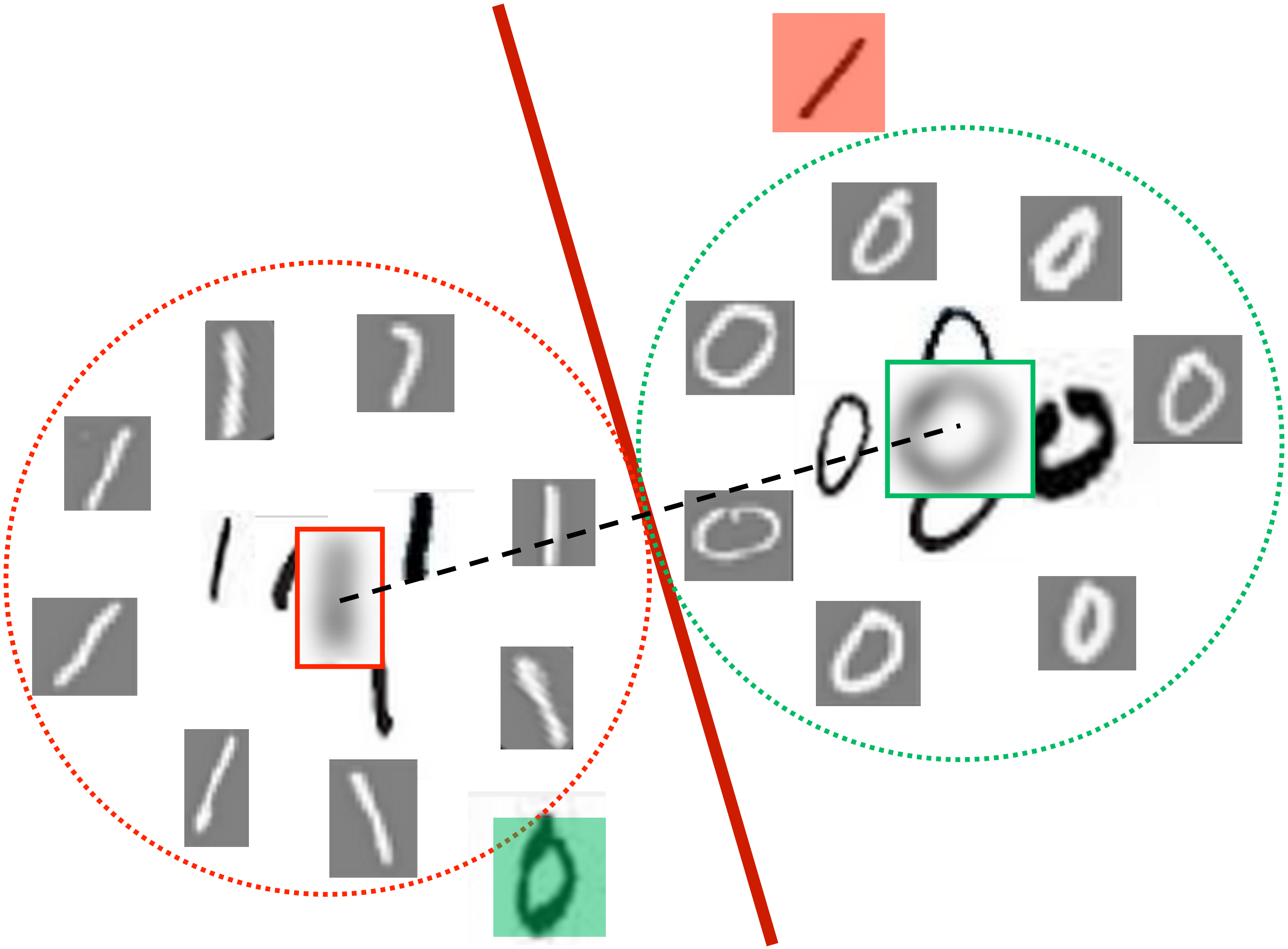
More “robust” solution



Better "fit"



The "robust" solution has training errors



Fit vs. Robustness tradeoff

Best fit: Zero training error. Based on “marginal” possibly “atypical” examples. Could be outliers.

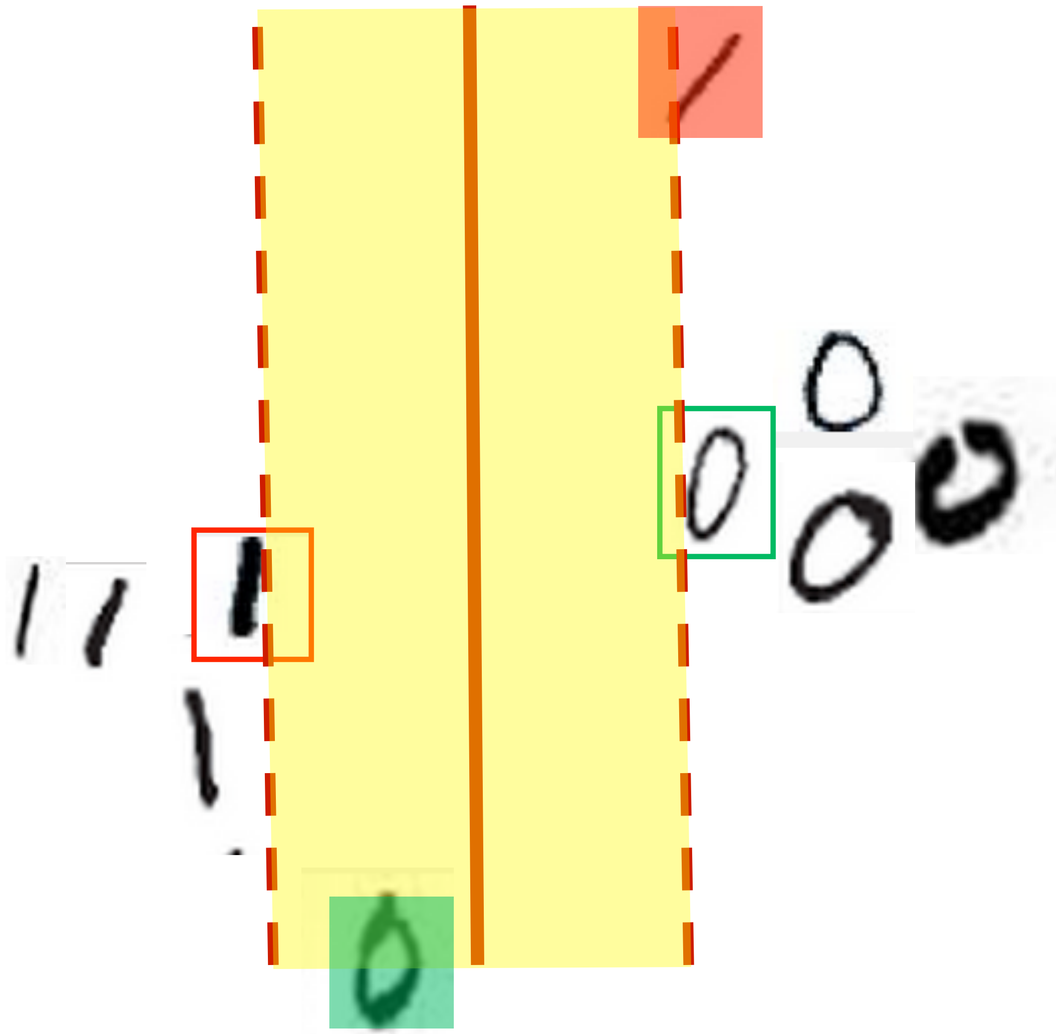
Best robustness: Based on typical examples or average examples (centroids).

Compromise

Good fit: Allow a few training errors.

Good robustness: Maximize the margin.

“Soft” margin



Soft Margin **C**ompromise

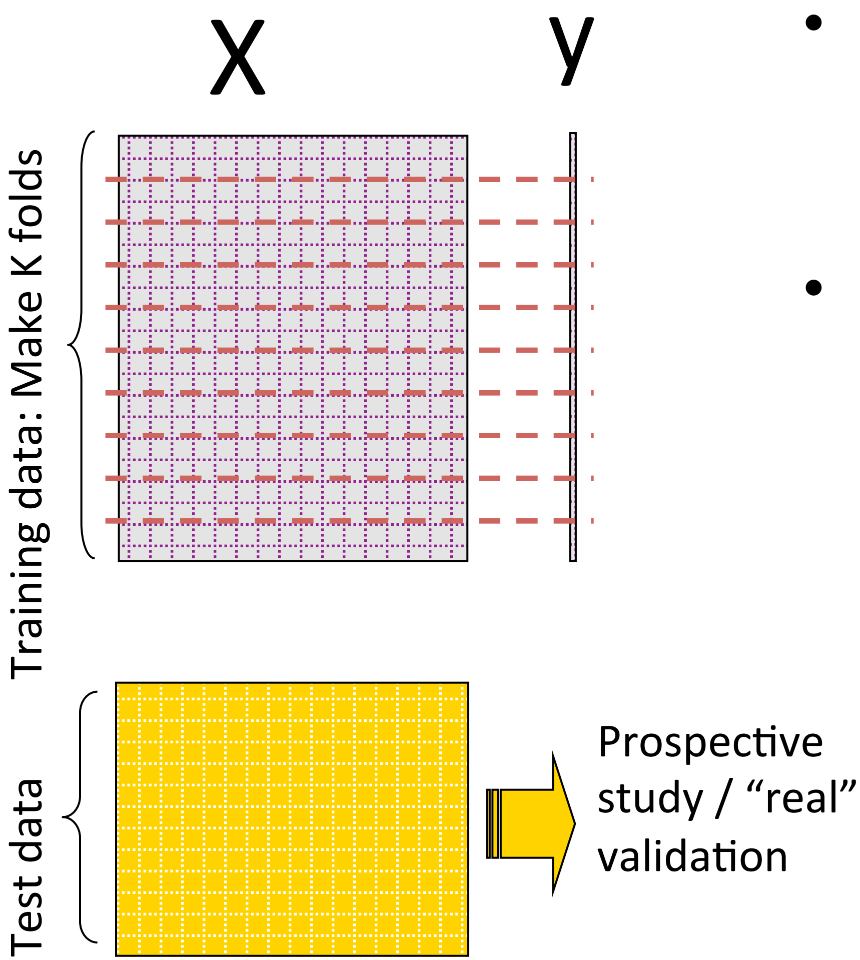
Minimize

(1/Margin) + C Training error

Good robustness

Good fit

Hyper-parameter Selection



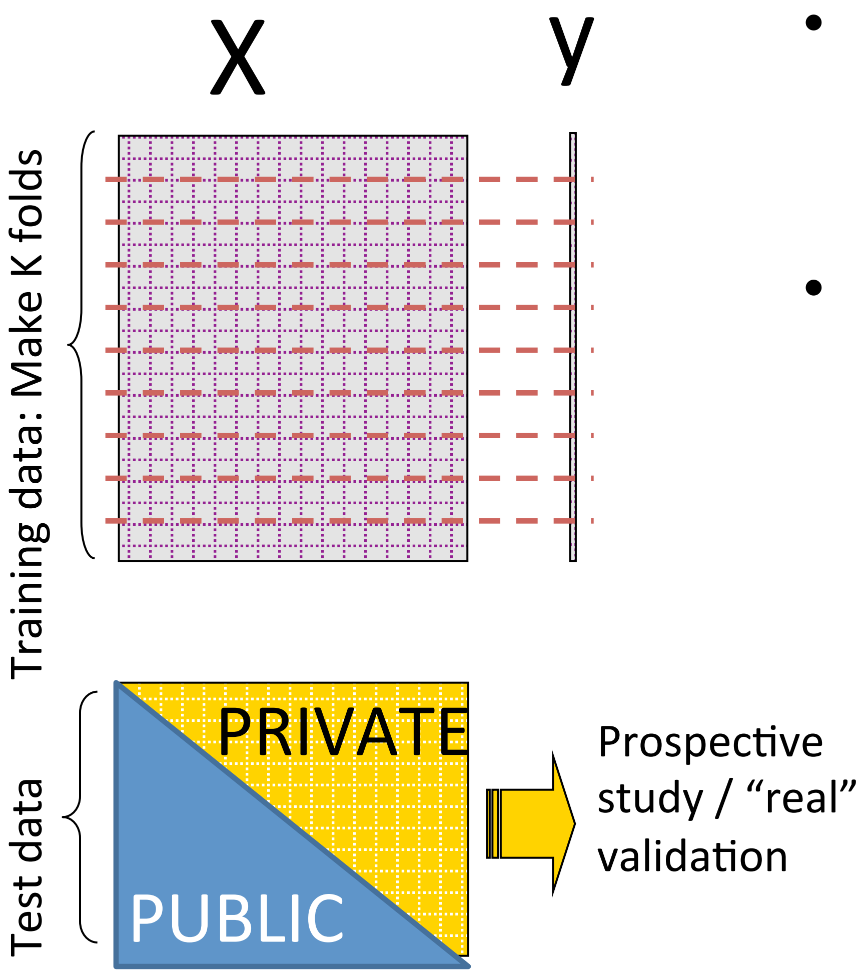
- **Learning = adjusting:**
  parameters (**w** vector).
  hyper-parameter (**C**).

- ***Cross-validation* with K-folds:**

For various values of **C**:

- Adjust **w** on a fraction (K-1)/K of training examples *e.g.* 9/10th.
- Test on 1/K remaining examples *e.g.* 1/10th.
- Rotate examples and average test results (CV error).
- Select **C** to minimize CV error.
- Re-compute **w** on **all** training examples using optimal **C.**

Hyper-parameter Selection



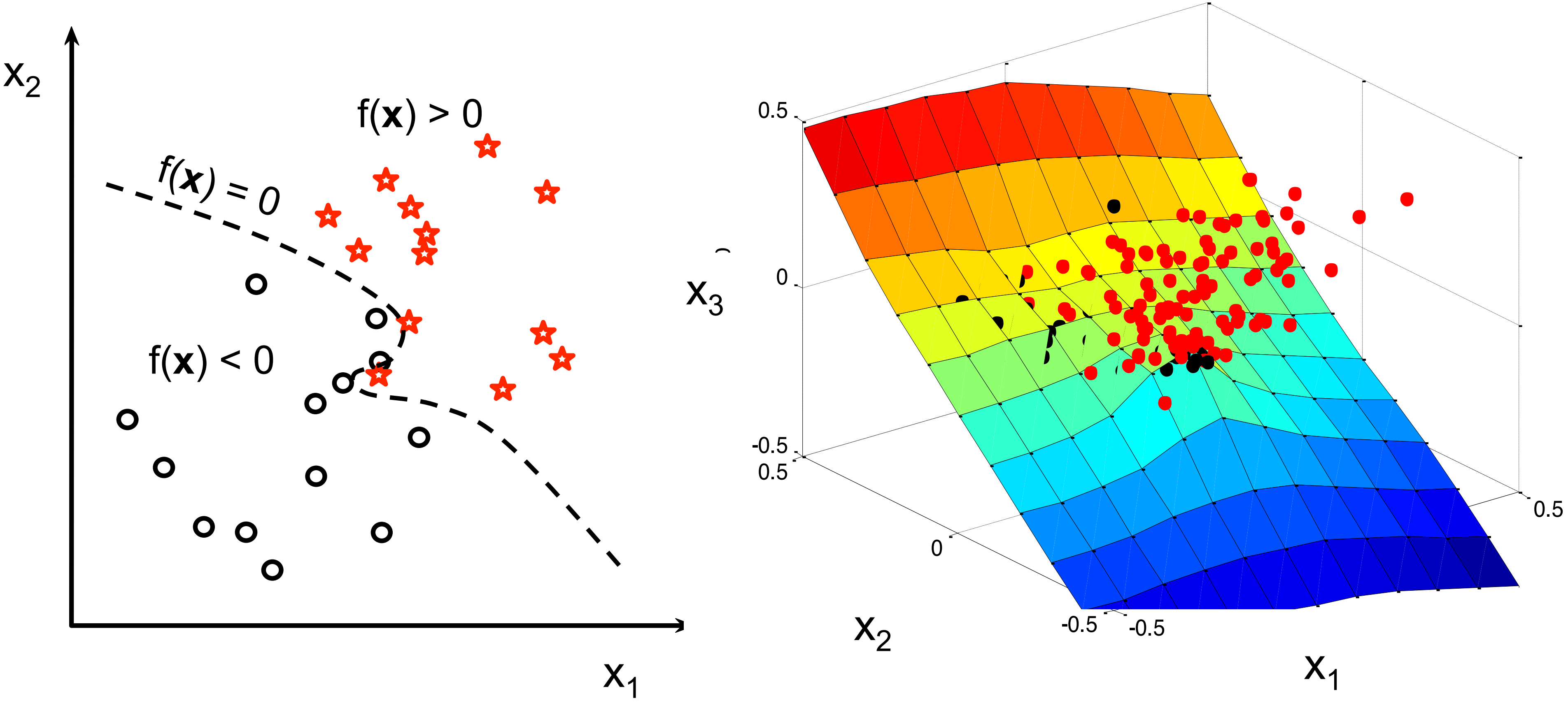
- **Learning = adjusting:**
  parameters (**w** vector).
  hyper-parameter (**C**).

- ***Cross-validation* with K-folds:**

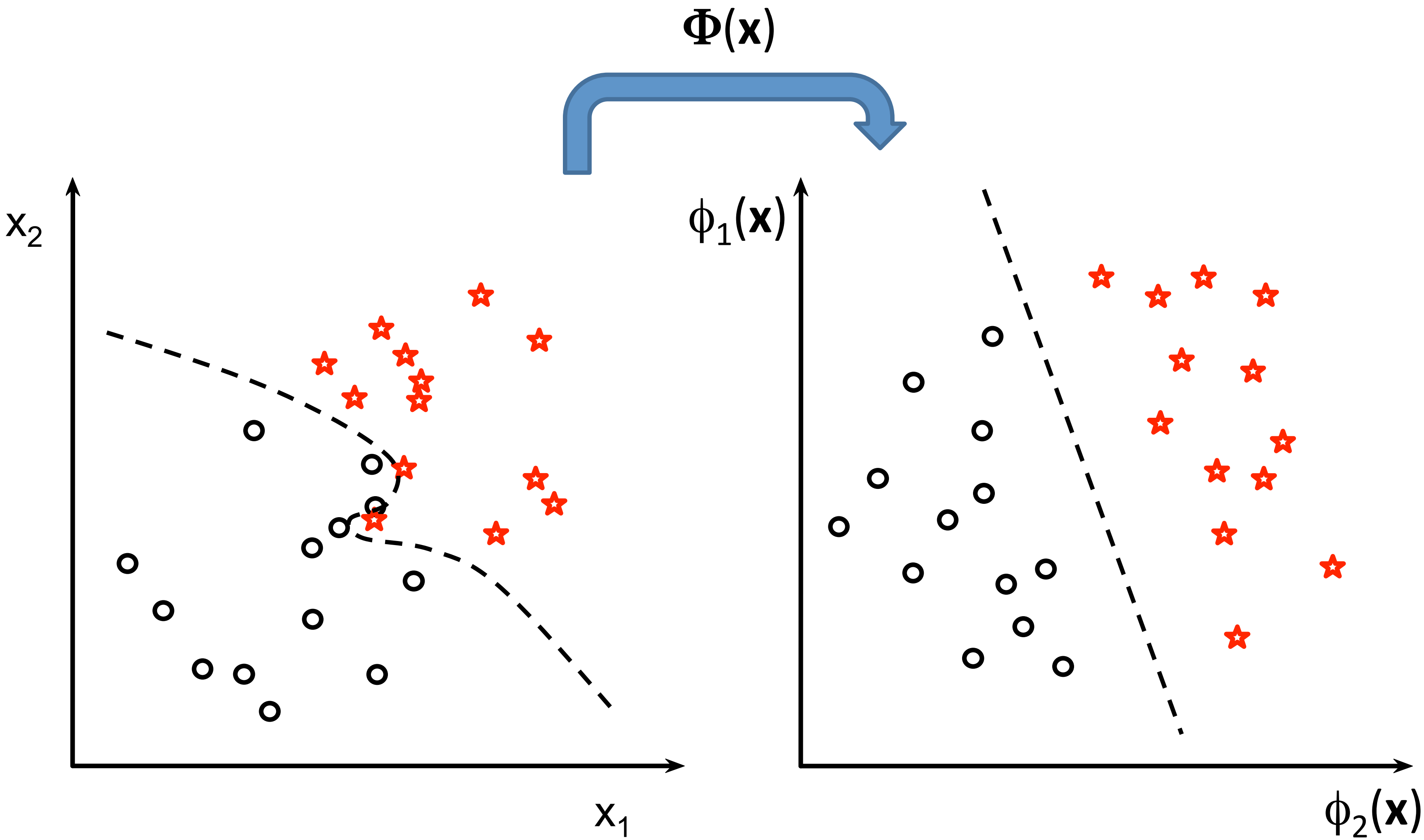
For various values of **C**:

- Adjust **w** on a fraction (K-1)/K of training examples *e.g.* 9/10th.
- Test on 1/K remaining examples *e.g.* 1/10th.
- Rotate examples and average test results (CV error).
- Select **C** to minimize CV error.
- Re-compute **w** on **all** training examples using optimal **C.**

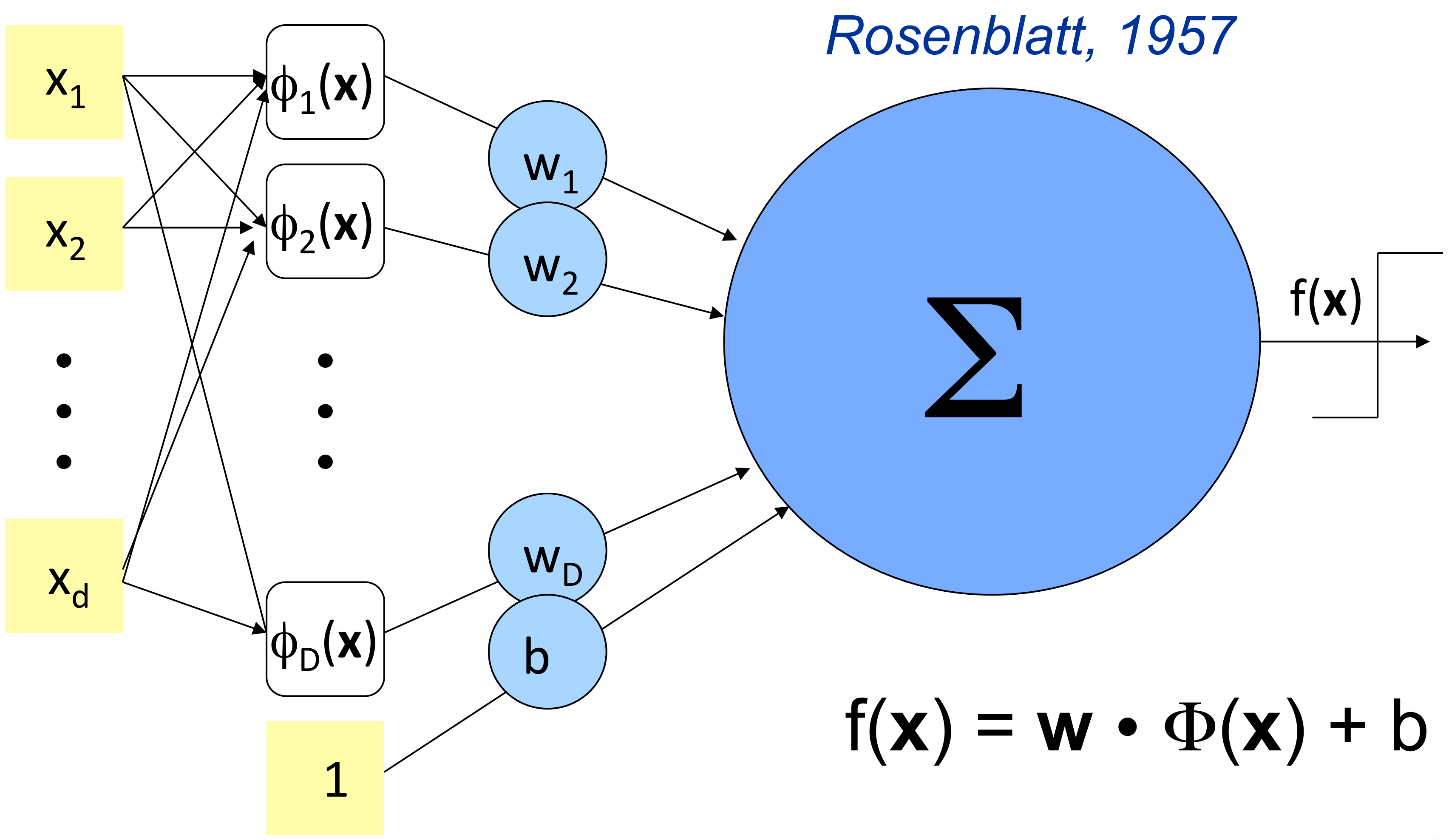
Non-linear decision boundary



Linear decision boundary in Φ-space



Perceptron

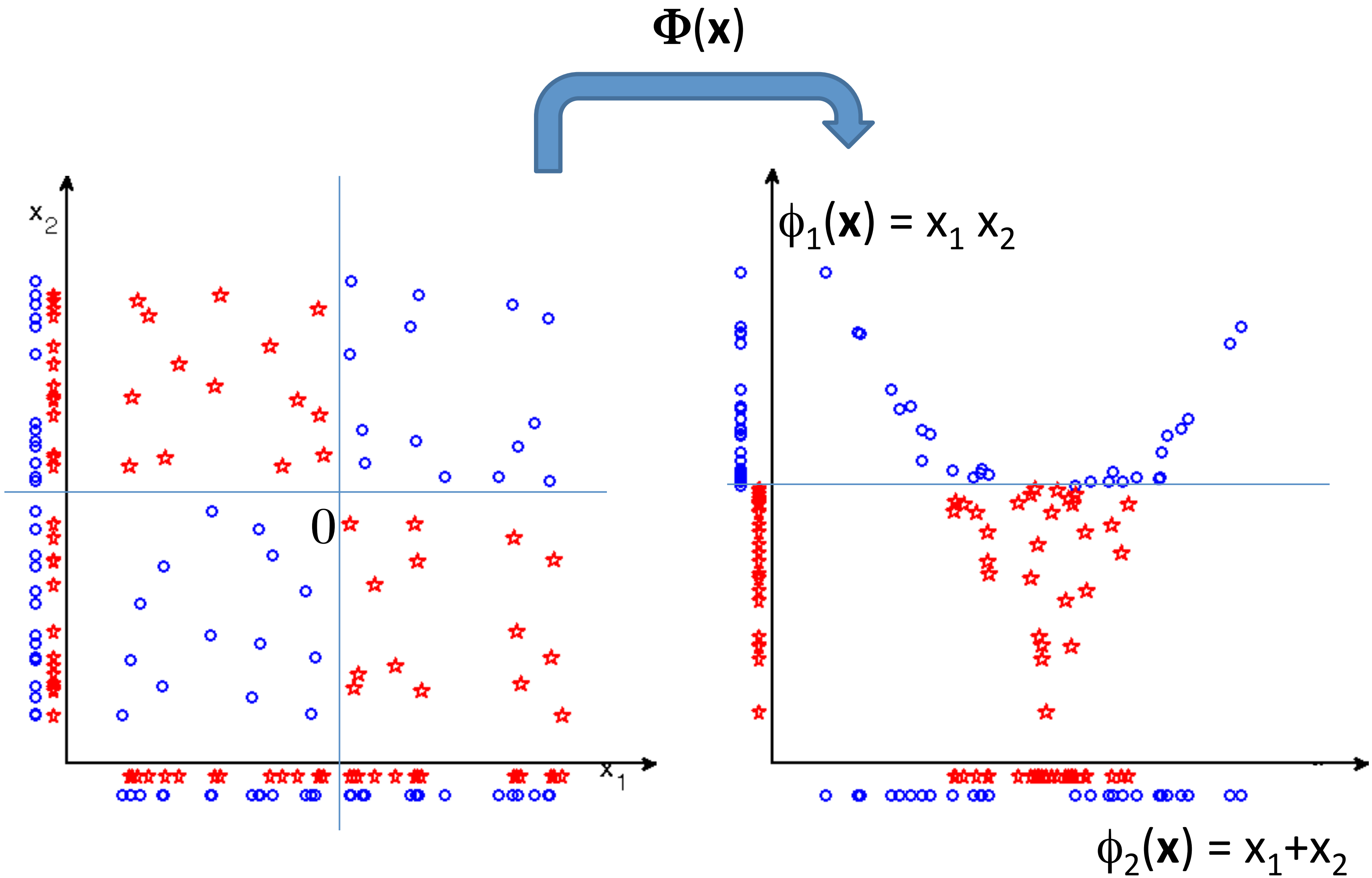


$$f(\mathbf{x}) = \mathbf{w} \bullet \Phi(\mathbf{x}) + b$$

What can Φ features be?

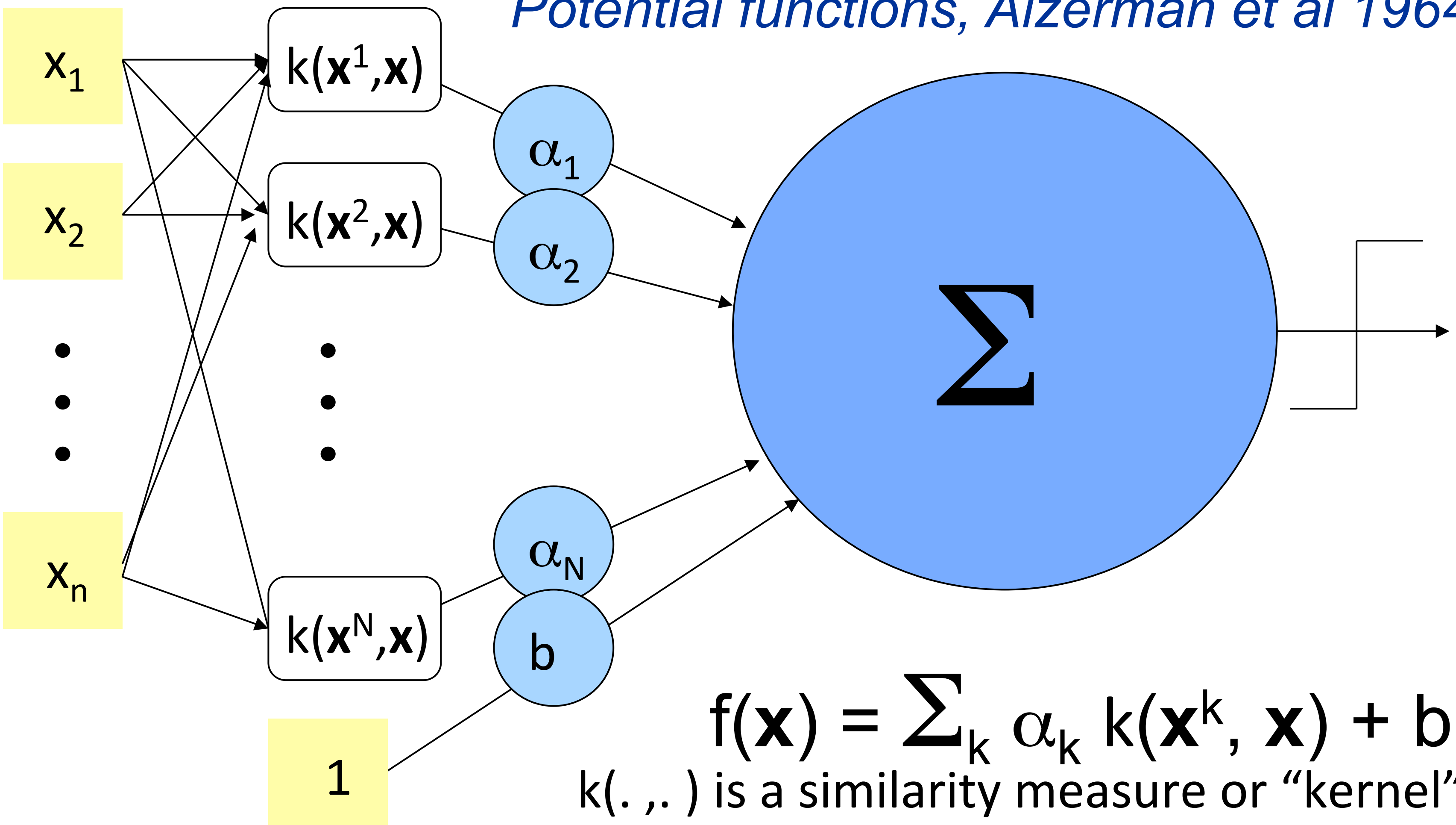
- **Hand crafted features**
  (lines, crosses, corners).
- **Randomly generated functions**
  (sums, products).
- **“Dictionary” features**
  (little pieces of images).
- **Basis functions of transforms**
  (Fourier, Wavelet)

Chessboard problem



Kernel Method

Potential functions, Aizerman et al 1964

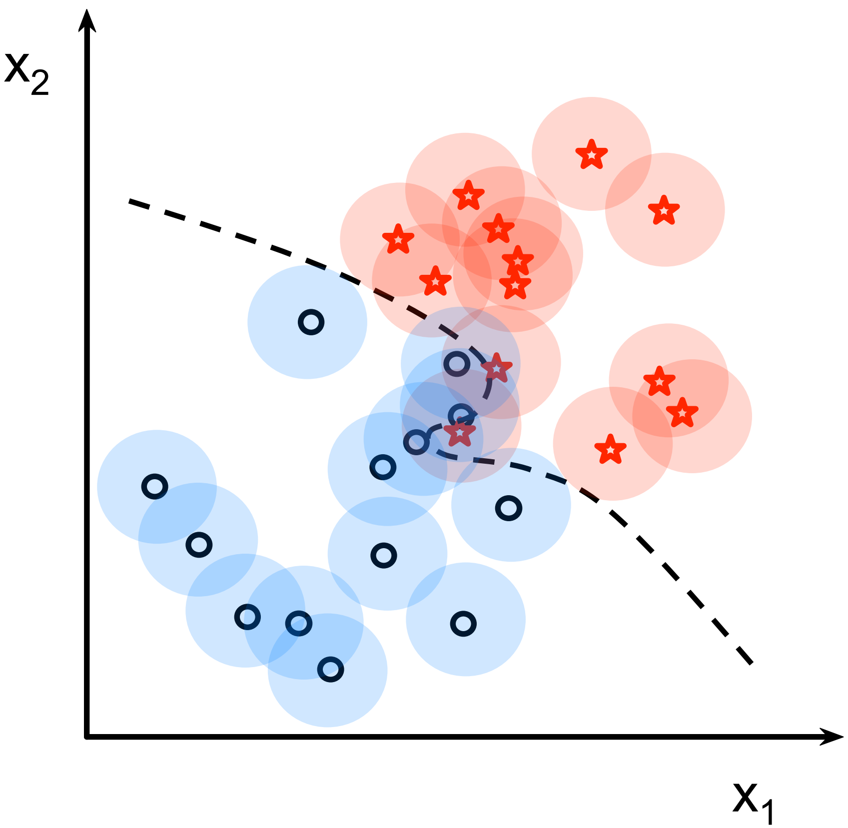


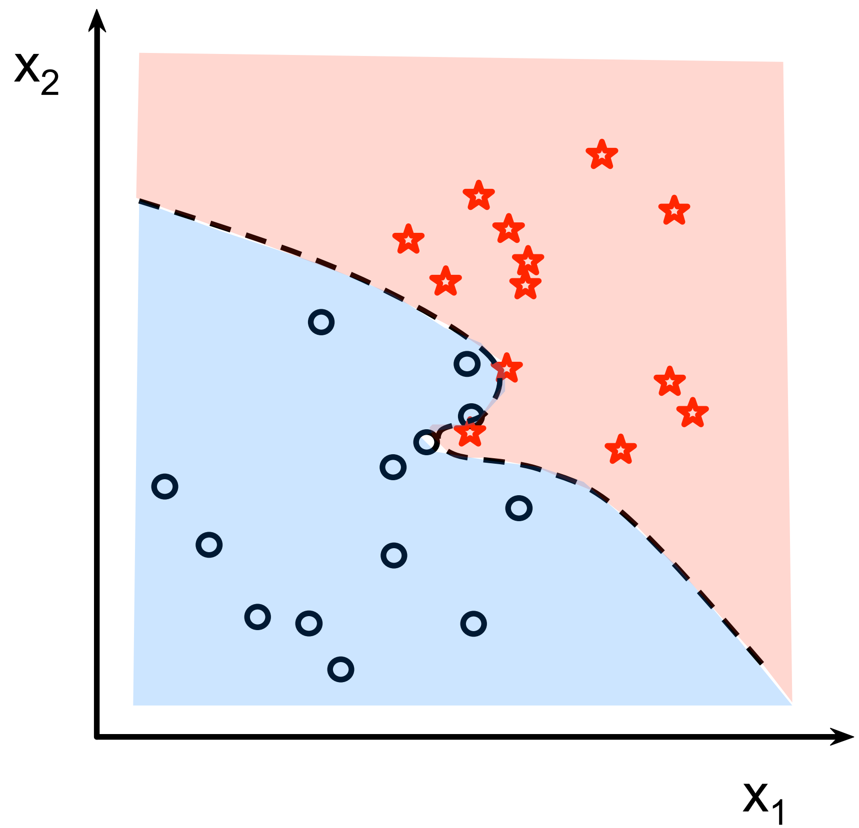
$$f(\mathbf{x}) = \sum_k \alpha_k \, k(\mathbf{x}^k, \mathbf{x}) + b$$

k(. ,.) is a similarity measure or "kernel".

Radial basis functions

$$f(\mathbf{x}) = \sum_k \alpha_k \, k(\mathbf{x}^k, \mathbf{x}) + b$$





What is a Kernel?

A kernel is:

- a **similarity measure**
- a **dot product** in *some* feature space:

$$k(\mathbf{s}, \mathbf{t}) = \Phi(\mathbf{s}) \bullet \Phi(\mathbf{t})$$

But we do not need to know the Φ representation.

- $k(\mathbf{s}, \mathbf{t}) = \exp(-\|\mathbf{s}-\mathbf{t}\|^2/\sigma^2)$ Gaussian kernel
- $k(\mathbf{s}, \mathbf{t}) = 1/\|\mathbf{s}-\mathbf{t}\|$ Potential function
- $k(\mathbf{s}, \mathbf{t}) = (\mathbf{s} \bullet \mathbf{t})^q$ Polynomial kernel

$$\underbrace{([s_1,s_2]\bullet[t_1,t_2])^2}_{k(\mathbf{s},\,\mathbf{t})} = \underbrace{[s_1{}^2,s_2{}^2,\sqrt{2}s_1s_2]}_{\Phi(\mathbf{s})} \,.\underbrace{[t_1{}^2,t_2{}^2,\sqrt{2}t_1t_2]}_{\Phi(\mathbf{t})}$$

What is a Kernel?

A kernel is:

- a **similarity measure**
- a **dot product** in *some* feature space:

$$k(\mathbf{s}, \mathbf{t}) = \Phi(\mathbf{s}) \bullet \Phi(\mathbf{t})$$

But we do not need to know the Φ representation.

- $k(\mathbf{s}, \mathbf{t}) = \exp(-\|\mathbf{s}-\mathbf{t}\|^2/\sigma^2)$ Gaussian kernel
- $k(\mathbf{s}, \mathbf{t}) = 1/\|\mathbf{s}-\mathbf{t}\|$ Potential function
- $k(\mathbf{s}, \mathbf{t}) = (\mathbf{s} \bullet \mathbf{t})^q$ Polynomial kernel

$$\underbrace{([s_1,s_2]\bullet[t_1,t_2])^2}_{k(\mathbf{s},\,\mathbf{t})} = \underbrace{[s_1{}^2,s_2{}^2,\sqrt{2}s_1s_2]}_{\Phi(\mathbf{s})} \,.\underbrace{[t_1{}^2,t_2{}^2,\sqrt{2}t_1t_2]}_{\Phi(\mathbf{t})}$$

What is a Kernel?

A kernel is:

- a **similarity measure**
- a **dot product** in *some* feature space:

$$k(\mathbf{s}, \mathbf{t}) = \Phi(\mathbf{s}) \bullet \Phi(\mathbf{t})$$

But we do not need to know the Φ representation.

- $k(\mathbf{s}, \mathbf{t}) = \exp(-\|\mathbf{s}-\mathbf{t}\|^2/\sigma^2)$ Gaussian kernel
- $k(\mathbf{s}, \mathbf{t}) = 1/\|\mathbf{s}-\mathbf{t}\|$ Potential function
- $k(\mathbf{s}, \mathbf{t}) = (\mathbf{s} \bullet \mathbf{t})^q$ Polynomial kernel

$$\underbrace{([s_1,s_2]\bullet[t_1,t_2])^2}_{k(\mathbf{s},\,\mathbf{t})} = \underbrace{[s_1{}^2,s_2{}^2,\sqrt{2}s_1s_2]}_{\Phi(\mathbf{s})} \,.\underbrace{[t_1{}^2,t_2{}^2,\sqrt{2}t_1t_2]}_{\Phi(\mathbf{t})}$$

What is a Kernel?

A kernel is:

- a **similarity measure**
- a **dot product** in *some* feature space:

$$k(\mathbf{s}, \mathbf{t}) = \mathbf{\Phi}(\mathbf{s}) \bullet \mathbf{\Phi}(\mathbf{t})$$

But we do not need to know the Φ representation.

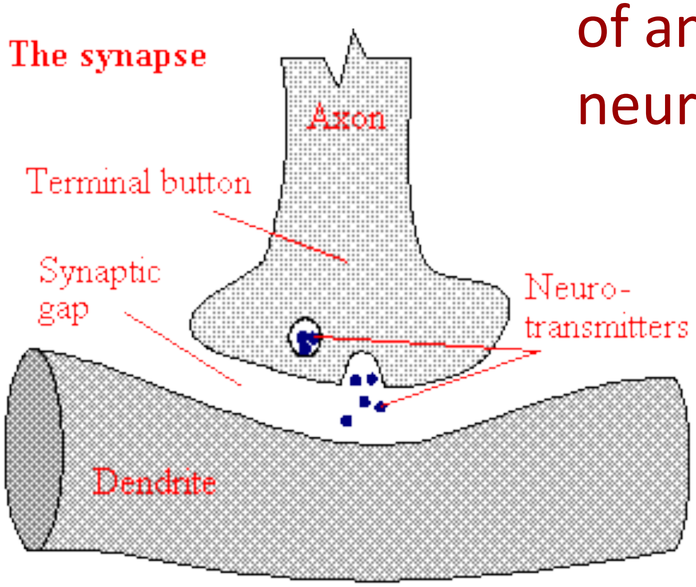
- $k(\mathbf{s}, \mathbf{t}) = \exp(-\|\mathbf{s}-\mathbf{t}\|^2/\sigma^2)$ — Gaussian kernel
- $k(\mathbf{s}, \mathbf{t}) = 1/\|\mathbf{s}-\mathbf{t}\|$ — Potential function
- $k(\mathbf{s}, \mathbf{t}) = (\mathbf{s} \bullet \mathbf{t} + 1)^q$ — Polynomial kernel

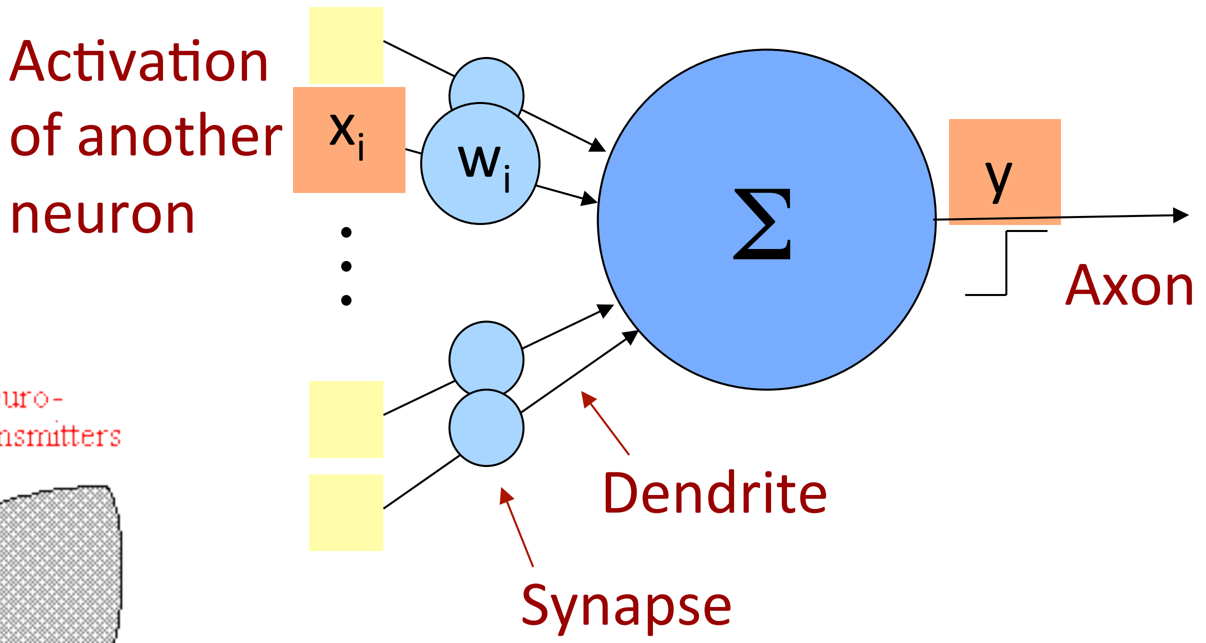
$$\underbrace{([s_1,s_2]\bullet[t_1,t_2])^2}_{k(\mathbf{s},\,\mathbf{t})} = \underbrace{[s_1^{\,2}, s_2^{\,2}, \sqrt{2}s_1s_2]}_{\mathbf{\Phi}(\mathbf{s})} \,.\, \underbrace{[t_1^{\,2}, t_2^{\,2}, \sqrt{2}t_1t_2]}_{\mathbf{\Phi}(\mathbf{t})}$$

Hebb's Rule

D. Hebb, 1949

$$\boxed{w_i \leftarrow w_i + y^k x^k_i}$$

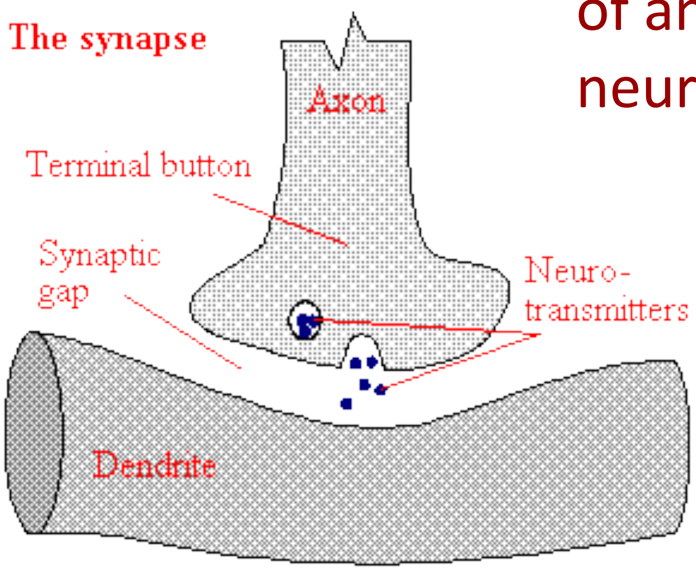


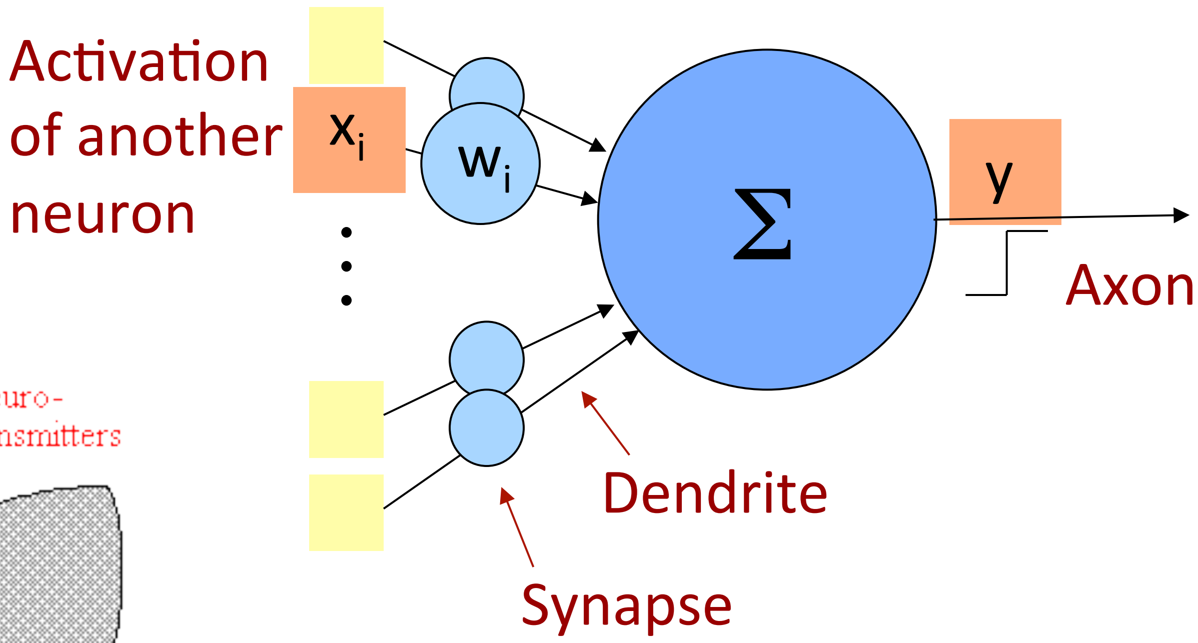


Hebb's Rule

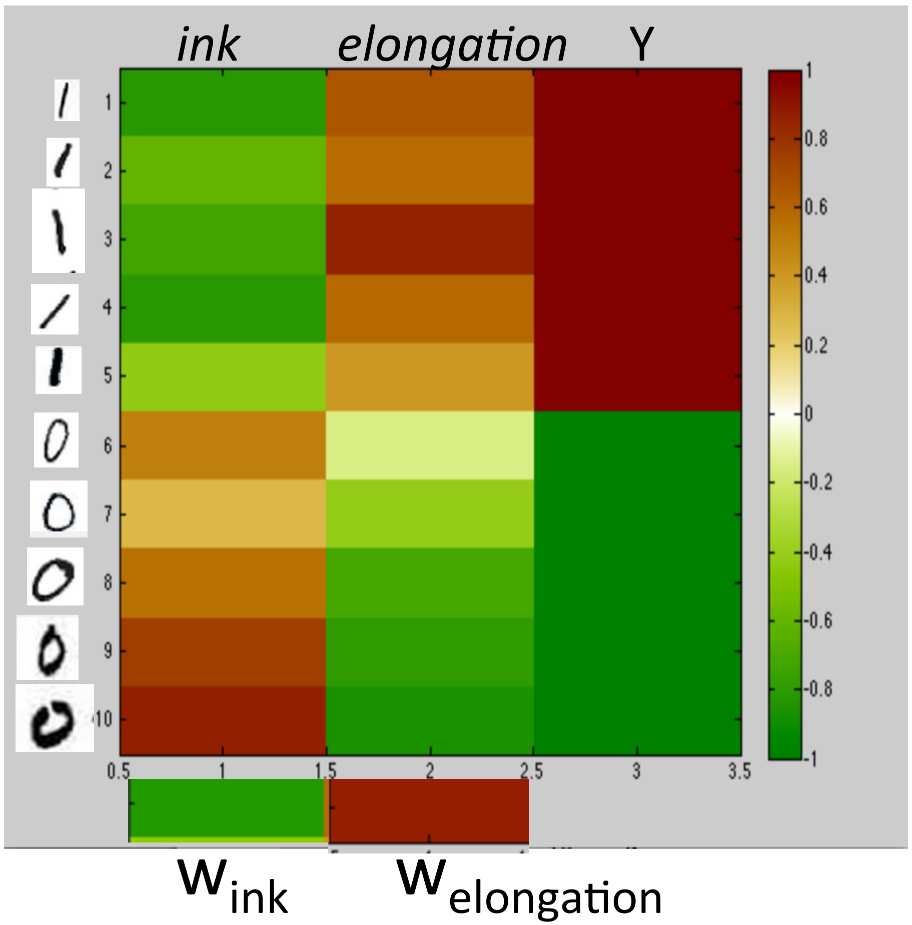
D. Hebb, 1949

$$\boxed{\mathbf{w} \leftarrow \mathbf{w} + y^k \, \mathbf{x}^k}$$





Hebb's rule



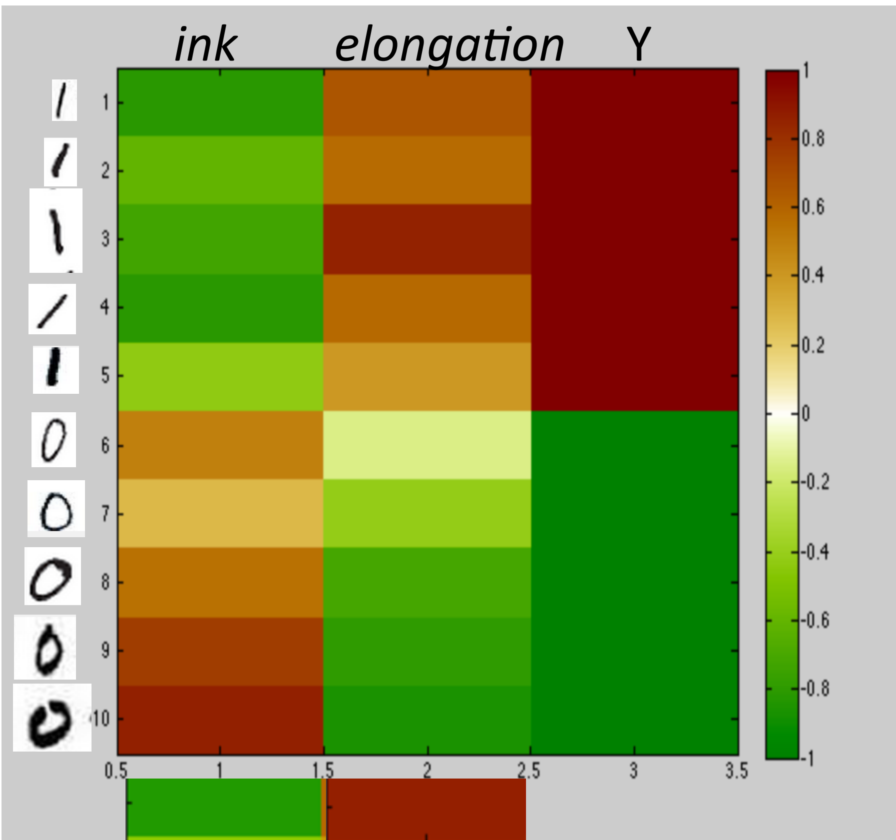
$$\boxed{w_i \leftarrow w_i + y^k x^k_i}$$

$$w_i = \Sigma_k y^k x^k_i = \mathbf{y} \bullet \mathbf{x}_i$$

$$\mathbf{w} = \sum_k y^k \mathbf{x}^k$$

$$\begin{aligned} f(x) &= \mathbf{w} \bullet \mathbf{x} \\ &= \Sigma_i w_i x_i \\ &= \sum_k y^k \mathbf{x}^k \bullet \mathbf{x} \\ &\sim \mathbf{w}^{[1]} \bullet \mathbf{x} - \mathbf{w}^{[0]} \bullet \mathbf{x} \end{aligned}$$

Hebb's rule



w_{ink} $\quad$ $w_{elongation}$

$$\boxed{w_i \leftarrow w_i + y^k x^k_i}$$

$$w_i = \Sigma_k y^k x^k_i = \mathbf{y} \bullet \mathbf{x}_i$$

$$\mathbf{w} = \sum_k y^k \mathbf{x}^k$$

$$f(x) = \mathbf{w} \bullet \mathbf{x}$$

$$= \Sigma_i w_i x_i$$

PARAMETRIC

$$= \sum_k y^k \mathbf{x}^k \bullet \mathbf{x}$$

NON PARAMETRIC

$$\sim \mathbf{w}^{[1]} \bullet \mathbf{x} - \mathbf{w}^{[0]} \bullet \mathbf{x}$$

Kernel "Trick" (for Hebb's rule)

$$\boxed{\mathbf{w} = \sum_k y^k \, \mathbf{x}^k}$$

- Hebb's rule for the Perceptron:

$$\mathbf{w} = \sum_k y^k \, \Phi(\mathbf{x}_k)$$

$$f(\mathbf{x}) = \mathbf{w} \bullet \Phi(\mathbf{x}) = \sum_k y^k \, \Phi(\mathbf{x}_k) \bullet \Phi(\mathbf{x})$$

- Define a dot product:

$$k(\mathbf{x}_i, \mathbf{x}) = \Phi(\mathbf{x}_i) \bullet \Phi(\mathbf{x})$$

$$f(\mathbf{x}) = \sum_i y_i \, k(\mathbf{x}_i, \mathbf{x})$$

Kernel "Trick" (for Hebb's rule)

$$\boxed{\mathbf{w} = \sum_k y^k \mathbf{x}^k}$$

- Hebb's rule for the Perceptron:

$$\mathbf{w} = \sum_k y^k \Phi(\mathbf{x}_k)$$

$$f(\mathbf{x}) = \mathbf{w} \bullet \Phi(\mathbf{x}) = \sum_k y^k \Phi(\mathbf{x}_k) \bullet \Phi(\mathbf{x})$$

PARAMETRIC

- Define a dot product:

$$k(\mathbf{x}_i, \mathbf{x}) = \Phi(\mathbf{x}_i) \bullet \Phi(\mathbf{x})$$

$$f(\mathbf{x}) = \sum_i y_i \, k(\mathbf{x}_i, \mathbf{x})$$

NON PARAMETRIC

Kernel "Trick" (general)

- $f(\mathbf{x}) = \sum_k \alpha_k \, k(\mathbf{x}^k, \mathbf{x})$
- $k(\mathbf{x}^k, \mathbf{x}) = \boldsymbol{\Phi}(\mathbf{x}^k) \bullet \boldsymbol{\Phi}(\mathbf{x})$

NON PARAMETRIC

Dual forms

- $f(\mathbf{x}) = \mathbf{w} \bullet \boldsymbol{\Phi}(\mathbf{x})$
- $\mathbf{w} = \sum_k \alpha_k \, \boldsymbol{\Phi}(\mathbf{x}^k)$

PARAMETRIC

Dual learning machines

PARAMETRIC

$f(\mathbf{x}) = \mathbf{w} \bullet \Phi(\mathbf{x})$

$\mathbf{w} = \sum_k \alpha_k \Phi(\mathbf{x}^k)$

Hebb's rule

$\mathbf{w} \leftarrow \mathbf{w} + y_k \Phi(\mathbf{x}^k)$

(Hebb 1949)

Perceptron algorithm

$\mathbf{w} \leftarrow \mathbf{w} + y_k \Phi(\mathbf{x}^k)$ if $y_k f(\mathbf{x}^k)<0$

(Rosenblatt 1958)

Minover (optimum margin)

$\mathbf{w} \leftarrow \mathbf{w} + y_k \Phi(\mathbf{x}^k)$ for min $y^k f(\mathbf{x}^k)$

(Krauth-Mézard 1987)

NON PARAMETRIC

$f(\mathbf{x}) = \sum_k \alpha_k \, k(\mathbf{x}^k, x)$

$k(\mathbf{x}^k, \mathbf{x}) = \Phi(\mathbf{x}^k).\Phi(\mathbf{x})$

Dual Hebb's rule

$\alpha_k \leftarrow \alpha_k + y_k$

Potential Function algorithm

$\alpha_k \leftarrow \alpha_k + y_k$ if $y_k f(\mathbf{x}^k)<0$

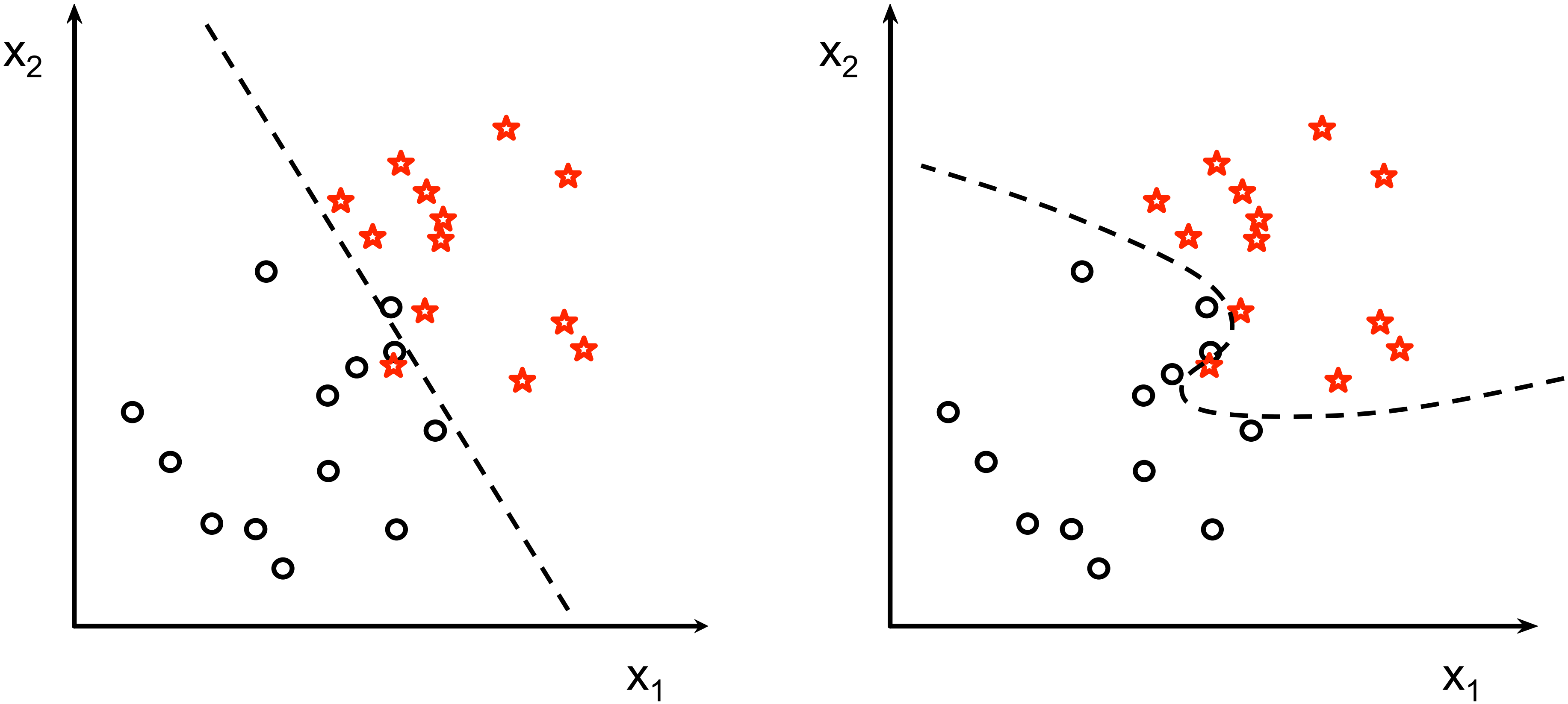
(Aizerman et al 1964)

Dual minover

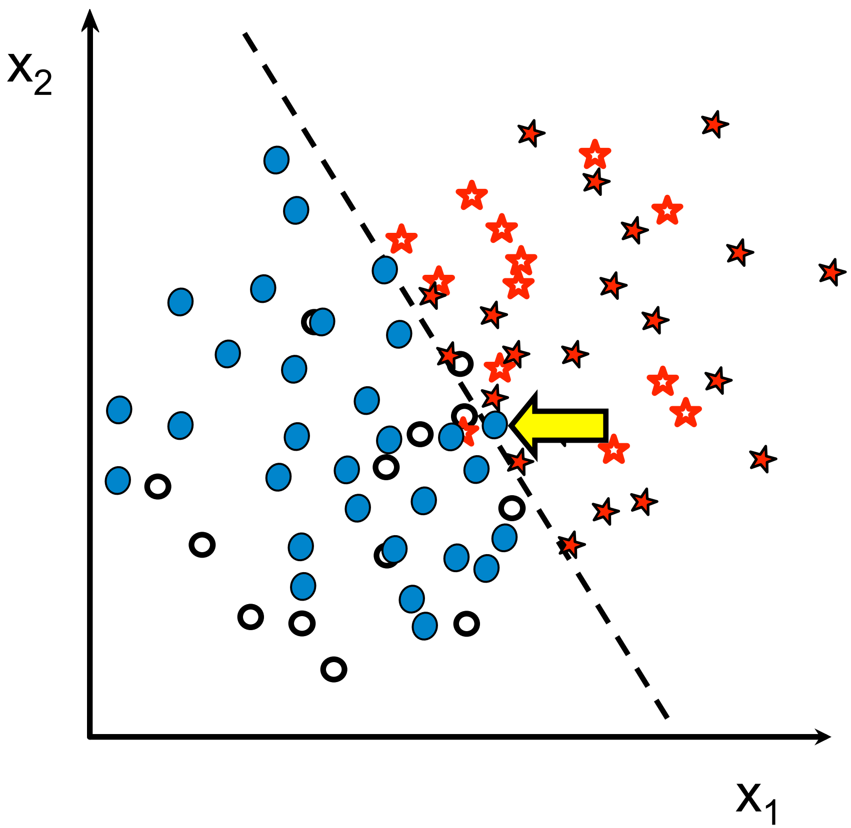
$\alpha_k \leftarrow \alpha_k + y^k$ for min $y_k f(\mathbf{x}^k)$

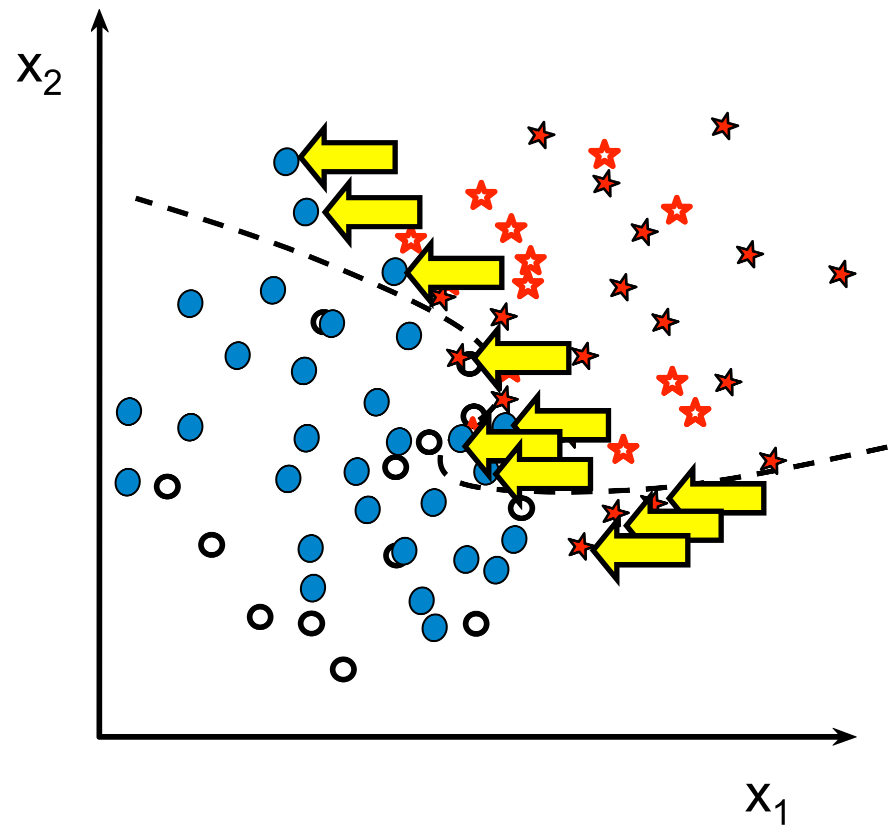
(ancestor of SVM)

Fit / Robustness Tradeoff



Fit / Robustness Tradeoff





Summary

- With linear threshold units ("neurons") we can build:
  - Linear discriminant
  - Kernel methods
  - Neural networks

DUAL

- The architectural hyper-parameters may include:
  - The choice of basis functions ϕ (features)
  - The kernel
  - The number of hidden units.
- "Complex" models are prone to overfitting.

Summary

- With linear threshold units ("neurons") we can build:
  - Linear discriminant
  - Kernel methods
  - Neural networks

PARAMETRIC

DUAL

NON PARAMETRIC

- The architectural hyper-parameters may include:
  - The choice of basis functions ϕ (features)
  - The kernel
  - The number of hidden units.
- "Complex" models are prone to overfitting.

Summary

- With linear threshold units ("neurons") we can build:
  - Linear discriminant
  - Kernel methods
  - Neural networks

PARAMETRIC

$\Updownarrow$ DUAL

NON PARAMETRIC

- The architectural hyper-parameters may include:
  - The choice of basis functions ϕ (features)
  - The kernel
  - The number of hidden units.
- "Complex" models are prone to overfitting.

Summary

- With linear threshold units ("neurons") we can build:
  – Linear discriminant
  – Kernel methods
  – Neural networks

DUAL

PARAMETRIC

NON PARAMETRIC

- The architectural hyper-parameters may include:
  – The choice of basis functions ϕ (features)
  – The kernel
  – The number of hidden units.

Control the COMPLEXITY

- "Complex" models are prone to overfitting.

Come to my office hours…
Wed 2:30-4:30 Soda 329

Next time

How to Train?

- Define a risk functional R[f(**x**,**w**)]
- Find a method to optimize it, typically "gradient descent"

$$w_j \leftarrow w_j - \eta\ \partial R / \partial w_j$$

or any optimization method (mathematical programming, simulated annealing, genetic algorithms, etc.)