

# $k$ -NN and Linear Classifiers, Learning

Tomáš Svoboda and Matěj Hoffmann




thanks to Daniel Novák and Filip Železný, Ondřej Drbohlav

Vision for Robots and Autonomous Systems, Center for Machine Perception

Department of Cybernetics

Faculty of Electrical Engineering, Czech Technical University in Prague

May 20, 2020

spojení dobré		82%
spojení postačující		517%
nedostatečné		00%

# $k$ -NN and Linear Classifiers, Learning

Tomáš Svoboda and Matěj Hoffmann




thanks to Daniel Novák and Filip Železný, Ondřej Drbohlav

Vision for Robots and Autonomous Systems, Center for Machine Perception

Department of Cybernetics

Faculty of Electrical Engineering, Czech Technical University in Prague

May 20, 2020

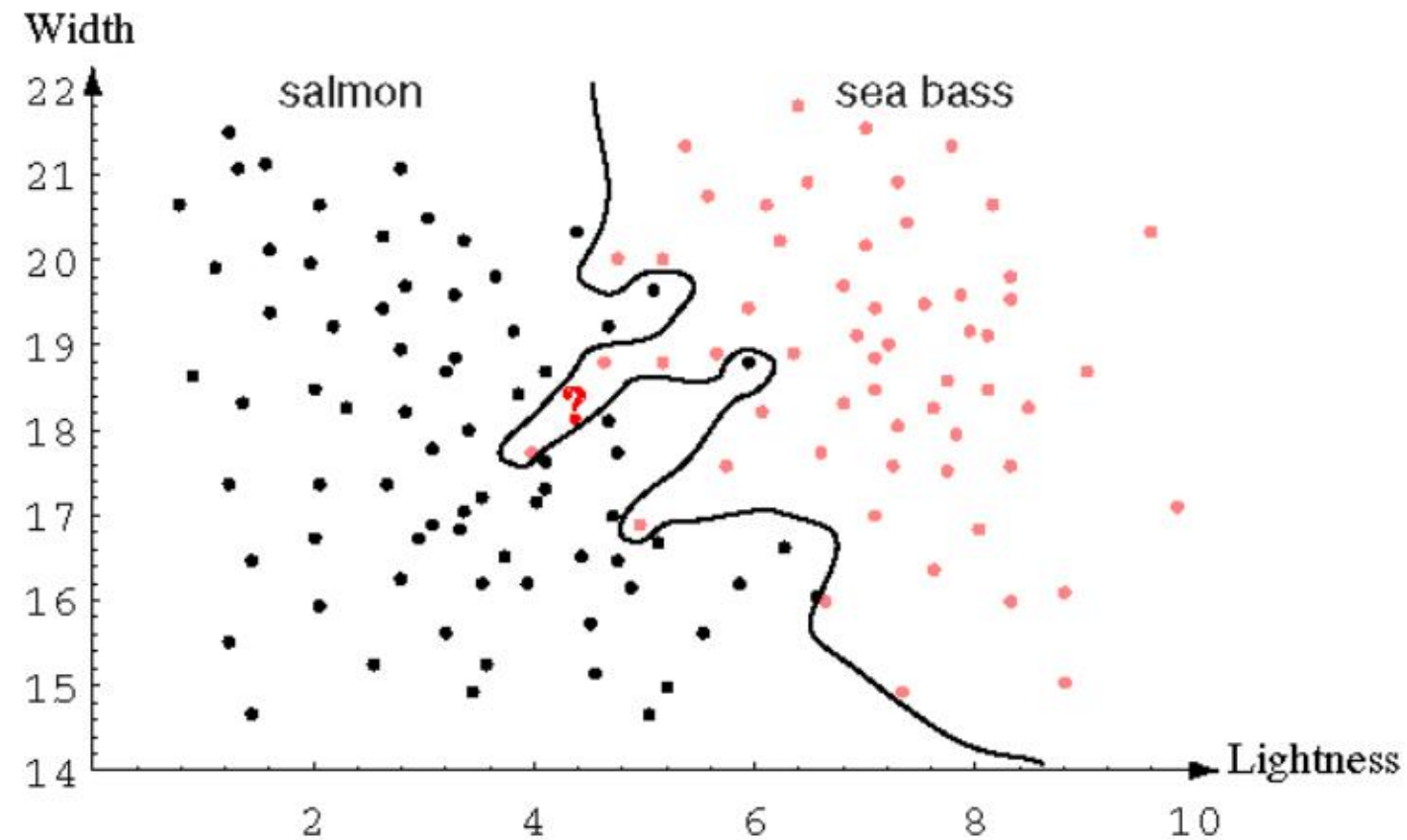
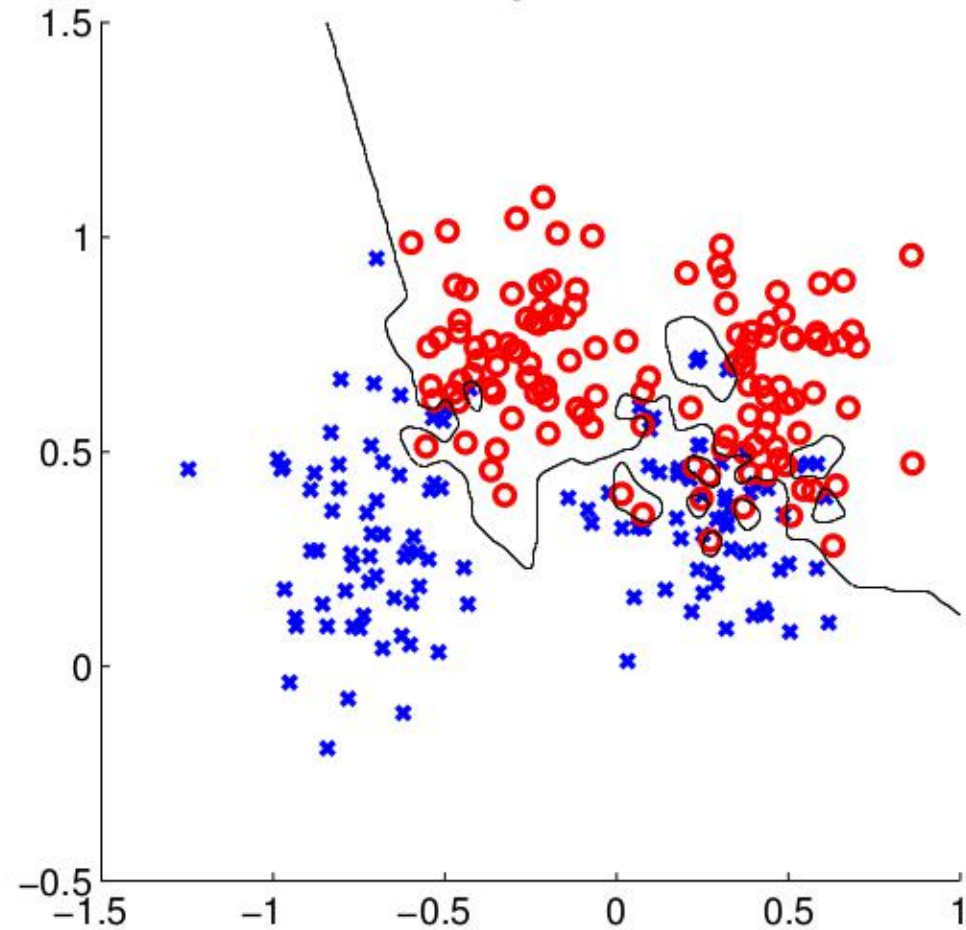
spojení dobré		82%
spojení postačující		517%
nedostatečné		00%

# $K$ -Nearest neighbors classification

For a query  $\vec{x}$ :

- ▶ Find  $K$  nearest  $\vec{x}$  from the training (labeled) data.
- ▶ Classify to the class with the most exemplars in the set above.

1-nearest neighbour classifier



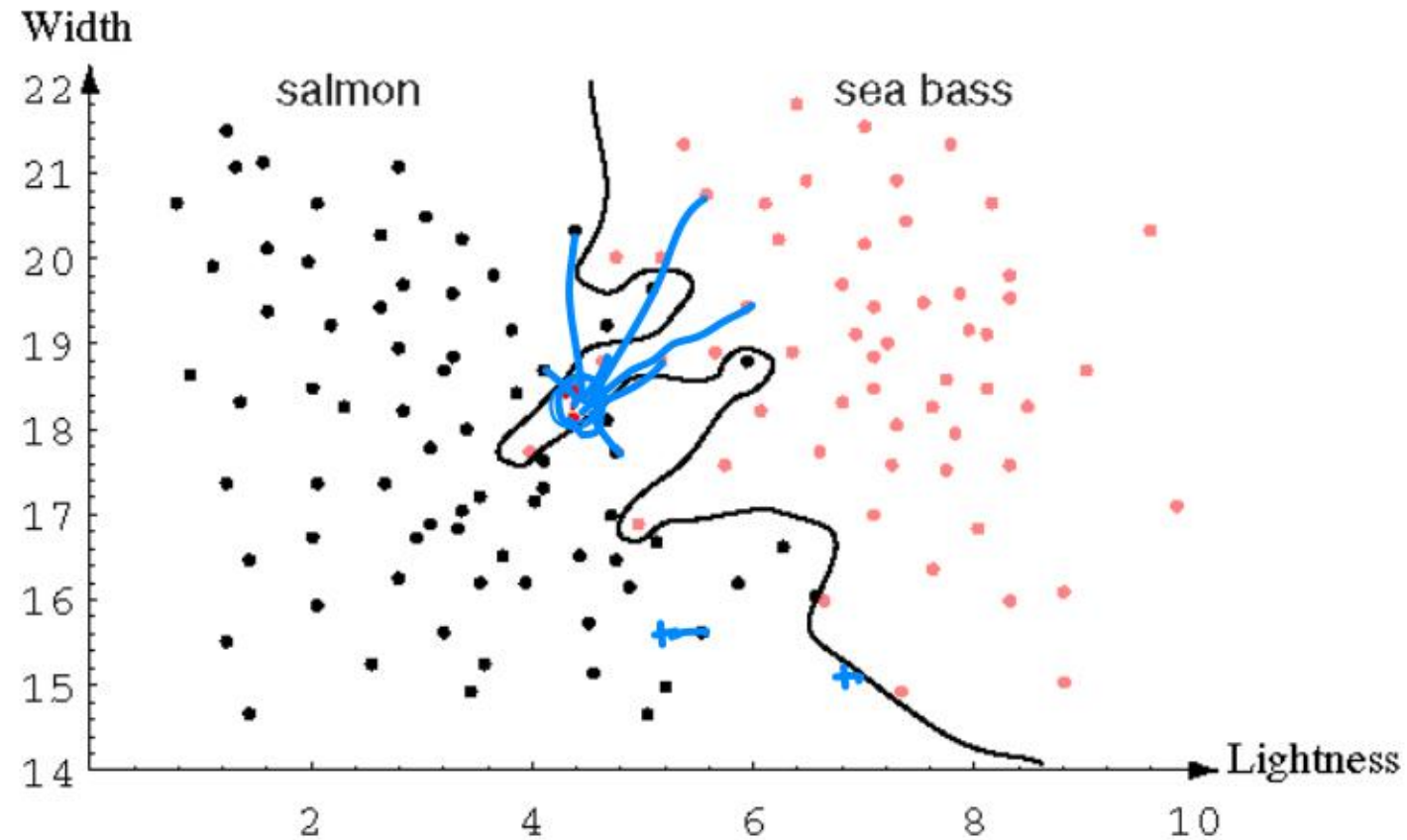
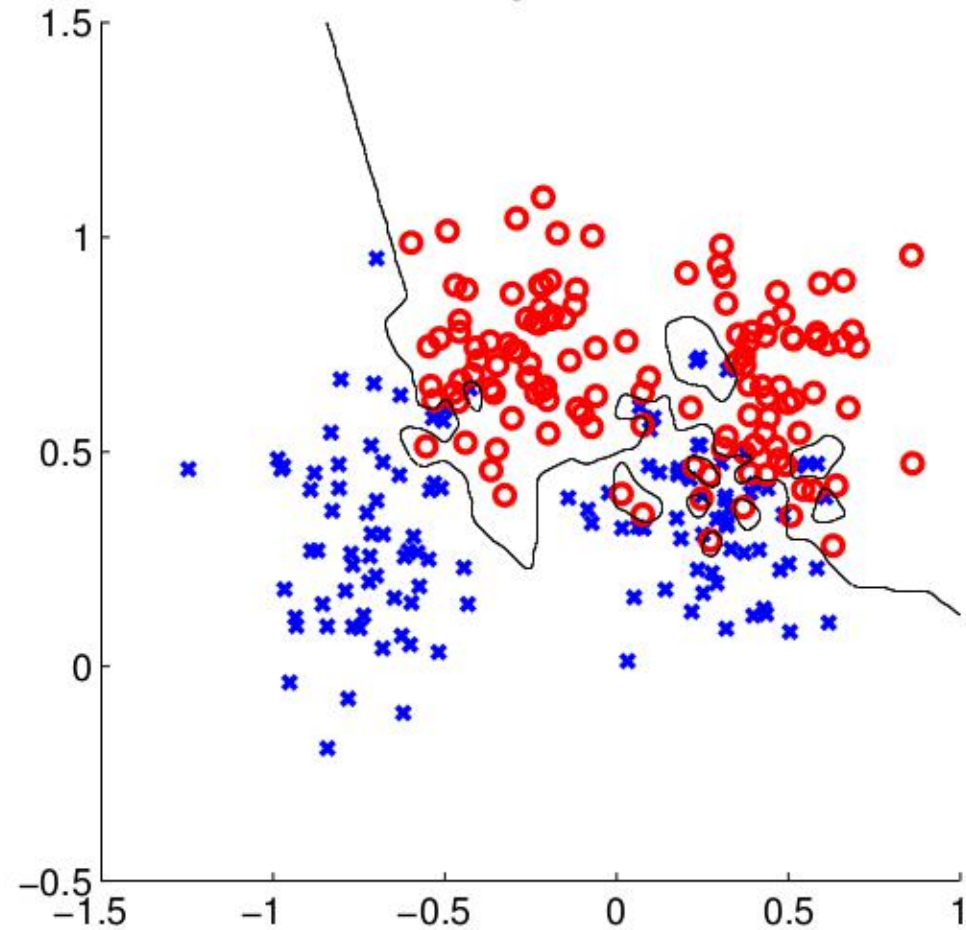


# $K$ -Nearest neighbors classification

For a query  $\vec{x}$ :

- ▶ Find  $K$  nearest  $\vec{x}$  from the training (labeled) data.
- ▶ Classify to the class with the most exemplars in the set above.

1-nearest neighbour classifier



*Baseline alg*

## $K$ – Nearest Neighbor and Bayes $j^* = \operatorname{argmax}_j P(s_j|\vec{x})$

Assume data:

- ▶  $N$  points  $\vec{x}$  in total.
- ▶  $N_j$  points in  $s_j$  class. Hence,  $\sum_j N_j = N$ .

We want classify  $\vec{x}$ . We draw a sphere centered at  $\vec{x}$  containing  $K$  points irrespective of class.

$V$  is the volume of this sphere.  $P(s_j|\vec{x}) = ?$

$$P(s_j|\vec{x}) = \frac{P(\vec{x}|s_j)P(s_j)}{P(\vec{x})}$$

$$P(s_j) = \frac{N_j}{N}$$

$$P(\vec{x}) = \frac{K}{NV}$$

$$P(\vec{x}|s_j) = \frac{K_j}{N_j V}$$

$$P(s_j|\vec{x}) = \frac{P(\vec{x}|s_j)P(s_j)}{P(\vec{x})} = \frac{K_j}{K}$$

## $K$ – Nearest Neighbor and Bayes $j^* = \operatorname{argmax}_j P(s_j|\vec{x})$

Assume data:

- ▶  $N$  points  $\vec{x}$  in total.
- ▶  $N_j$  points in  $s_j$  class. Hence,  $\sum_j N_j = N$ .

$\vec{x}$  – query in  $\mu$ /data

We want classify  $\vec{x}$ . We draw a sphere centered at  $\vec{x}$  containing  $K$  points irrespective of class.

$V$  is the volume of this sphere.  $P(s_j|\vec{x}) = ?$

$$P(s_j|\vec{x}) = \frac{P(\vec{x}|s_j)P(s_j)}{P(\vec{x})}$$

$$P(s_j) = \frac{N_j}{N}$$

$$P(\vec{x}) = \frac{K}{NV}$$

$$P(\vec{x}|s_j) = \frac{K_j}{N_j V}$$

$$P(s_j|\vec{x}) = \frac{P(\vec{x}|s_j)P(s_j)}{P(\vec{x})} = \frac{K_j}{K}$$

## $K$ – Nearest Neighbor and Bayes $j^* = \operatorname{argmax}_j P(s_j|\vec{x})$

Assume data:

- ▶  $N$  points  $\vec{x}$  in total.
- ▶  $N_j$  points in  $s_j$  class. Hence,  $\sum_j N_j = N$ .

We want classify  $\vec{x}$ . We draw a sphere centered at  $\vec{x}$  containing  $K$  points irrespective of class.

$V$  is the volume of this sphere.  $P(s_j|\vec{x}) = ?$

$$P(s_j|\vec{x}) = \frac{P(\vec{x}|s_j)P(s_j)}{P(\vec{x})}$$

$$P(s_j) = \frac{N_j}{N}$$

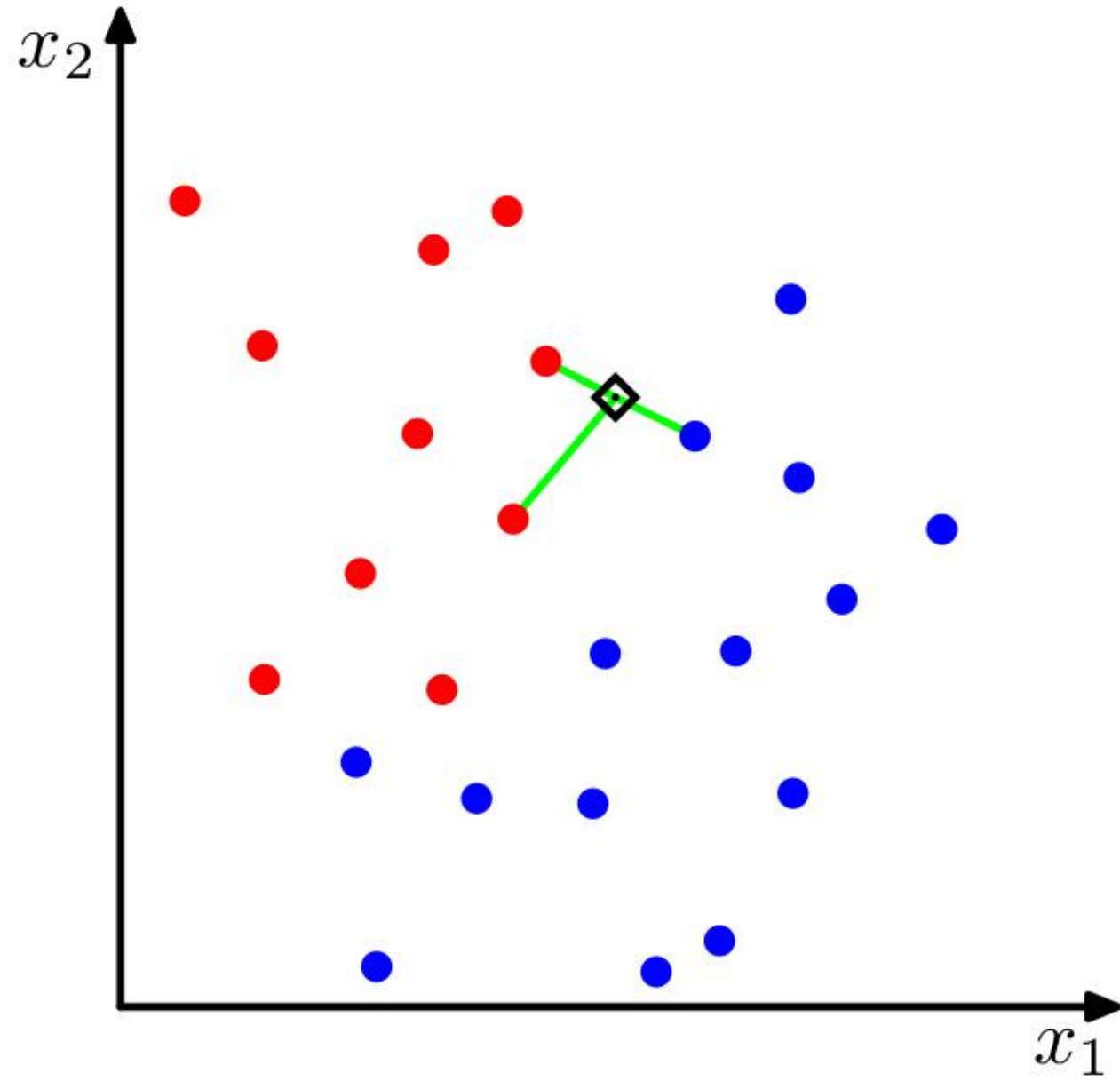
$$P(\vec{x}) = \frac{K}{NV}$$

$$P(\vec{x}|s_j) = \frac{K_j}{N_j V}$$

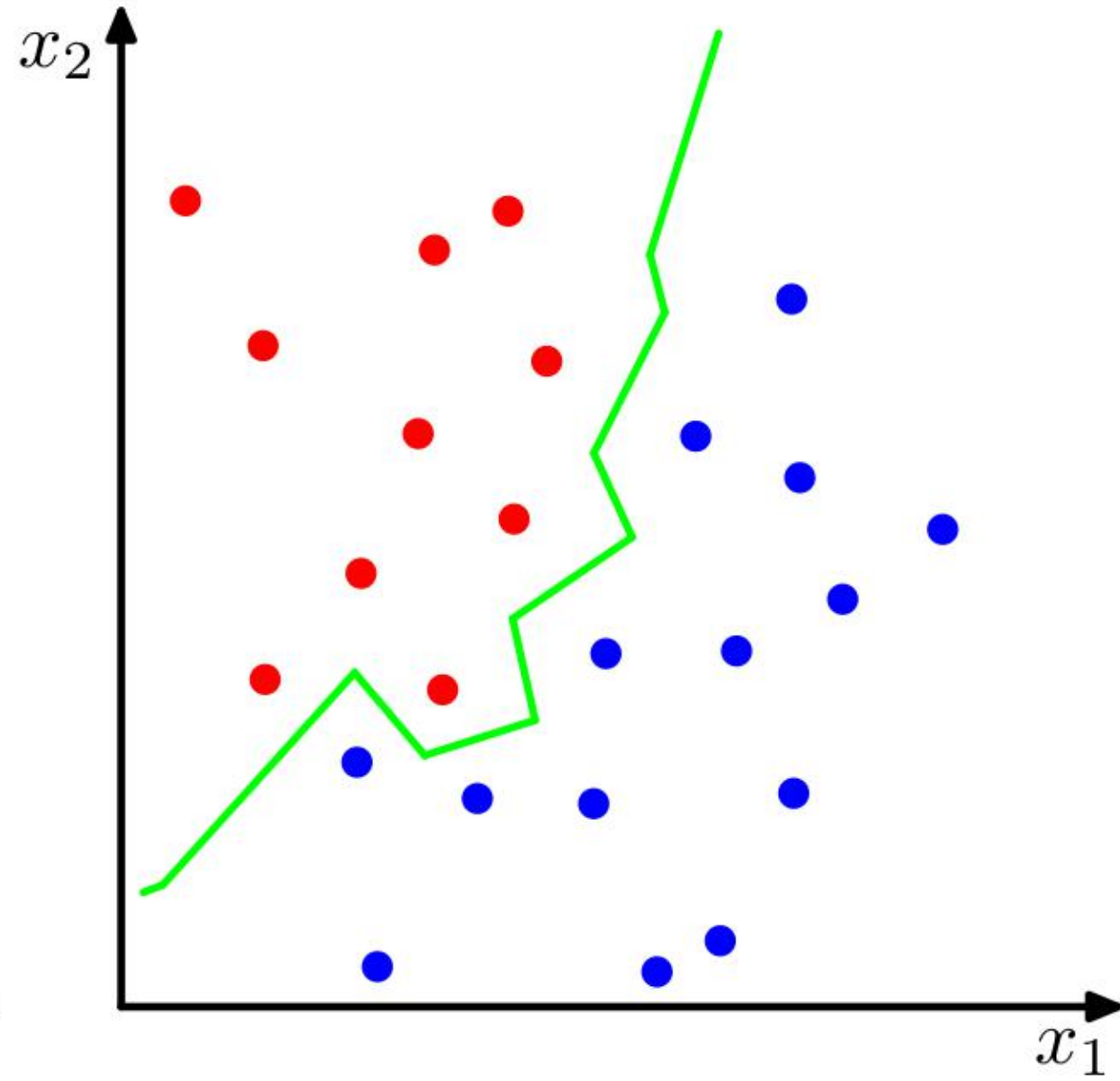
$$P(s_j|\vec{x}) = \frac{P(\vec{x}|s_j)P(s_j)}{P(\vec{x})} = \frac{K_j}{K}$$



# NN classification example



(a)

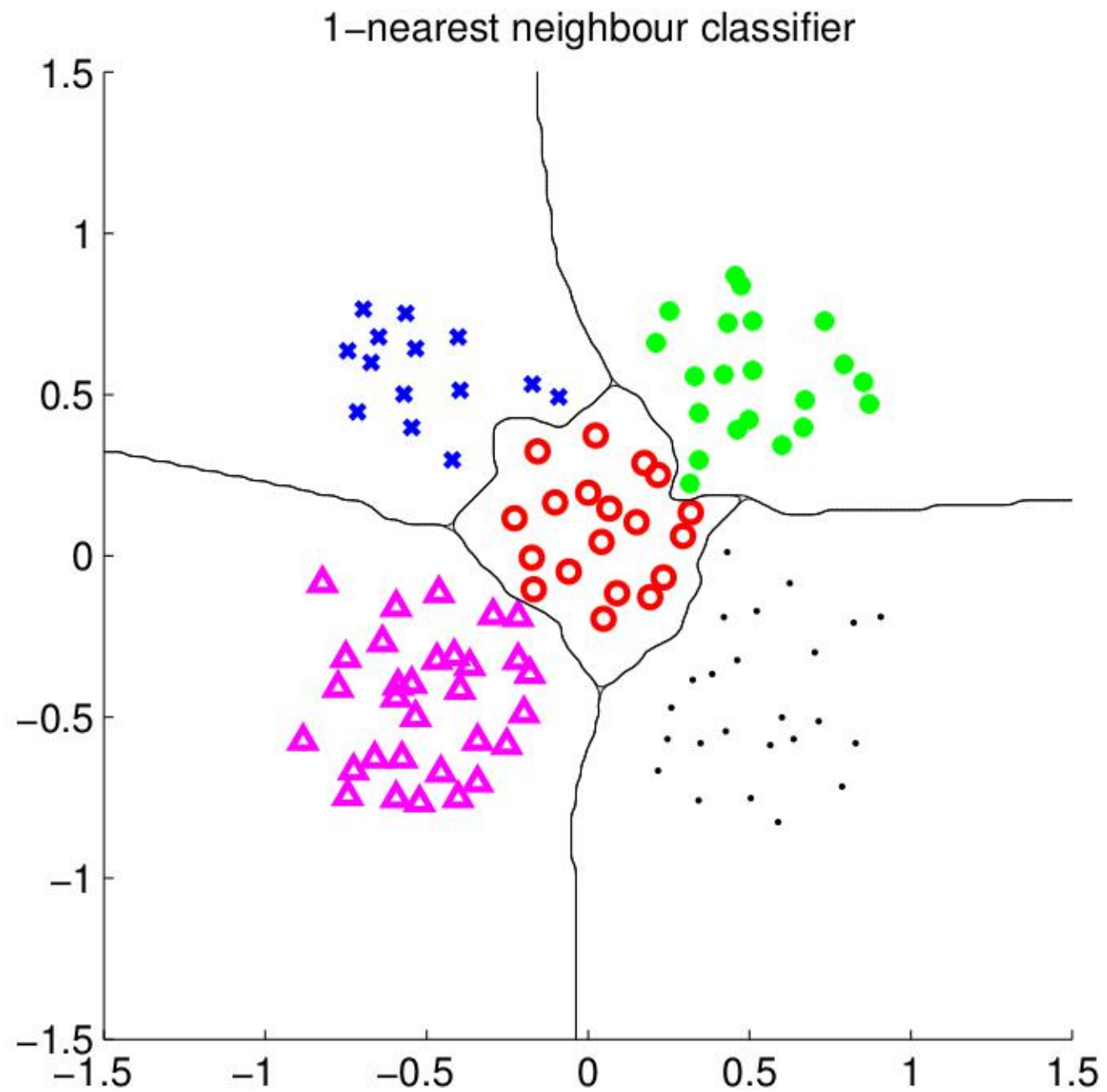
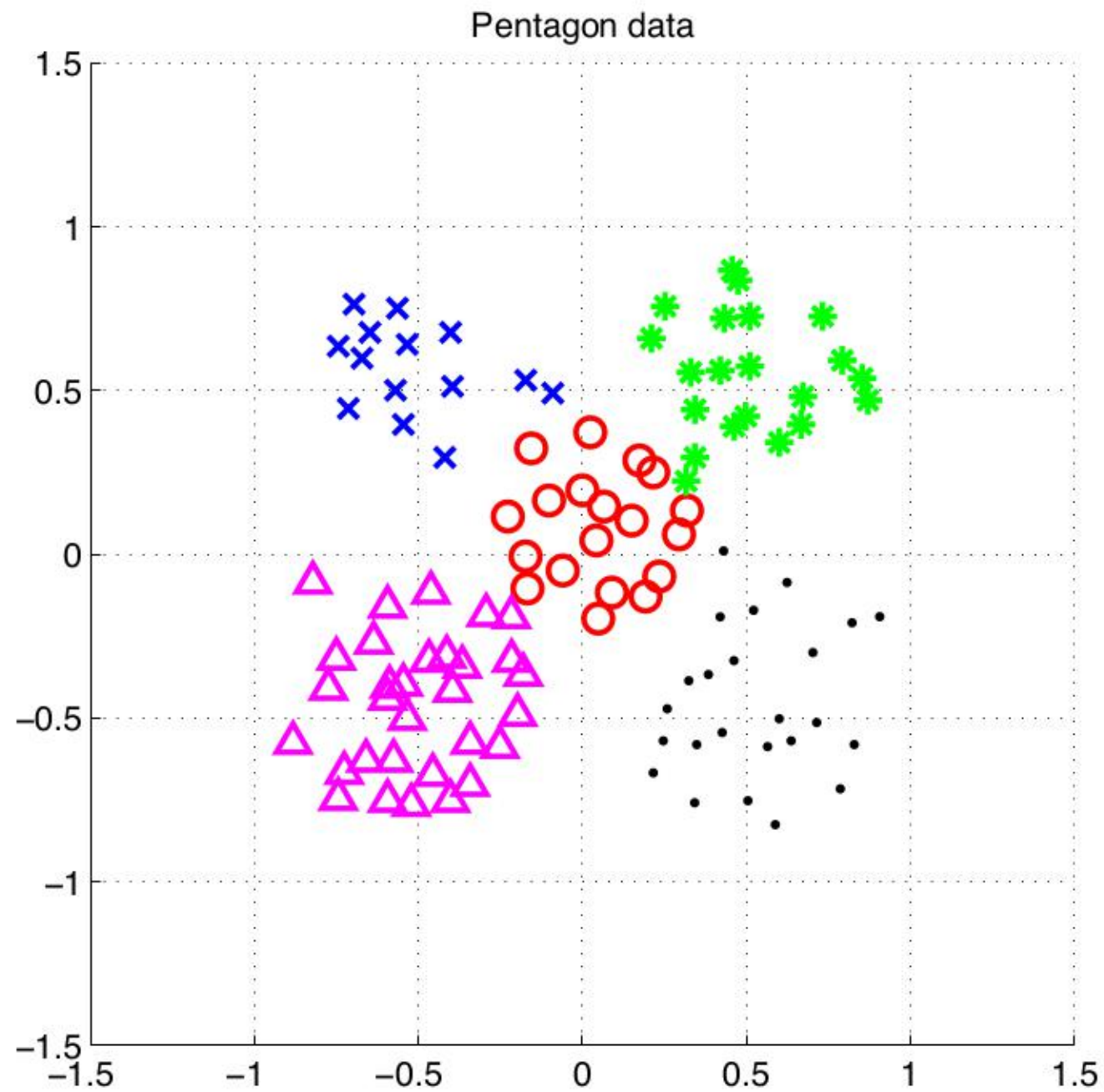


(b)

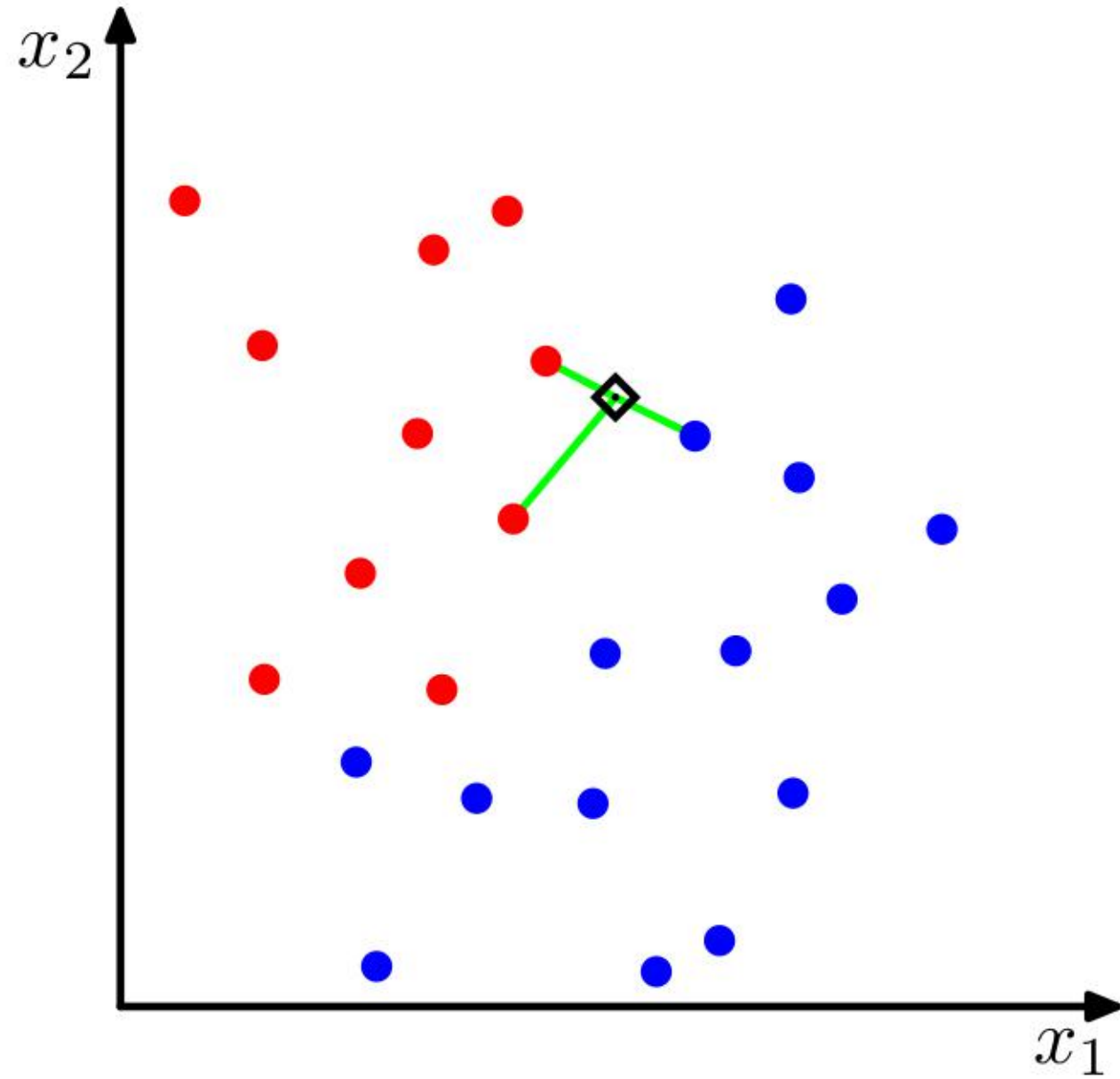
<sup>1</sup>Figs from [1]



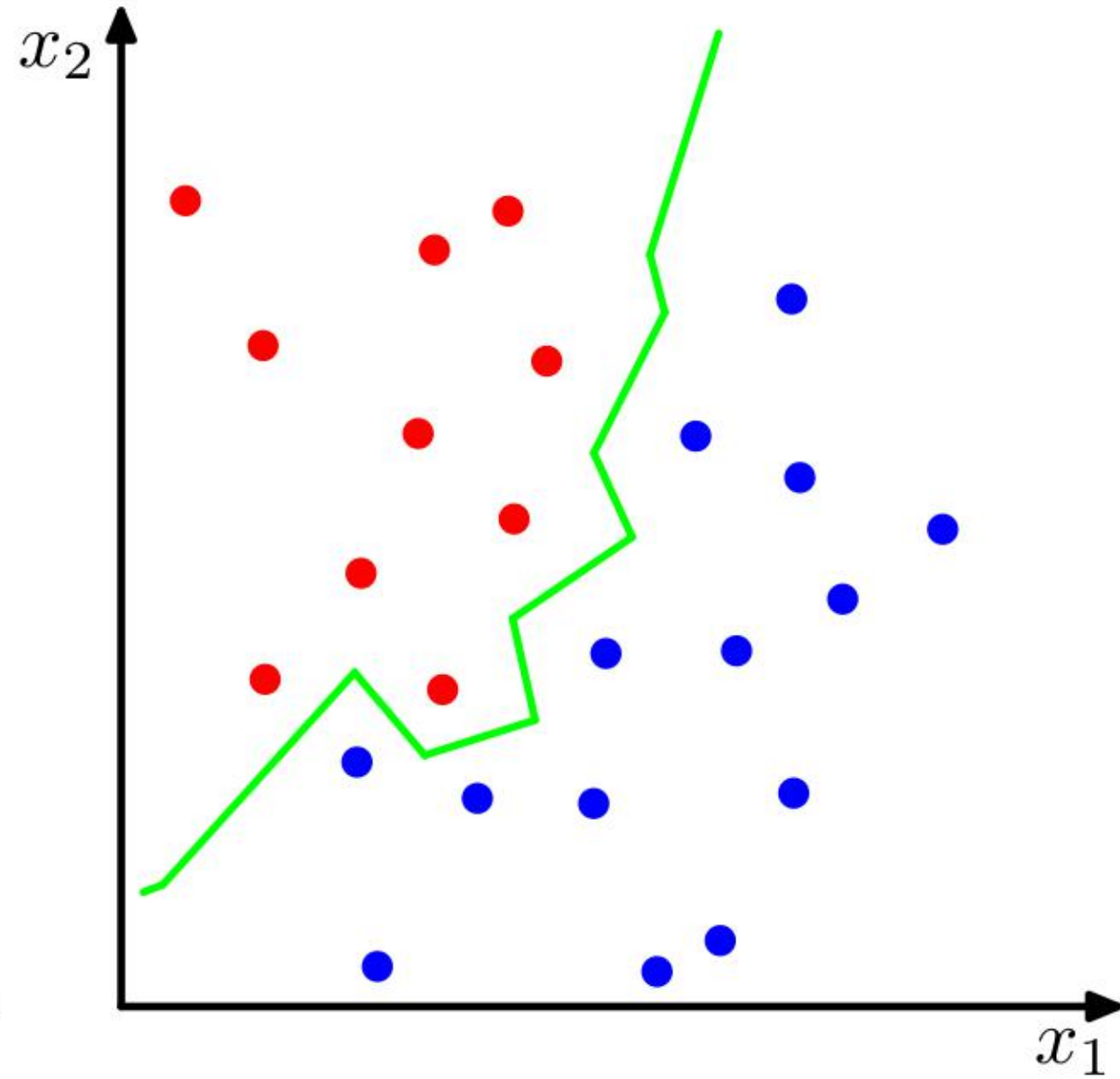
# NN classification example



# NN classification example



(a)



(b)

<sup>1</sup>Figs from [1]

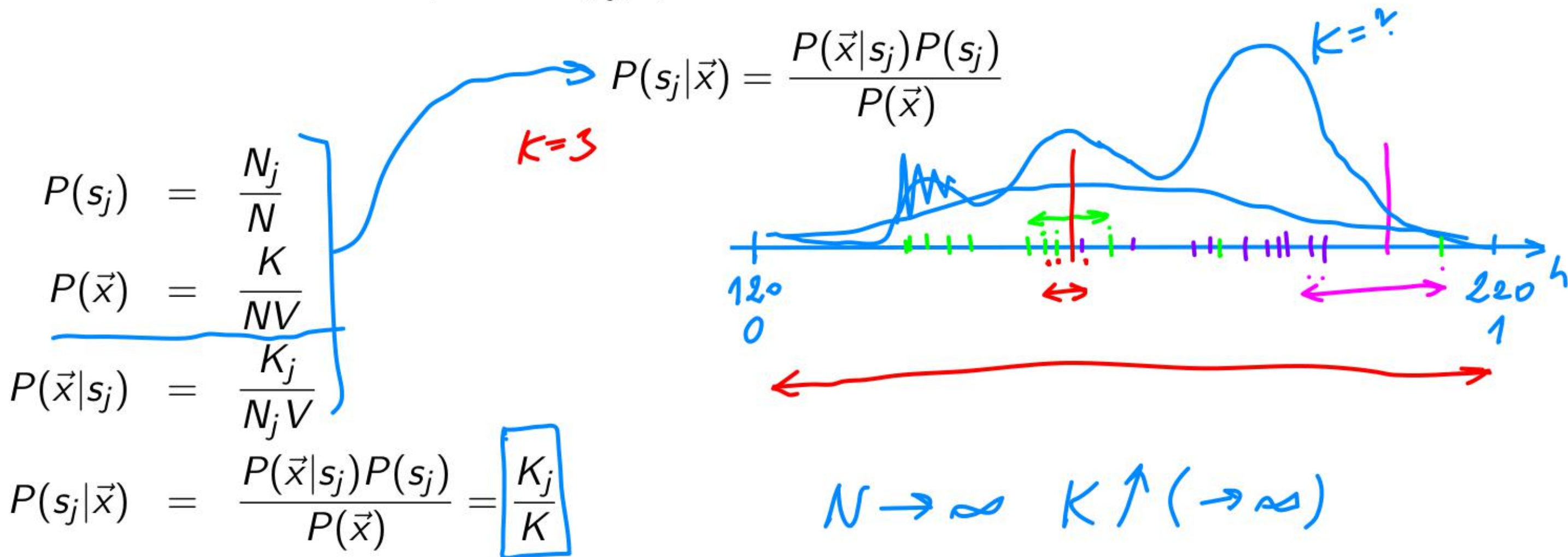
# $K$ – Nearest Neighbor and Bayes $j^* = \operatorname{argmax}_j P(s_j|\vec{x})$

Assume data: (training)

- ▶  $N$  points  $\vec{x}$  in total.
- ▶  $N_j$  points in  $s_j$  class. Hence,  $\sum_j N_j = N$ .

We want classify  $\vec{x}$ . We draw a sphere centered at  $\vec{x}$  containing  $K$  points irrespective of class.

$V$  is the volume of this sphere.  $P(s_j|\vec{x}) = ?$





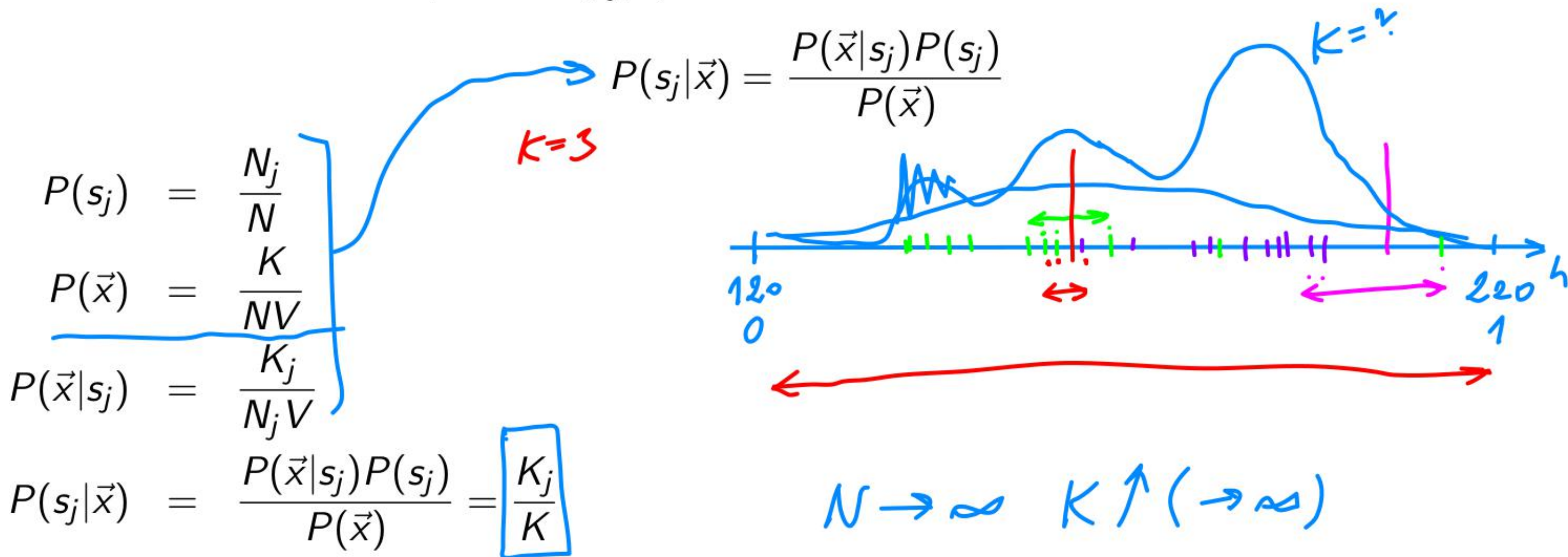
# $K$ – Nearest Neighbor and Bayes $j^* = \operatorname{argmax}_j P(s_j|\vec{x})$

Assume data: (training)

- ▶  $N$  points  $\vec{x}$  in total.
- ▶  $N_j$  points in  $s_j$  class. Hence,  $\sum_j N_j = N$ .

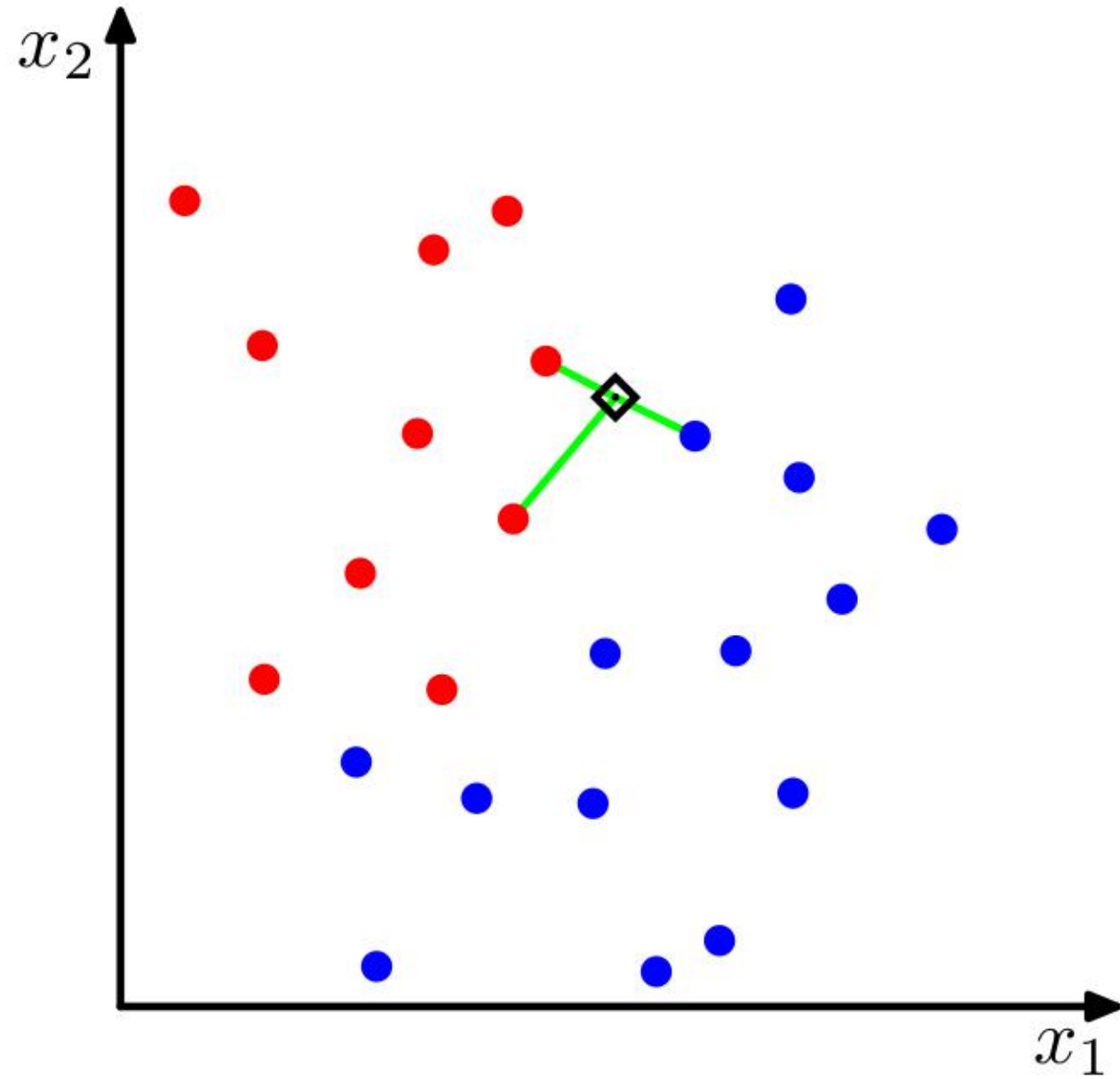
We want classify  $\vec{x}$ . We draw a sphere centered at  $\vec{x}$  containing  $K$  points irrespective of class.

$V$  is the volume of this sphere.  $P(s_j|\vec{x}) = ?$

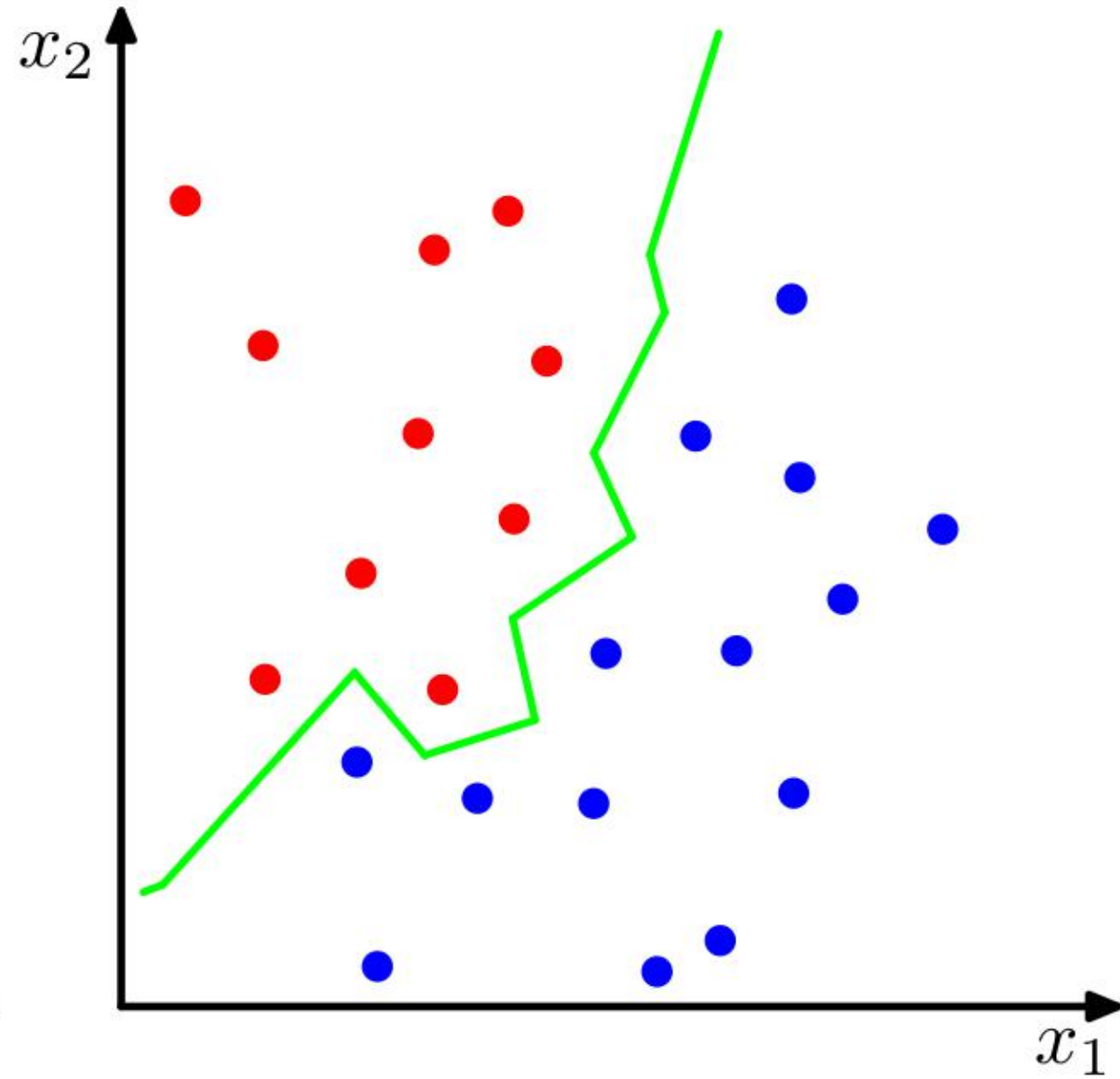




# NN classification example



(a)

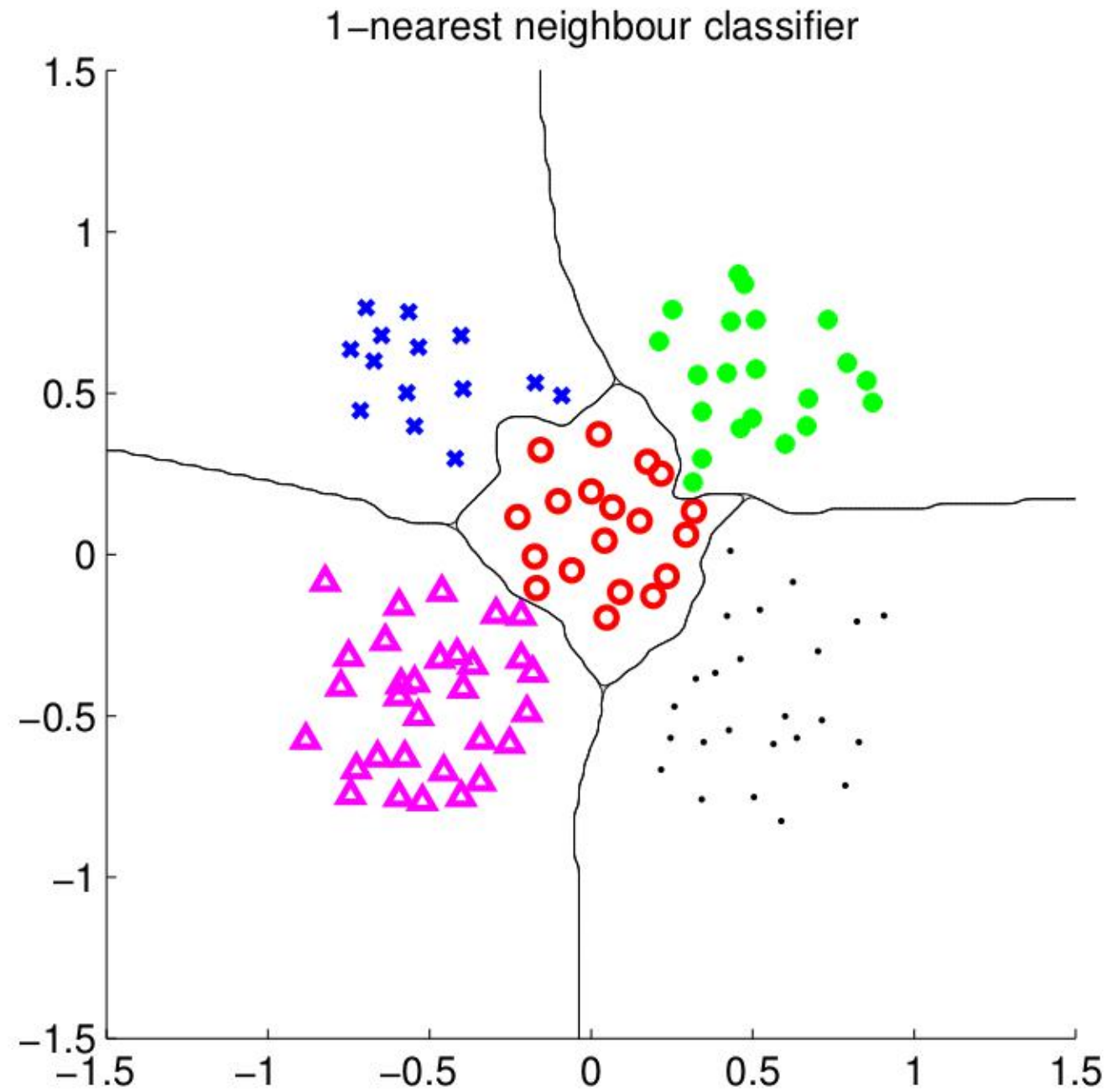
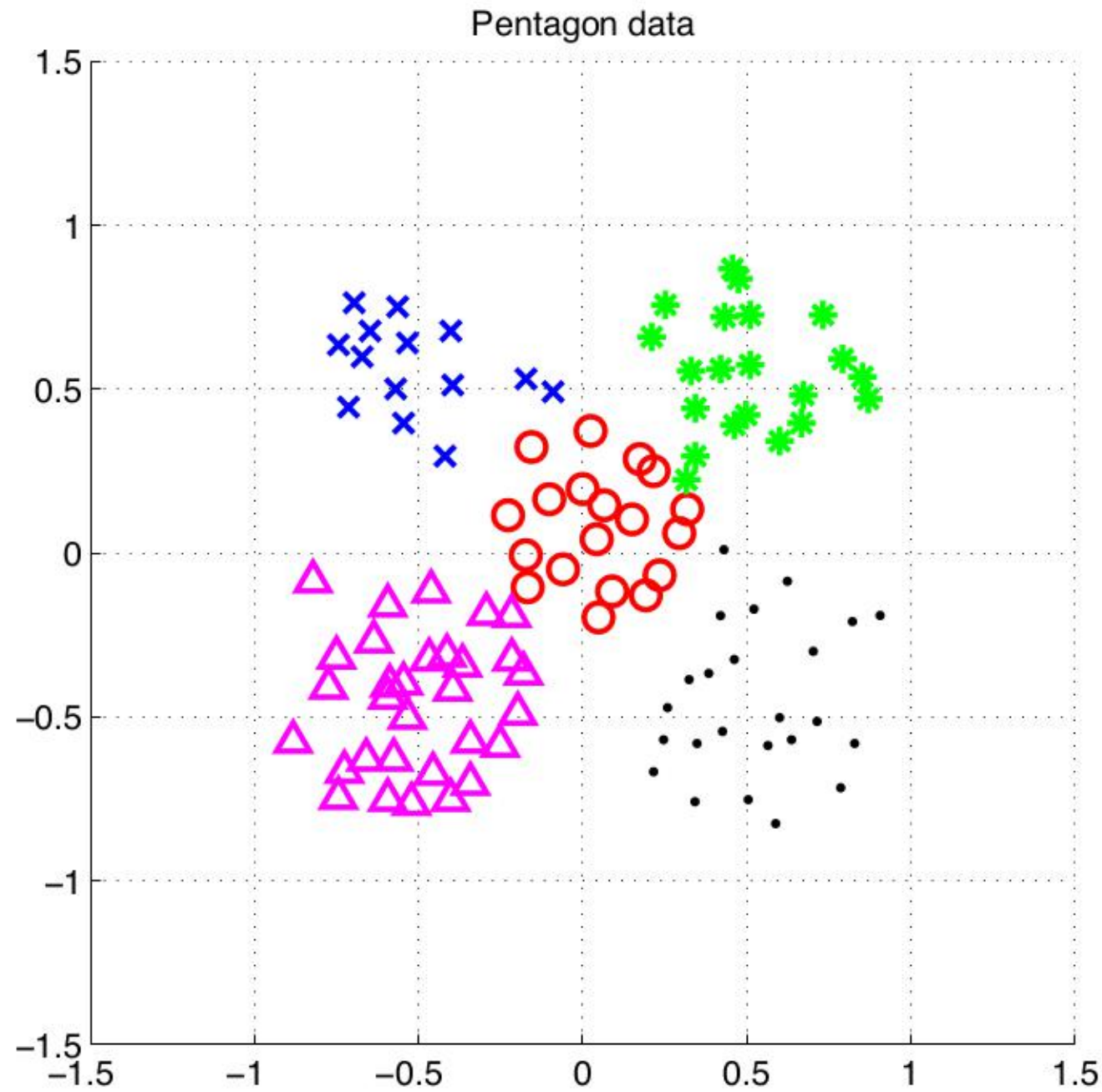


(b)

---

<sup>1</sup>Figs from [1]

# NN classification example



## What is *nearest*? Metrics for NN classification . . .

A function  $D$  which is: nonnegative, reflexive, symmetrical, satisfying triangle inequality:

$$D(\vec{a}, \vec{b}) \geq 0$$

$$D(\vec{a}, \vec{b}) = 0 \text{ iff } \vec{a} = \vec{b}$$

$$D(\vec{a}, \vec{b}) = D(\vec{b}, \vec{a})$$

$$D(\vec{a}, \vec{b}) + D(\vec{b}, \vec{c}) \geq D(\vec{a}, \vec{c})$$

## What is *nearest*? Metrics for NN classification . . .

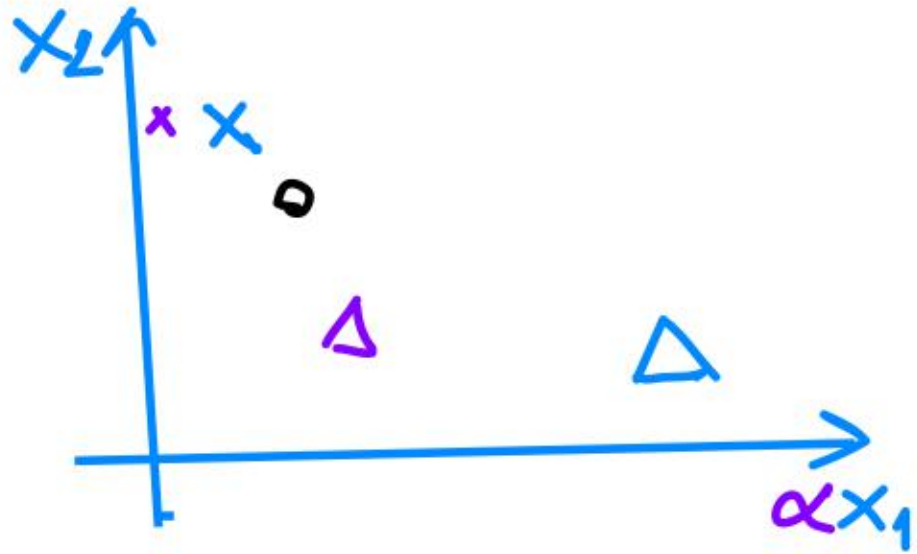
A function  $D$  which is: nonnegative, reflexive, symmetrical, satisfying triangle inequality:

$$D(\vec{a}, \vec{b}) \geq 0$$

$$D(\vec{a}, \vec{b}) = 0 \text{ iff } \vec{a} = \vec{b}$$

$$D(\vec{a}, \vec{b}) = D(\vec{b}, \vec{a})$$

$$D(\vec{a}, \vec{b}) + D(\vec{b}, \vec{c}) \geq D(\vec{a}, \vec{c})$$





## What is *nearest*? Metrics for NN classification ...

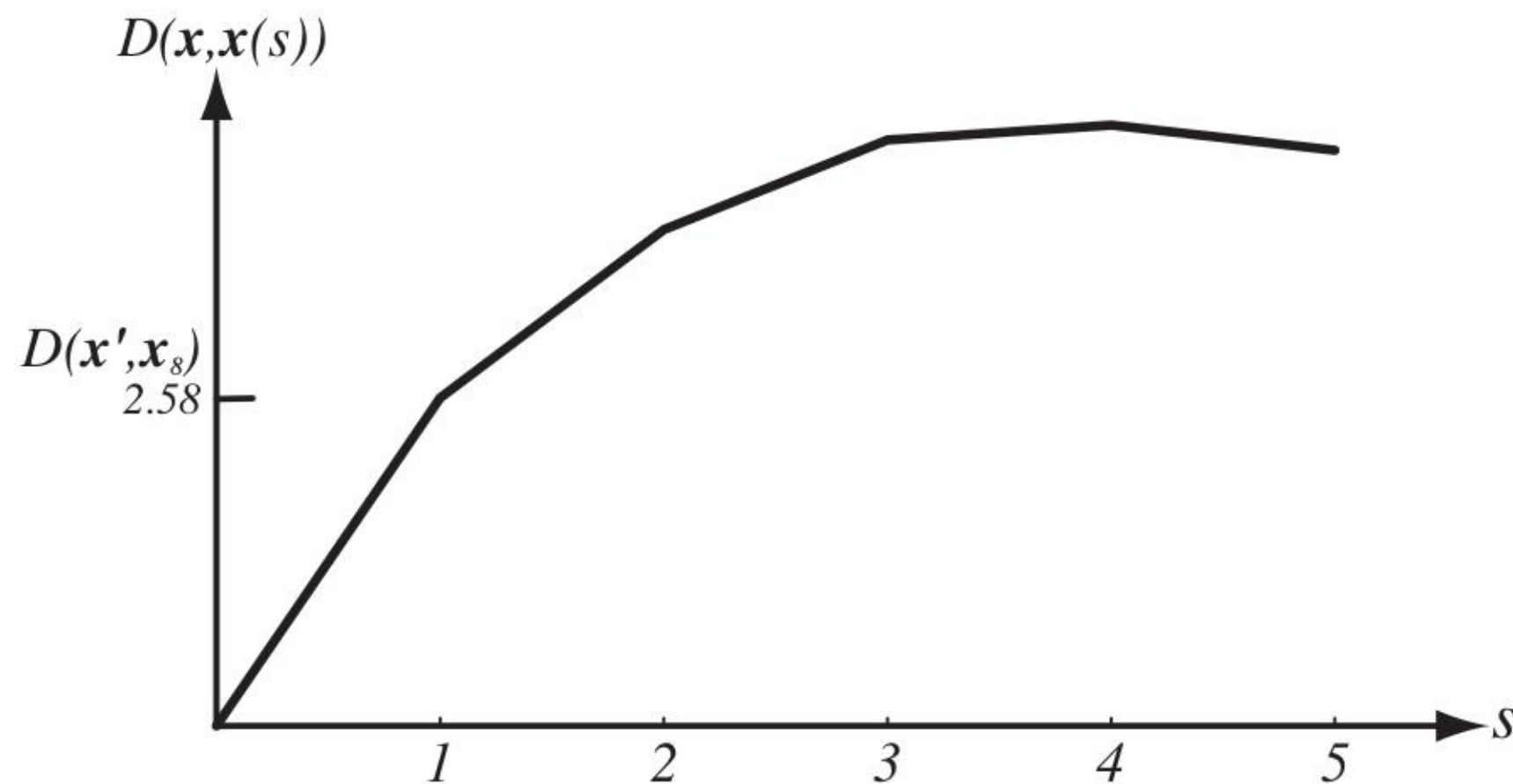
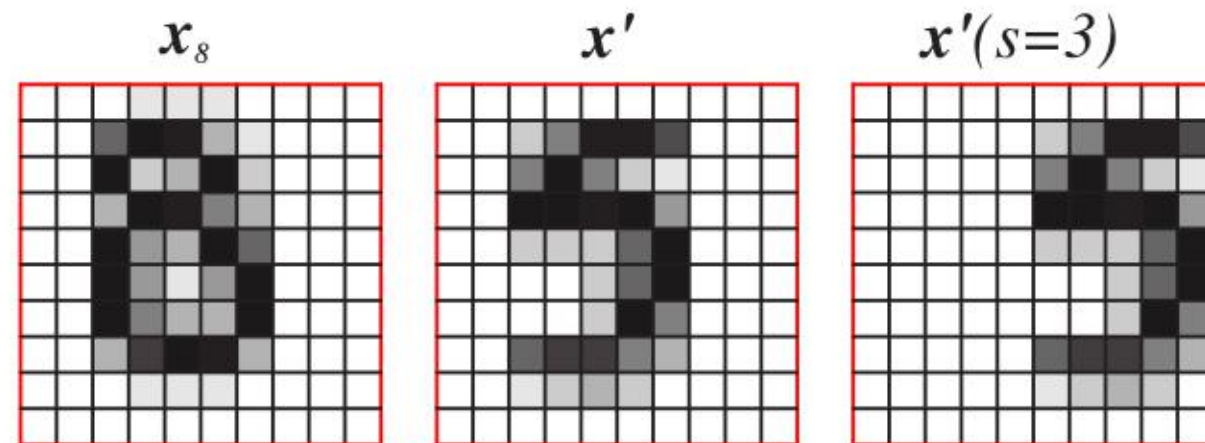
A function  $D$  which is: nonnegative, reflexive, symmetrical, satisfying triangle inequality:

$$D(\vec{a}, \vec{b}) \geq 0$$

$$D(\vec{a}, \vec{b}) = 0 \text{ iff } \vec{a} = \vec{b}$$

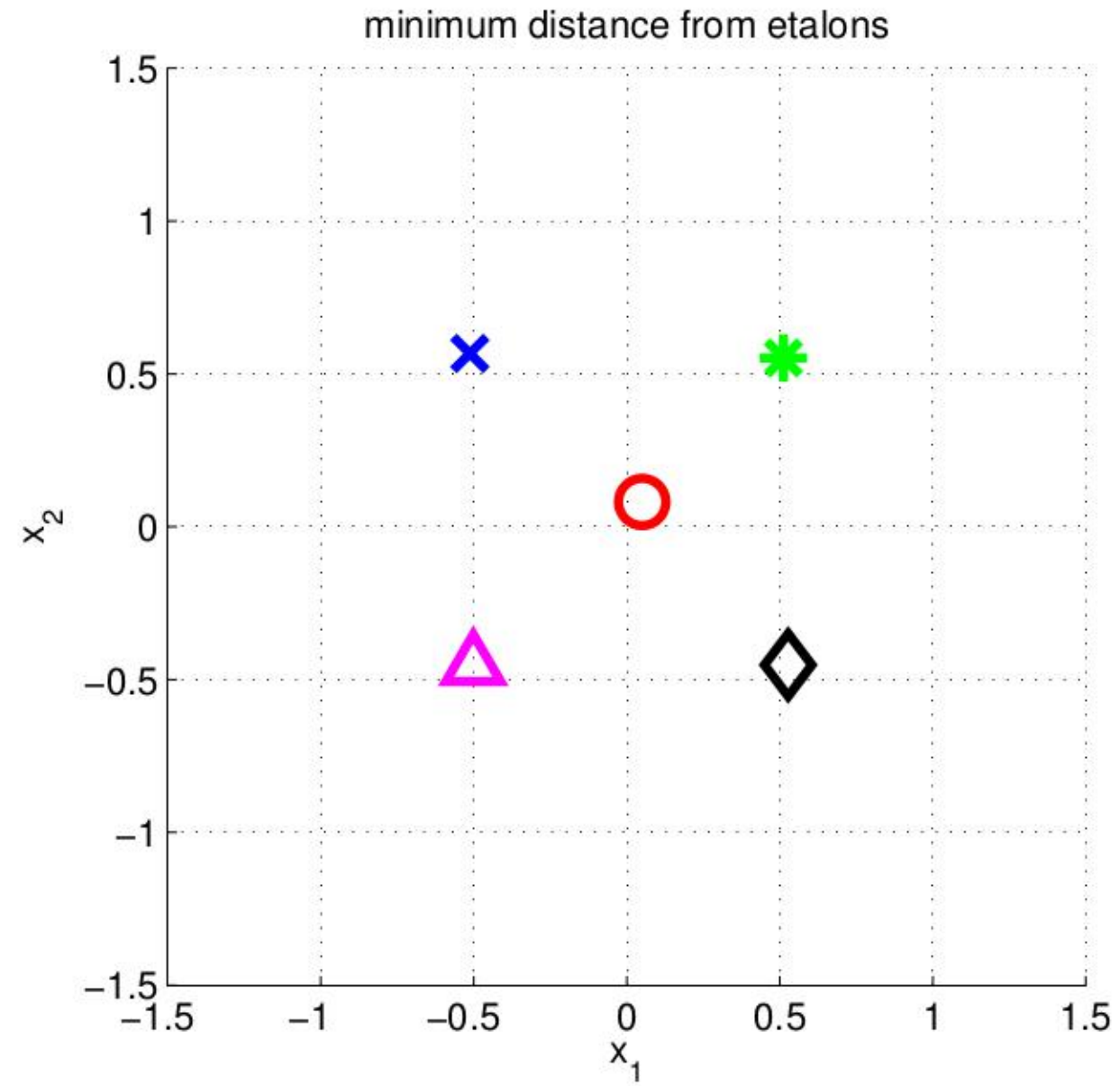
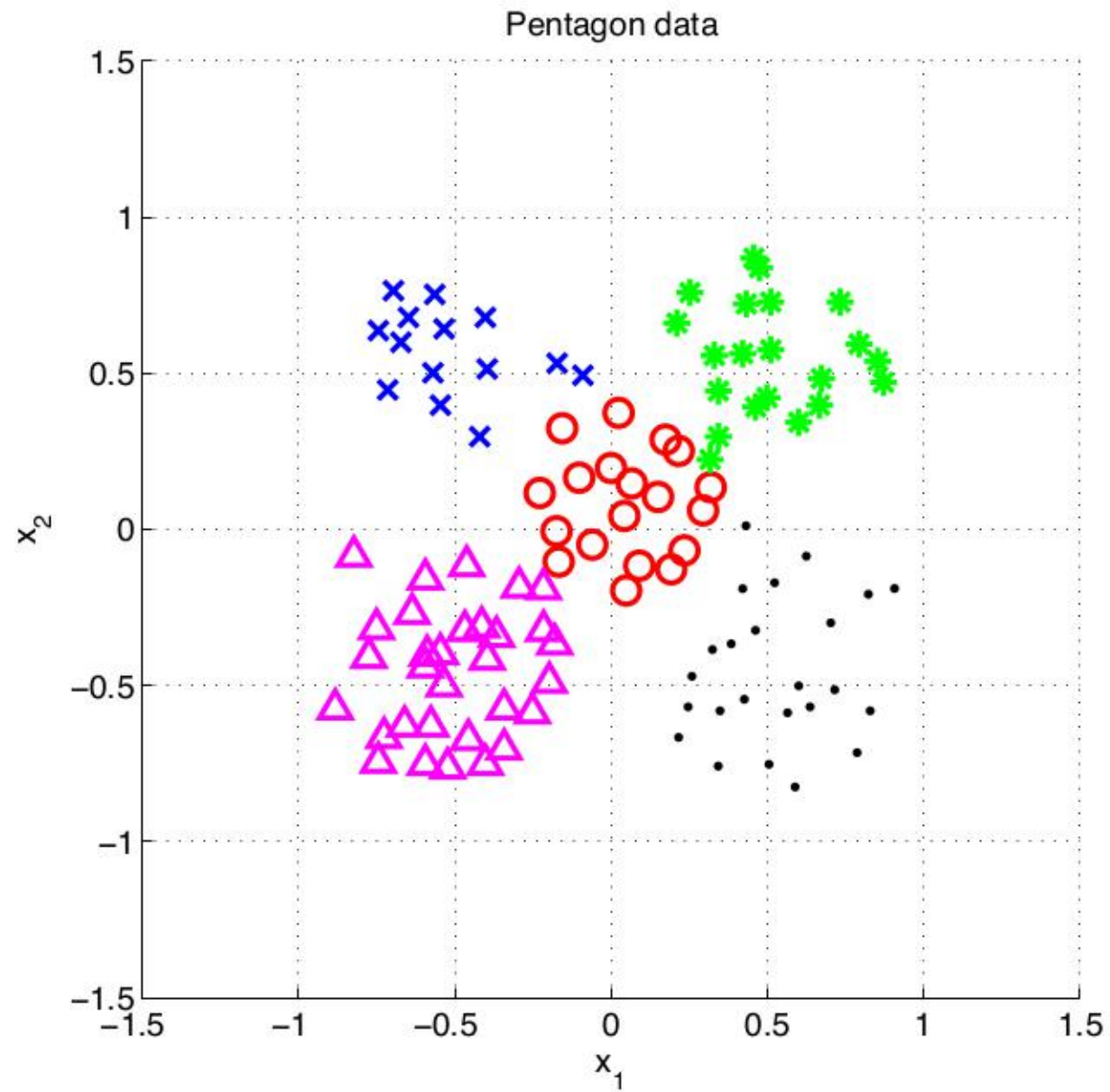
$$D(\vec{a}, \vec{b}) = D(\vec{b}, \vec{a})$$

$$D(\vec{a}, \vec{b}) + D(\vec{b}, \vec{c}) \geq D(\vec{a}, \vec{c})$$



Invariance to geometrical transformations? (figure from [2])

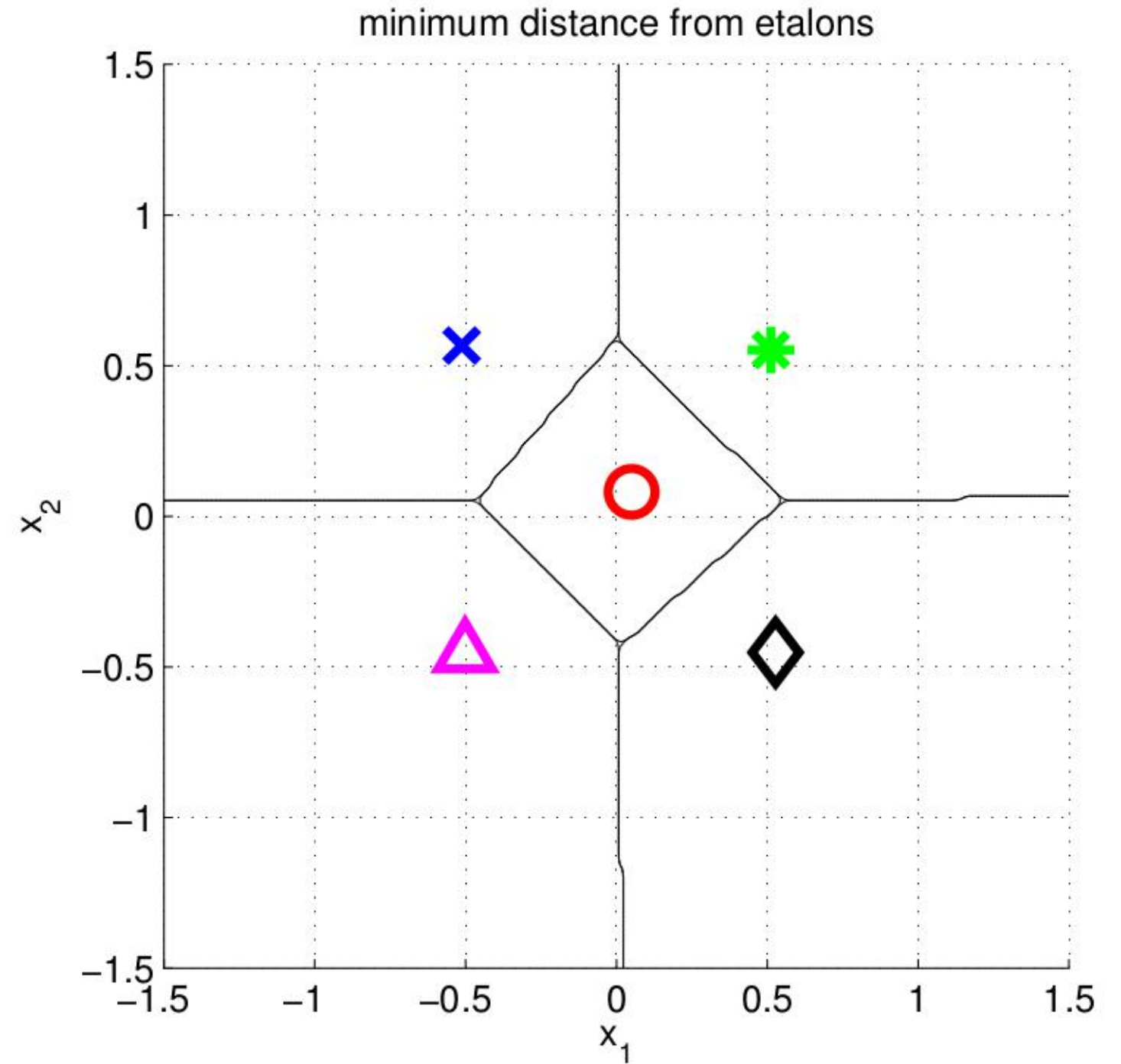
# Etalon based classification



Represent  $\vec{x}$  by **etalon** ,  $\vec{e}_s$  per each class  $s \in S$

# Separate etalons

$$s^* = \arg \min_{s \in S} \|\vec{x} - \vec{e}_s\|^2$$

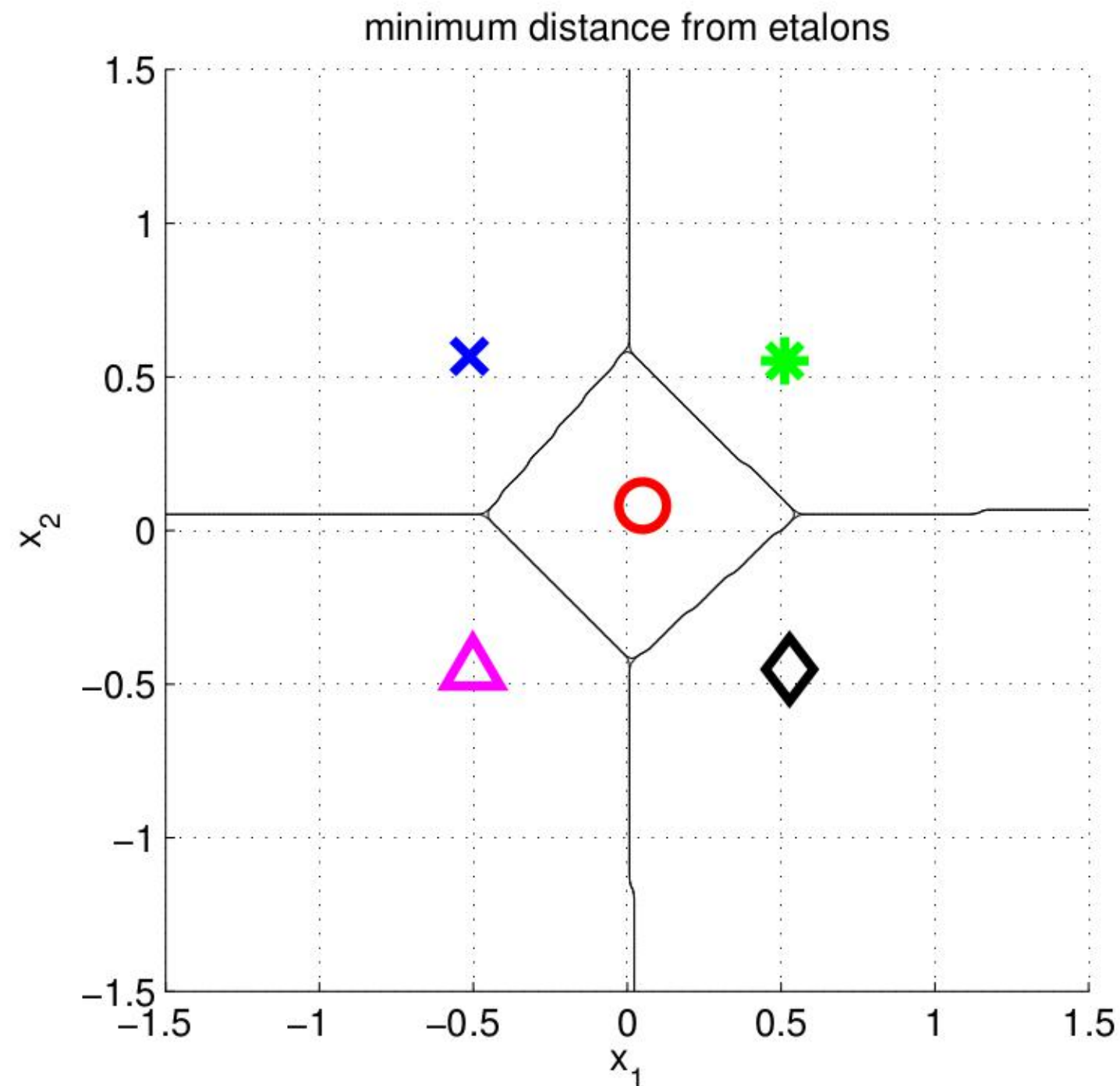


# What etalons?

If  $\mathcal{N}(\vec{x}|\vec{\mu}, \Sigma)$ ; all classes same covariance matrices, then

$$\vec{e}_s \stackrel{\text{def}}{=} \vec{\mu}_s = \frac{1}{|\mathcal{X}^s|} \sum_{i \in \mathcal{X}^s} \vec{x}_i^s$$

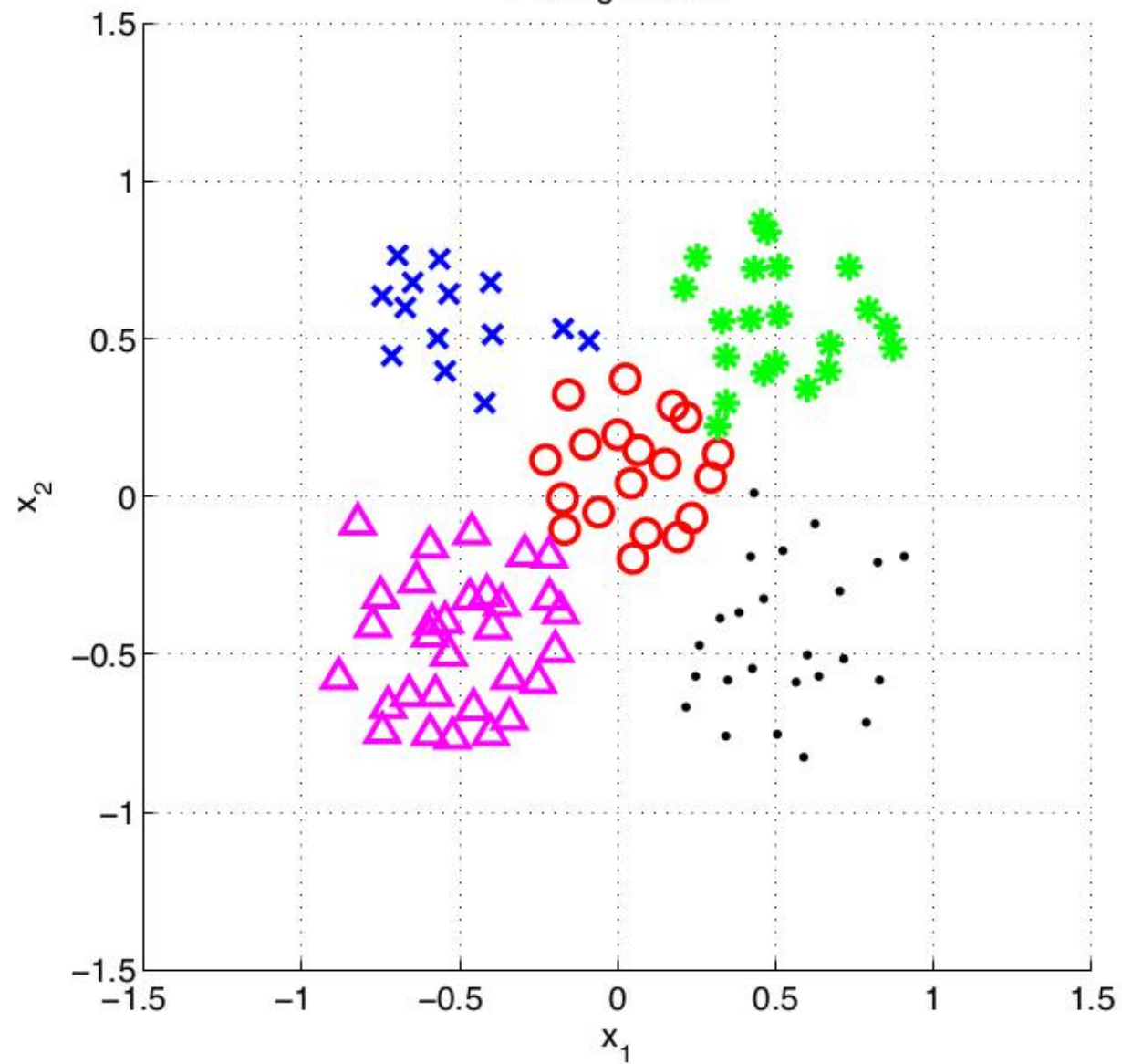
and separating hyperplanes halve distances between pairs.



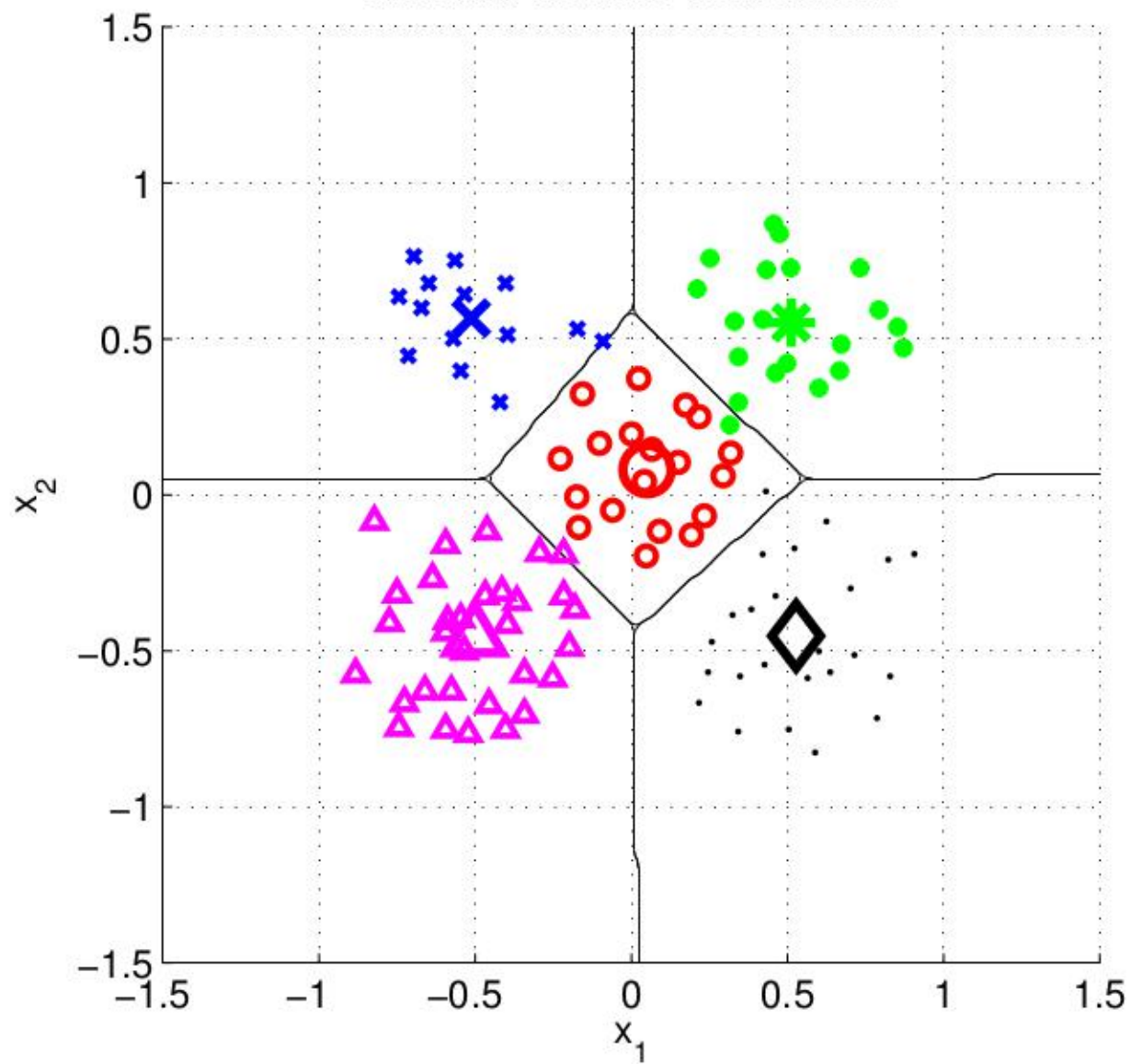


# Etalon based classification, $\vec{e}_s = \vec{\mu}_s$

Pentagon data

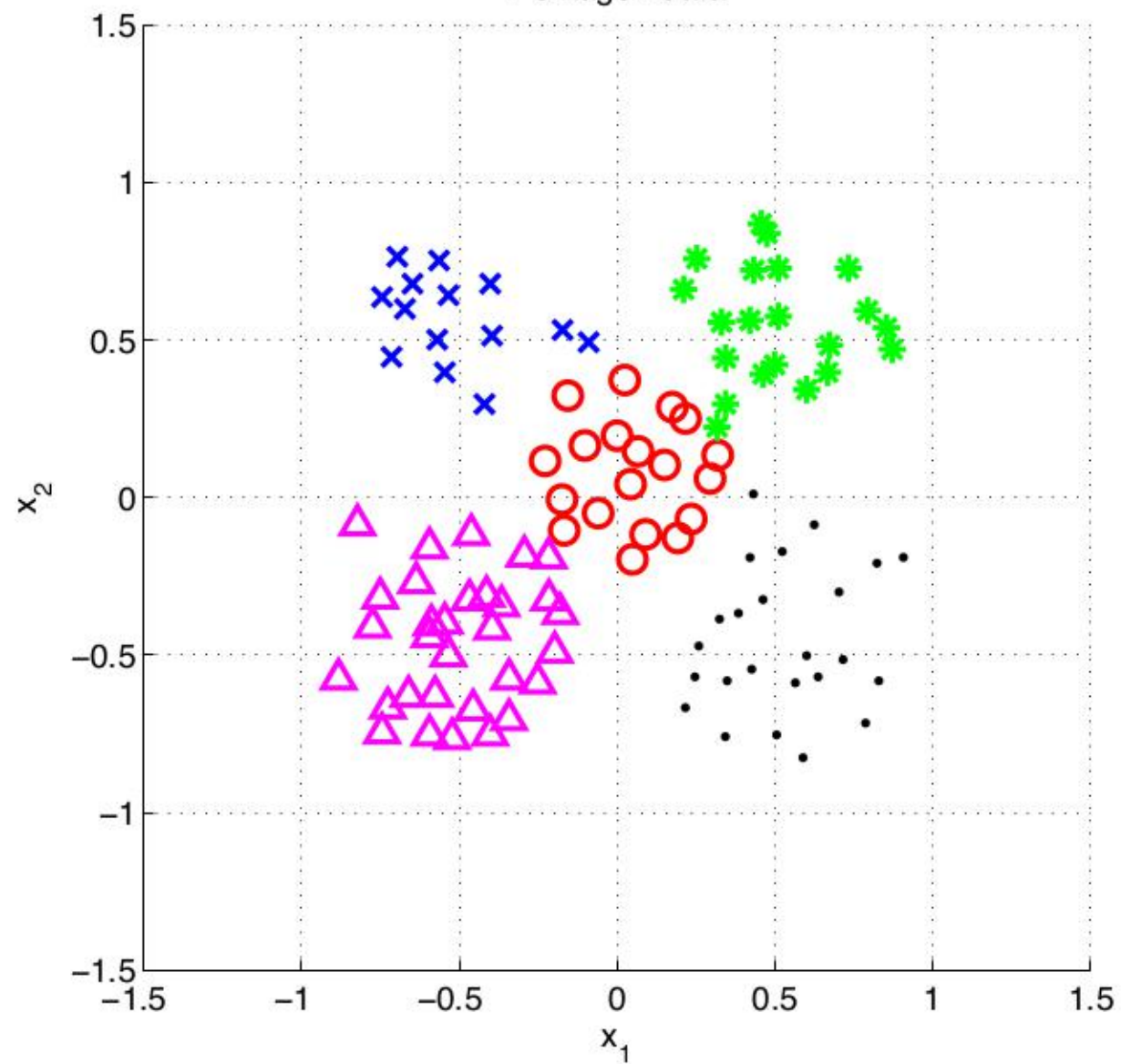


minimum distance from etalons

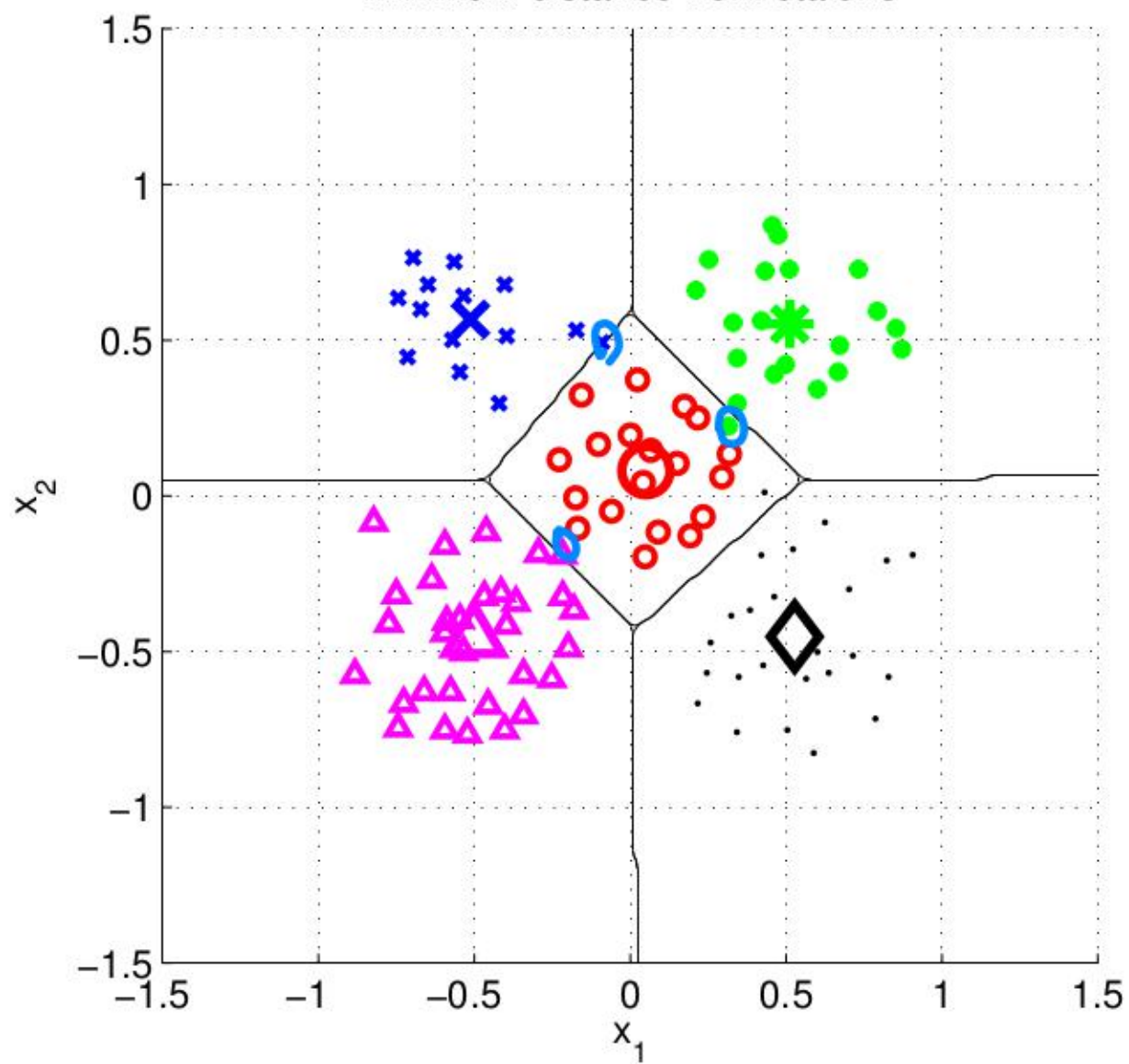


# Etalon based classification, $\vec{e}_s = \vec{\mu}_s$

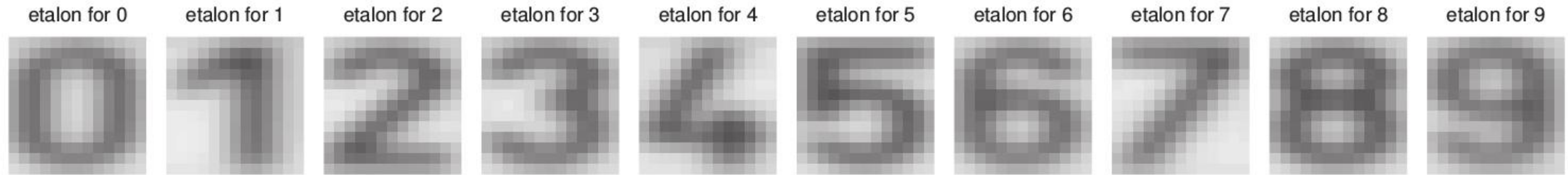
Pentagon data



minimum distance from etalons

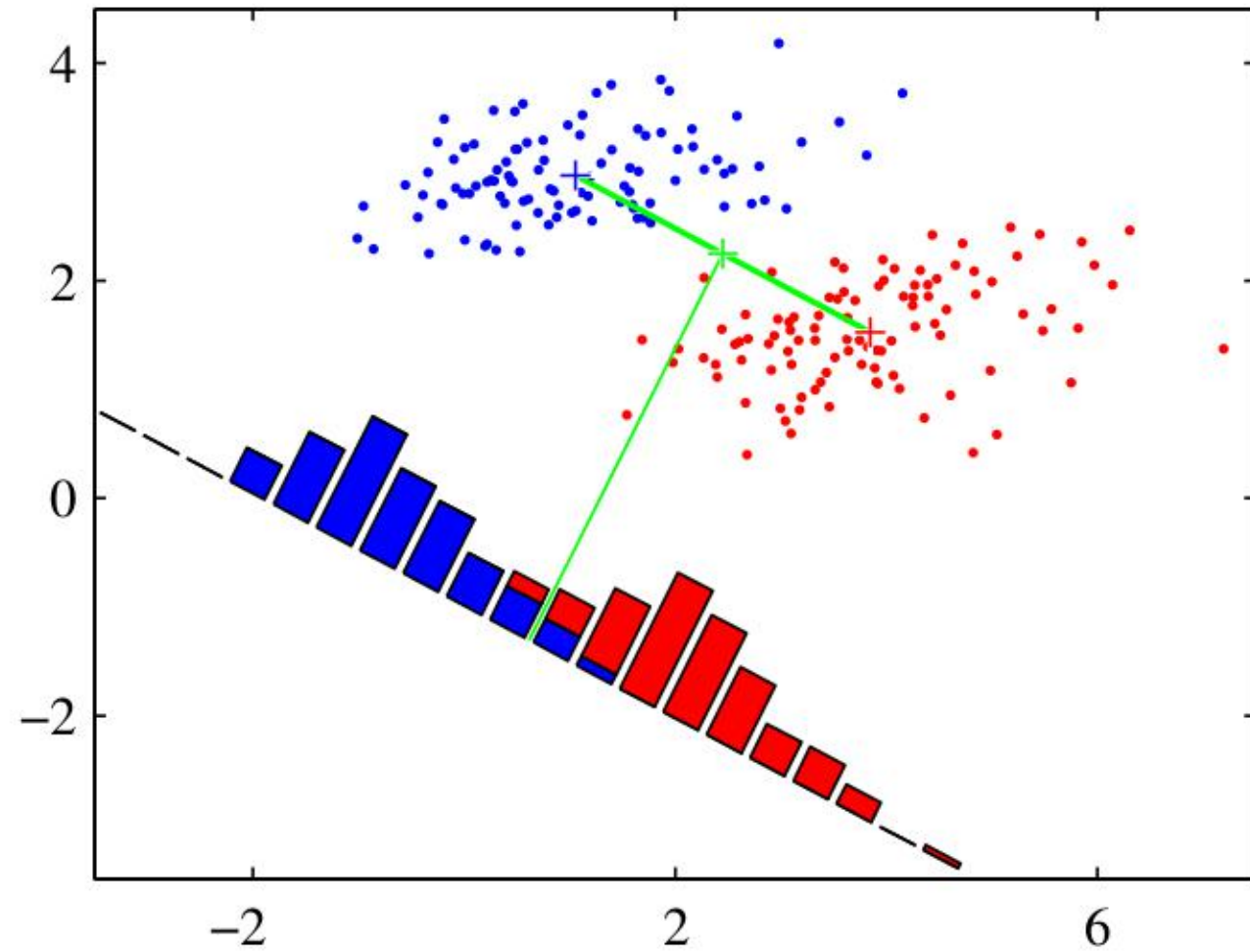


# Digit recognition - etalons $\vec{e}_s = \vec{\mu}_s$



Figures from [5]

## Better etalons – Fischer linear discriminant

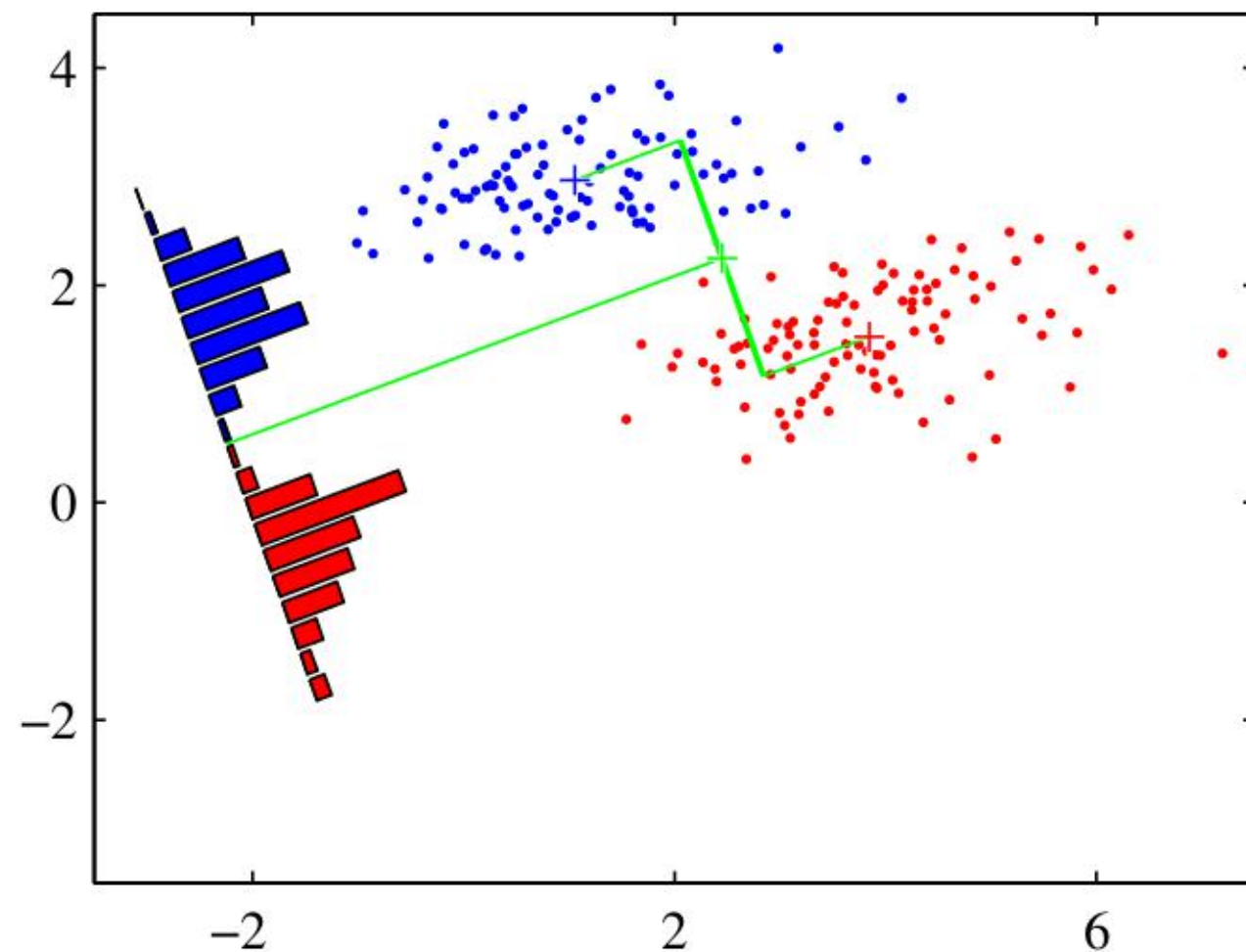
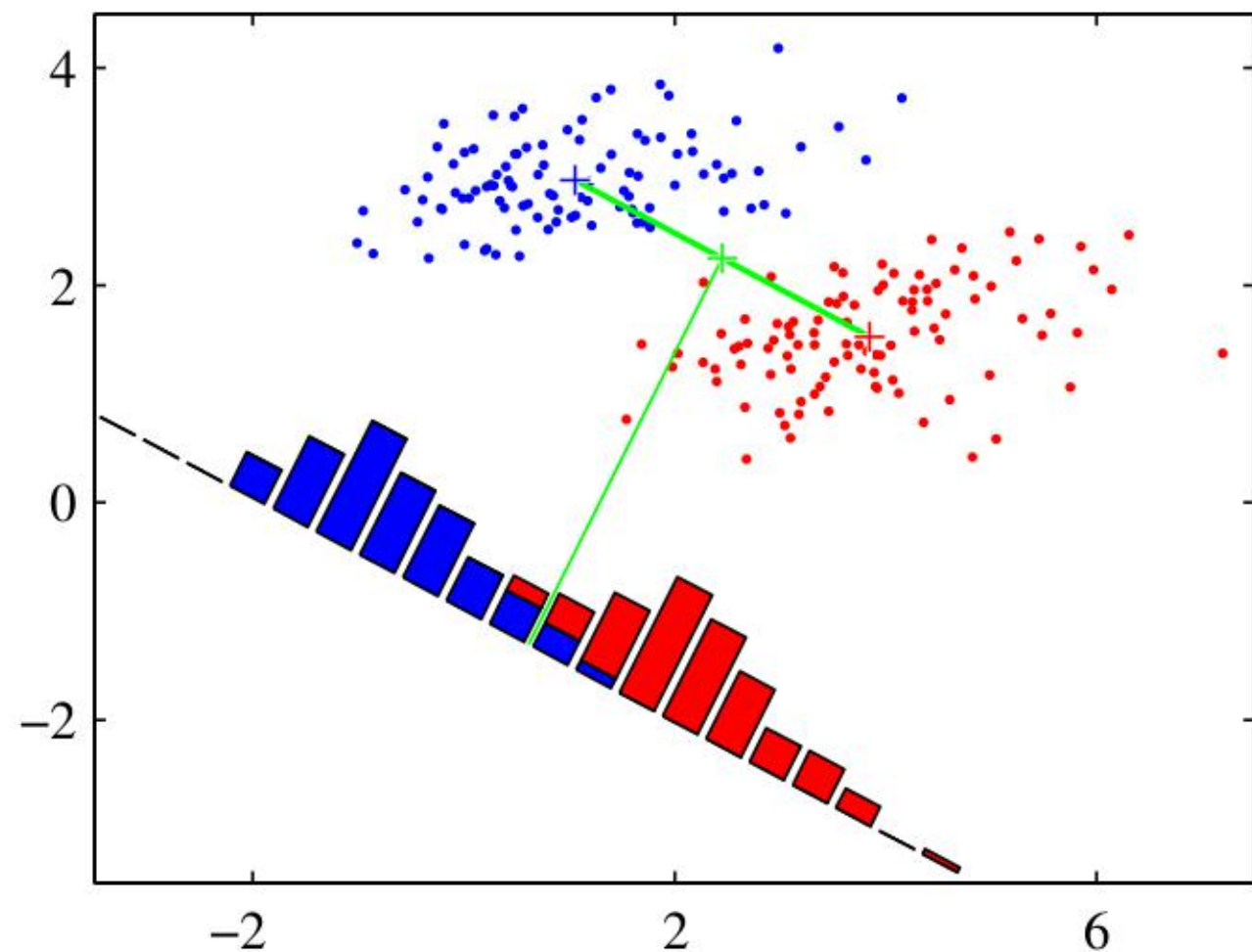


- ▶ Dimensionality reduction
- ▶ Maximize distance between means, ...
- ▶ ... and minimize within class variance. (minimize overlap)

Figures from [1]



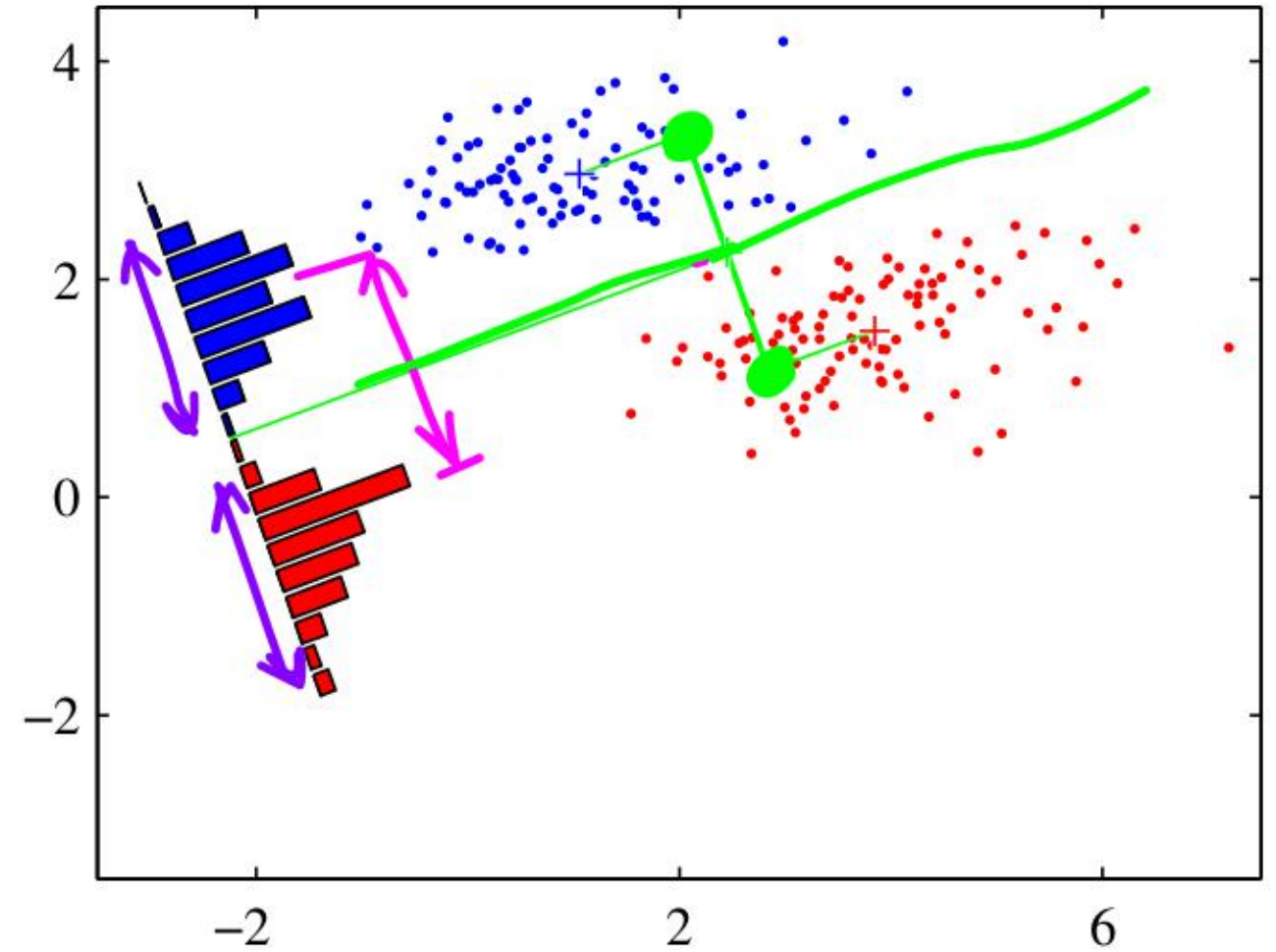
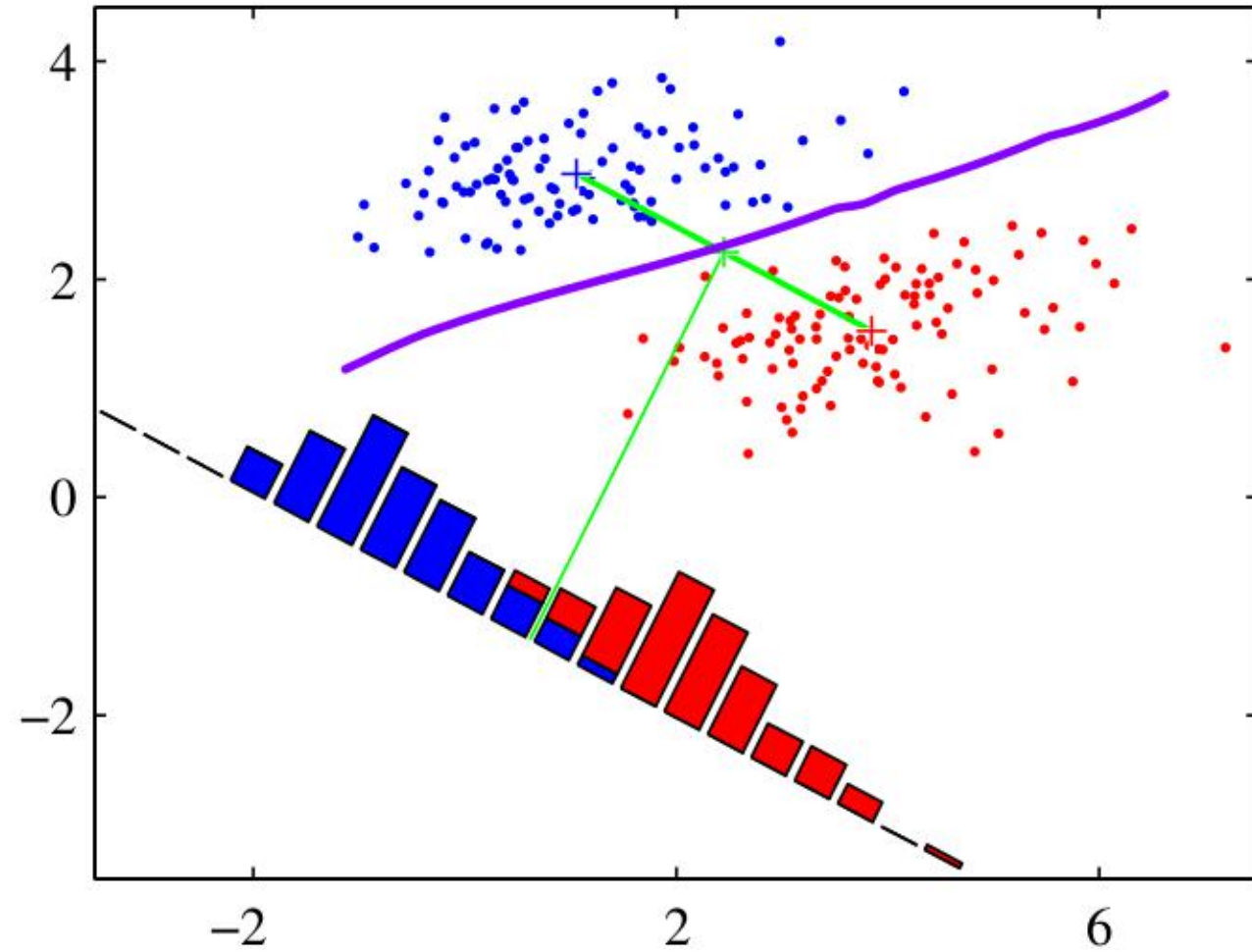
## Better etalons – Fischer linear discriminant



- ▶ Dimensionality reduction
- ▶ Maximize distance between means, ...
- ▶ ... and minimize within class variance. (minimize overlap)

Figures from [1]

# Better etalons – Fischer linear discriminant

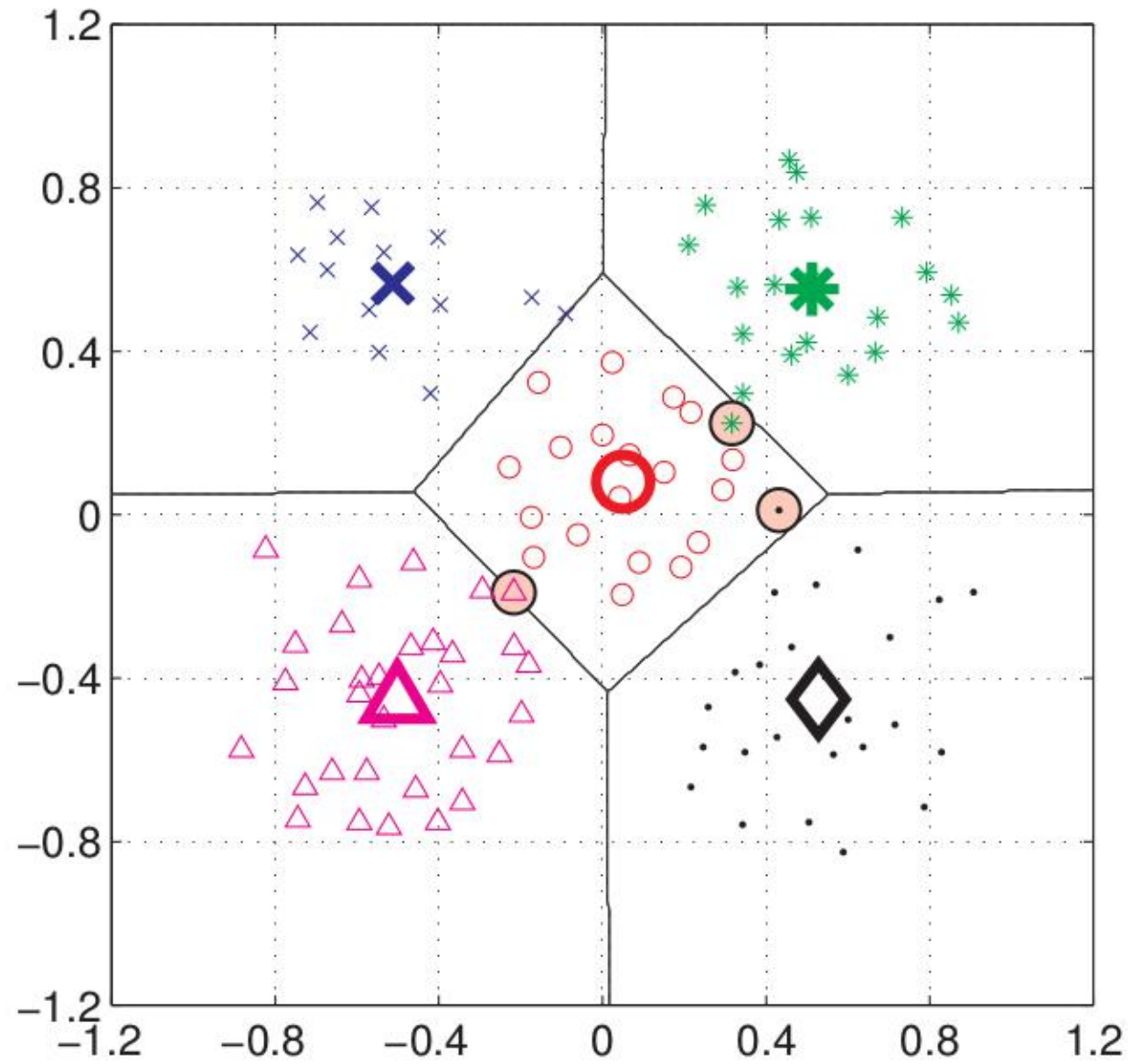


- ▶ Dimensionality reduction
- ▶ Maximize distance between means, ...
- ▶ ... and minimize within class variance. (minimize overlap)

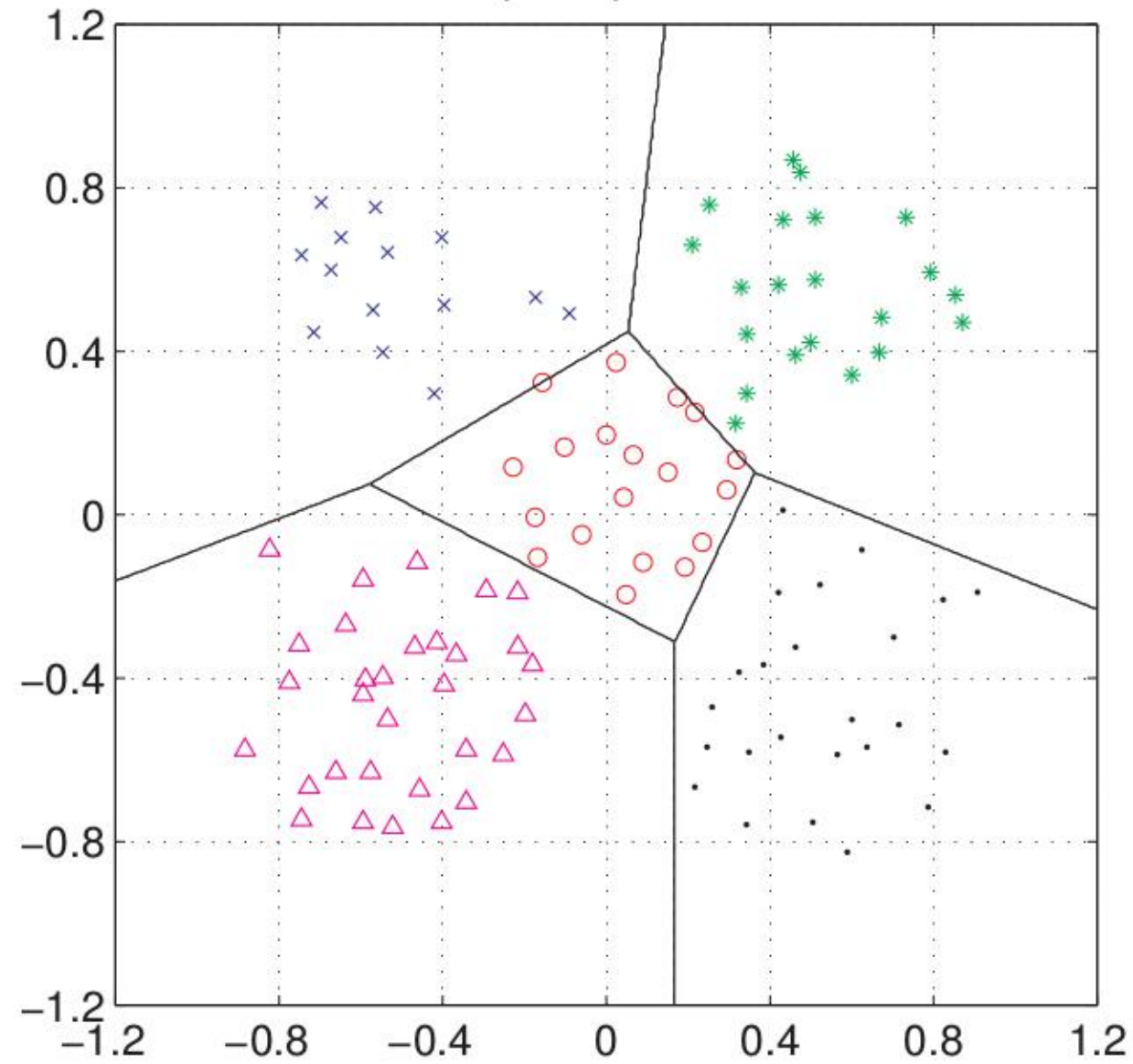
Figures from [1]

# Better etalons?

minimum distance from etalons



perceptron



Figures from [5]



## Etalon classifier – Linear classifier

$$\begin{aligned} s^* &= \arg \min_{s \in S} \|\vec{x} - \vec{e}_s\|^2 = \arg \min_{s \in S} (\vec{x}^\top \vec{x} - 2 \vec{e}_s^\top \vec{x} + \vec{e}_s^\top \vec{e}_s) = \\ &= \arg \min_{s \in S} \left( \vec{x}^\top \vec{x} - 2 \left( \vec{e}_s^\top \vec{x} - \frac{1}{2} (\vec{e}_s^\top \vec{e}_s) \right) \right) = \\ &= \arg \min_{s \in S} (\vec{x}^\top \vec{x} - 2 (\vec{e}_s^\top \vec{x} + b_s)) = \\ &= \boxed{\arg \max_{s \in S} (\vec{e}_s^\top \vec{x} + b_s)} = \arg \max_{s \in S} g_s(\vec{x}). \end{aligned} \quad b_s = -\frac{1}{2} \vec{e}_s^\top \vec{e}_s$$

Linear function (plus offset)

$$g_s(\mathbf{x}) = \mathbf{w}_s^\top \mathbf{x} + w_{s0}$$



## Etalon classifier – Linear classifier

$$\begin{aligned} s^* &= \arg \min_{s \in S} \|\vec{x} - \vec{e}_s\|^2 = \arg \min_{s \in S} (\vec{x}^\top \vec{x} - 2 \vec{e}_s^\top \vec{x} + \vec{e}_s^\top \vec{e}_s) = \\ &= \arg \min_{s \in S} \left( \vec{x}^\top \vec{x} - 2 \left( \vec{e}_s^\top \vec{x} - \frac{1}{2} (\vec{e}_s^\top \vec{e}_s) \right) \right) = \\ &= \arg \min_{s \in S} (\vec{x}^\top \vec{x} - 2 (\vec{e}_s^\top \vec{x} + b_s)) = \\ &= \boxed{\arg \max_{s \in S} (\vec{e}_s^\top \vec{x} + b_s)} = \arg \max_{s \in S} g_s(\vec{x}). \end{aligned}$$

$$b_s = -\frac{1}{2} \vec{e}_s^\top \vec{e}_s$$

Linear function (plus offset)

$$g_s(\mathbf{x}) = \mathbf{w}_s^\top \mathbf{x} + w_{s0}$$

# Learning and decision

**Learning** stage - learning models/function/parameters from data.

**Decision** stage - decide about a query  $\vec{x}$ .

What to learn?

- ▶ **Generative model** : Learn  $P(\vec{x}, s)$ . Decide by computing  $P(s|\vec{x})$ .
- ▶ **Discriminative model** : Learn  $P(s|\vec{x})$
- ▶ **Discriminant function** : Learn  $g(\vec{x})$  which maps  $\vec{x}$  directly into class labels.

# Learning and decision

**Learning** stage - learning models/function/parameters from data.

**Decision** stage - decide about a query  $\vec{x}$ .

What to learn?

- ▶ **Generative model** : Learn  $P(\vec{x}, s)$ . Decide by computing  $P(s|\vec{x})$ .
- ▶ **Discriminative model** : Learn  $P(s|\vec{x})$
- ▶ **Discriminant function** : Learn  $g(\vec{x})$  which maps  $\vec{x}$  directly into class labels.

  $\square \rightarrow \text{Convolution Deep Nets} \rightarrow s^*$

## (1) Linear discriminant function - two class case

$$g(\mathbf{x}) = \mathbf{w}^\top \mathbf{x} + w_0$$

Decide  $s_1$  if  $g(\mathbf{x}) > 0$  and  $s_2$  if  $g(\mathbf{x}) < 0$



# (1) Linear discriminant function - two class case



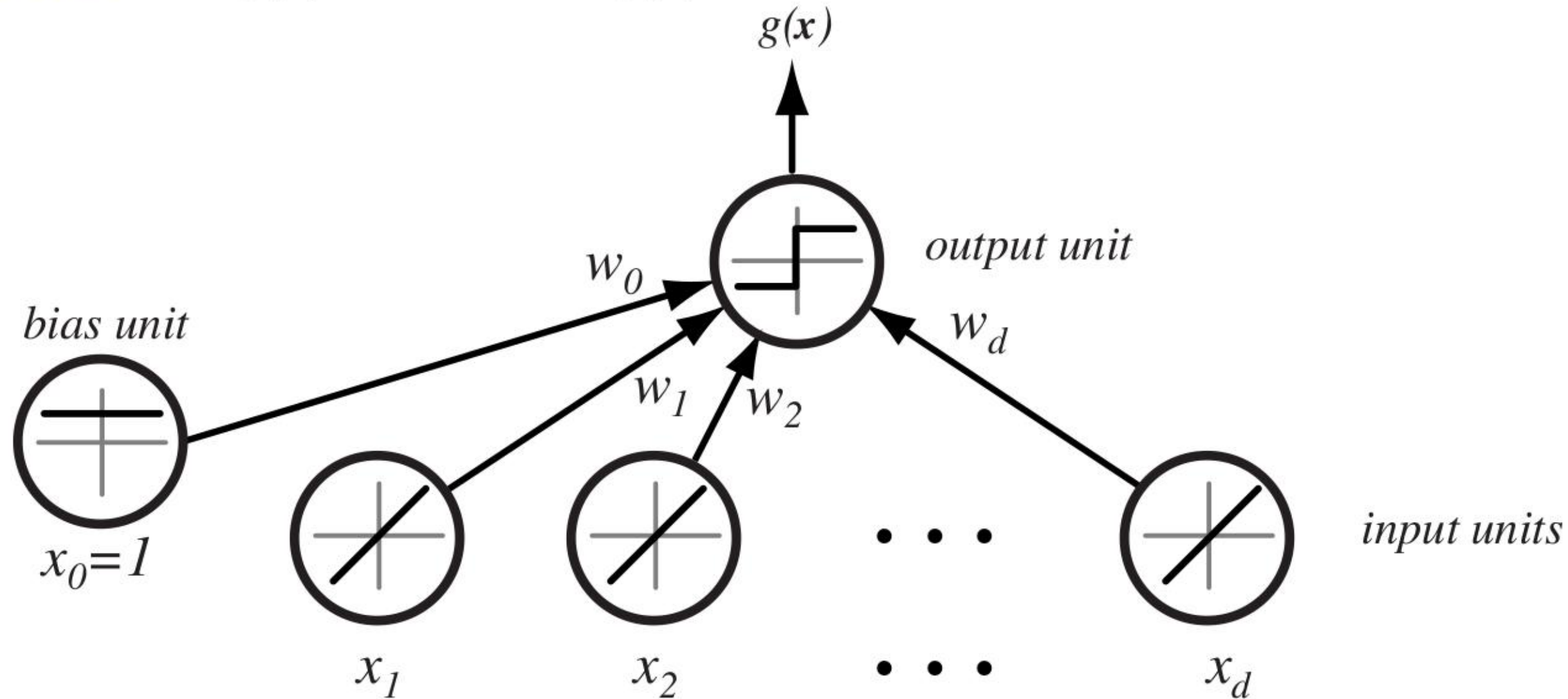
$$g(\mathbf{x}) = \mathbf{w}^T \mathbf{x} + w_0$$

Decide  $s_1$  if  $g(\mathbf{x}) > 0$  and  $s_2$  if  $g(\mathbf{x}) < 0$

# (1) Linear discriminant function - two class case

$$g(\mathbf{x}) = \mathbf{w}^T \mathbf{x} + w_0$$

Decide  $s_1$  if  $g(\mathbf{x}) > 0$  and  $s_2$  if  $g(\mathbf{x}) < 0$



# Separating hyperplane

$$\mathbf{w}^\top \mathbf{x}_1 + w_0 = \mathbf{w}^\top \mathbf{x}_2 + w_0$$

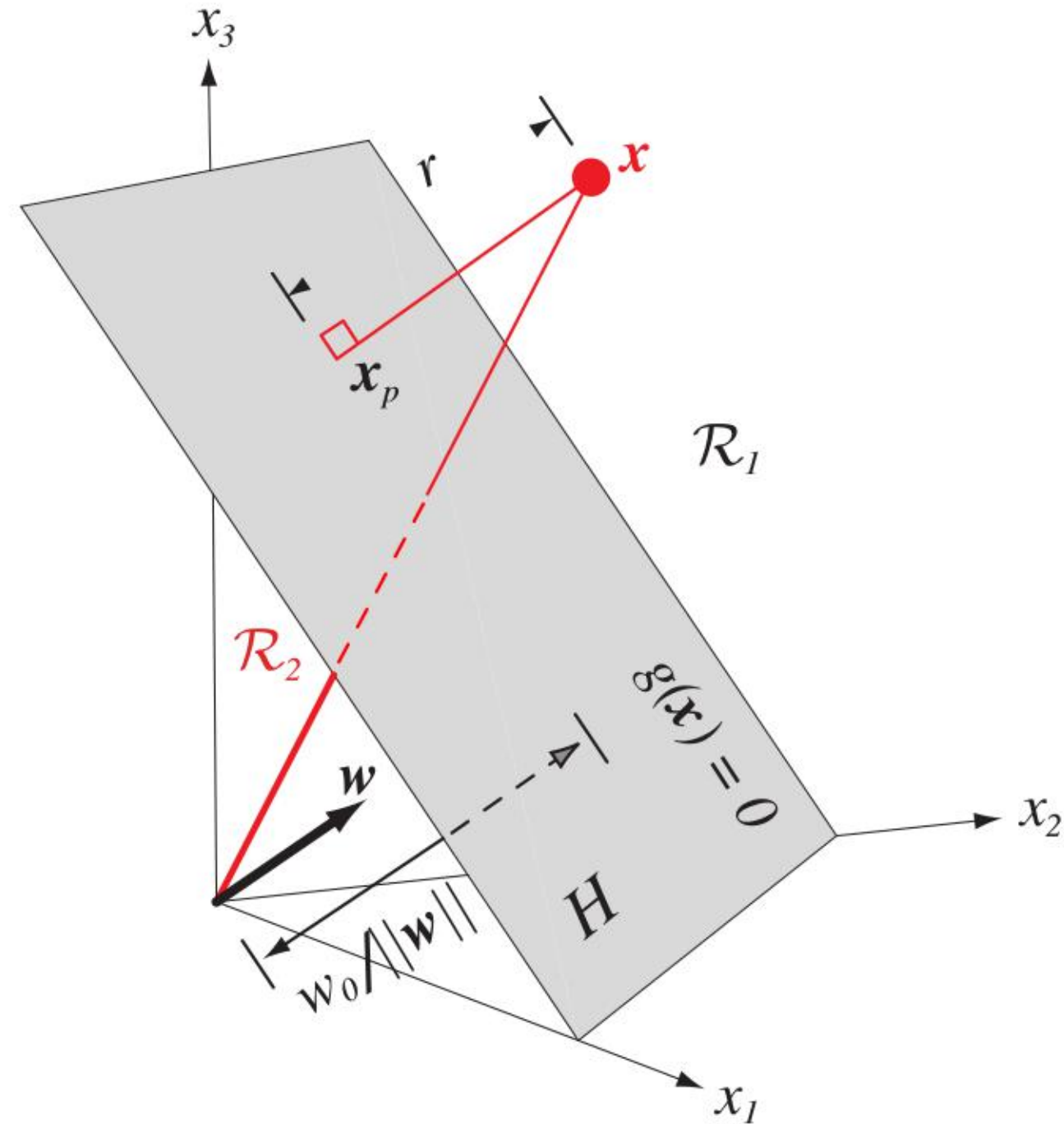
$$\mathbf{w}^\top (\mathbf{x}_1 - \mathbf{x}_2) = 0$$

$g(\mathbf{x})$  gives an algebraic measure of the distance from  $\mathbf{x}$  to the hyperplane.

$$\mathbf{x} = \mathbf{x}_p + r \frac{\mathbf{w}}{\|\mathbf{w}\|}$$

as  $g(\mathbf{x}_p) = 0$ ,  
and  $g(\mathbf{x}) = \mathbf{w}^\top \mathbf{x} + w_0$ , then:

$$g(\mathbf{x}) = r \|\mathbf{w}\|$$



# Separating hyperplane

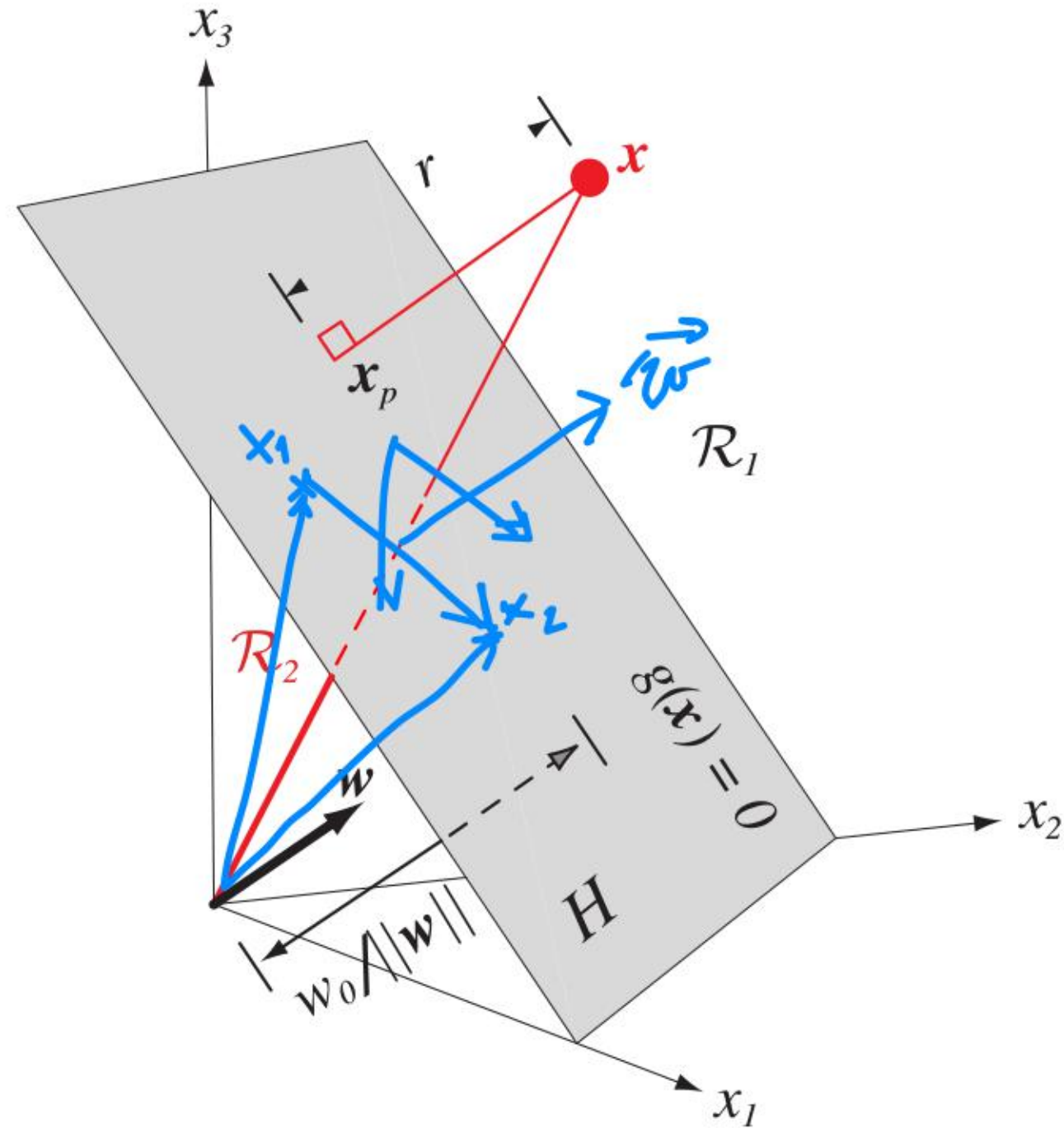
$$\mathbf{w}^\top \mathbf{x}_1 + w_0 = \mathbf{w}^\top \mathbf{x}_2 + w_0$$
$$\mathbf{w}^\top (\mathbf{x}_1 - \mathbf{x}_2) = 0$$

$g(\mathbf{x})$  gives an algebraic measure of the distance from  $\mathbf{x}$  to the hyperplane.

$$\mathbf{x} = \mathbf{x}_p + r \frac{\mathbf{w}}{\|\mathbf{w}\|}$$

as  $g(\mathbf{x}_p) = 0$ ,  
and  $g(\mathbf{x}) = \mathbf{w}^\top \mathbf{x} + w_0$ , then:

$$g(\mathbf{x}) = r \|\mathbf{w}\|$$





# Separating hyperplane

$$\mathbf{w}^\top \mathbf{x}_1 + w_0 = \mathbf{w}^\top \mathbf{x}_2 + w_0$$

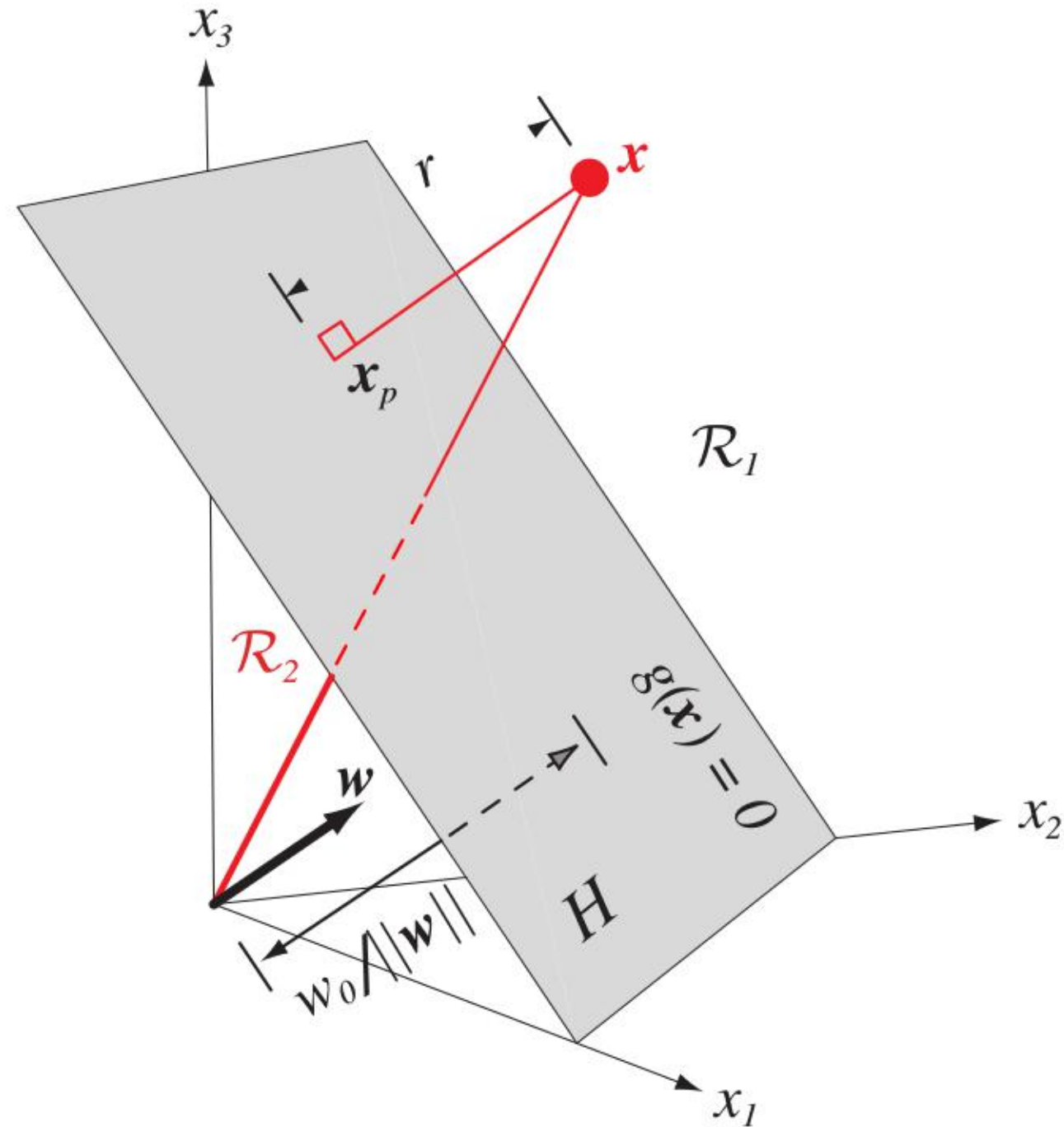
$$\mathbf{w}^\top (\mathbf{x}_1 - \mathbf{x}_2) = 0$$

$g(\mathbf{x})$  gives an algebraic measure of the distance from  $\mathbf{x}$  to the hyperplane.

$$\mathbf{x} = \mathbf{x}_p + r \frac{\mathbf{w}}{\|\mathbf{w}\|}$$

as  $g(\mathbf{x}_p) = 0$ ,  
and  $g(\mathbf{x}) = \mathbf{w}^\top \mathbf{x} + w_0$ , then:

$$g(\mathbf{x}) = r \|\mathbf{w}\|$$



# Separating hyperplane

$$\mathbf{w}^\top \mathbf{x}_1 + w_0 = \mathbf{w}^\top \mathbf{x}_2 + w_0$$

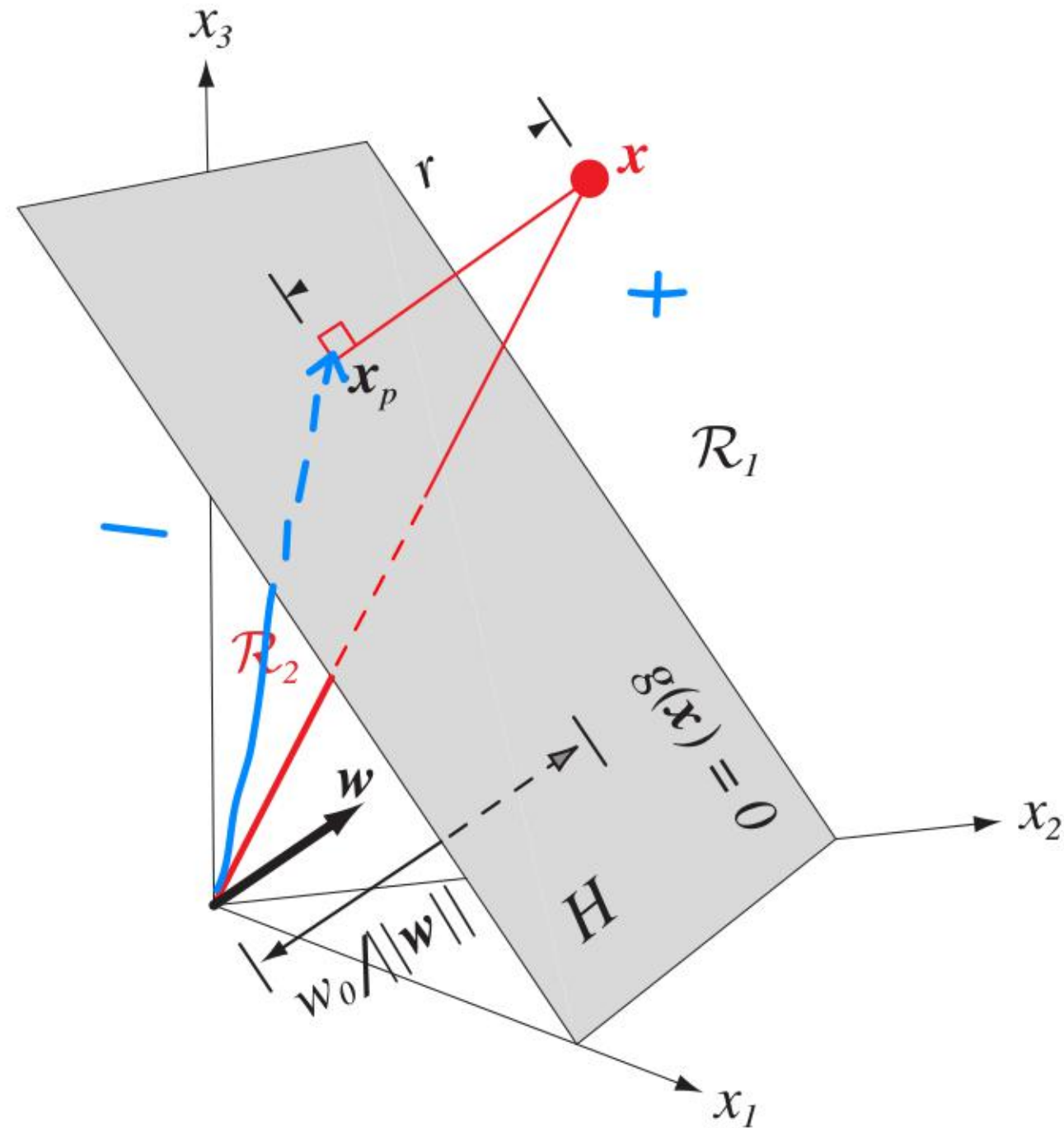
$$\mathbf{w}^\top (\mathbf{x}_1 - \mathbf{x}_2) = 0$$

$g(\mathbf{x})$  gives an algebraic measure of the distance from  $\mathbf{x}$  to the hyperplane.

$$\mathbf{x} = \mathbf{x}_p + r \frac{\mathbf{w}}{\|\mathbf{w}\|}$$

as  $g(\mathbf{x}_p) = 0$ ,  
and  $g(\mathbf{x}) = \mathbf{w}^\top \mathbf{x} + w_0$ , then:

$$g(\mathbf{x}) = r \|\mathbf{w}\|$$



## Separating hyperplane from $g_1$ and $g_2$

$$g_1(\vec{x}) = \vec{\mu}_1^\top \vec{x} - \frac{1}{2} \vec{\mu}_1^\top \vec{\mu}_1$$

$$g_2(\vec{x}) = \vec{\mu}_2^\top \vec{x} - \frac{1}{2} \vec{\mu}_2^\top \vec{\mu}_2$$

Separating hyperplane:

$$g_1(\vec{x}) = g_2(\vec{x})$$

$$(\vec{\mu}_1 - \vec{\mu}_2)^\top \vec{x} = \frac{1}{2} (\vec{\mu}_1^\top \vec{\mu}_1 - \vec{\mu}_2^\top \vec{\mu}_2)$$

# Separating hyperplane from $g_1$ and $g_2$

$$g_1(\vec{x}) = \vec{\mu}_1^\top \vec{x} - \frac{1}{2} \vec{\mu}_1^\top \vec{\mu}_1$$

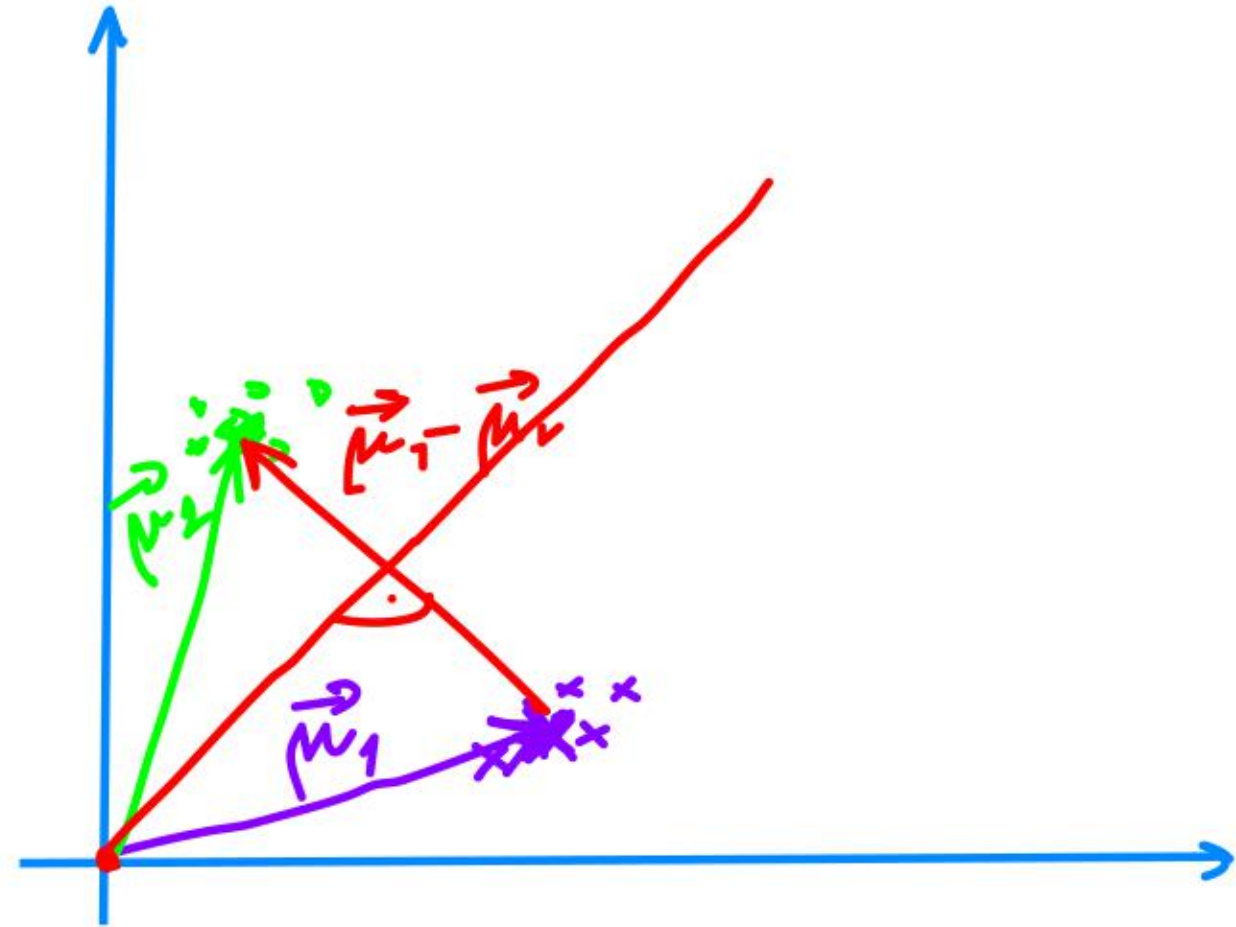
$$g_2(\vec{x}) = \vec{\mu}_2^\top \vec{x} - \frac{1}{2} \vec{\mu}_2^\top \vec{\mu}_2$$

Separating hyperplane:

$$g_1(\vec{x}) = g_2(\vec{x})$$

~~$$(\vec{\mu}_1 - \vec{\mu}_2)^\top \vec{x} = \frac{1}{2} (\vec{\mu}_1^\top \vec{\mu}_1 - \vec{\mu}_2^\top \vec{\mu}_2)$$~~

$\neq$





## Two classes set-up

$|S| = 2$ , i.e. two states (typically also classes)

$$g(\mathbf{x}) = \begin{cases} s = 1, & \text{if } \mathbf{w}^\top \mathbf{x} + w_0 > 0, \\ s = -1, & \text{if } \mathbf{w}^\top \mathbf{x} + w_0 < 0. \end{cases}$$

$$\mathbf{x}'_j = s_j \begin{bmatrix} 1 \\ \mathbf{x}_j \end{bmatrix}, \quad \mathbf{w}' = \begin{bmatrix} w_0 \\ \mathbf{w} \end{bmatrix}$$

for all  $\mathbf{x}'$

$$\mathbf{w}'^\top \mathbf{x}' > 0$$

drop the dashes to avoid notation clutter.

## Two classes set-up

$|S| = 2$ , i.e. two states (typically also classes)

$$g(\mathbf{x}) = \begin{cases} s = 1, & \text{if } \mathbf{w}^\top \mathbf{x} + w_0 > 0, \\ s = -1, & \text{if } \mathbf{w}^\top \mathbf{x} + w_0 < 0. \end{cases}$$

$$\mathbf{x}'_j = s_j \begin{bmatrix} 1 \\ \mathbf{x}_j \end{bmatrix}, \quad \mathbf{w}' = \begin{bmatrix} w_0 \\ \mathbf{w} \end{bmatrix}$$

for all  $\mathbf{x}'$

$$\mathbf{w}'^\top \mathbf{x}' > 0$$

drop the dashes to avoid notation clutter.

## Two classes set-up

$|S| = 2$ , i.e. two states (typically also classes)

$$g(\mathbf{x}) = \begin{cases} s = 1, & \text{if } \mathbf{w}^\top \mathbf{x} + w_0 > 0, \\ s = -1, & \text{if } \mathbf{w}^\top \mathbf{x} + w_0 < 0. \end{cases}$$

$$\mathbf{x}'_j = s_j \begin{bmatrix} 1 \\ \mathbf{x}_j \end{bmatrix}, \quad \mathbf{w}' = \begin{bmatrix} w_0 \\ \mathbf{w} \end{bmatrix}$$

for all  $\mathbf{x}'$

$$\mathbf{w}'^\top \mathbf{x}' > 0$$

drop the dashes to avoid notation clutter.

## Two classes set-up

$|S| = 2$ , i.e. two states (typically also classes)

$$g(\mathbf{x}) = \begin{cases} s = 1, & \text{if } \mathbf{w}^\top \mathbf{x} + w_0 > 0, \\ s = -1, & \text{if } \mathbf{w}^\top \mathbf{x} + w_0 < 0. \end{cases}$$

$$\mathbf{w}'^\top \mathbf{x}'$$

$$\mathbf{x}'_j = s_j \begin{bmatrix} 1 \\ \mathbf{x}_j \end{bmatrix}, \quad \mathbf{w}' = \begin{bmatrix} w_0 \\ \mathbf{w} \end{bmatrix}$$

for all  $\mathbf{x}'$

$$\mathbf{w}'^\top \mathbf{x}' > 0$$

drop the dashes to avoid notation clutter.



## Two classes set-up

$|S| = 2$ , i.e. two states (typically also classes)

$$g(\mathbf{x}) = \begin{cases} s = 1, & \text{if } \mathbf{w}^\top \mathbf{x} + w_0 > 0, \\ s = -1, & \text{if } \mathbf{w}^\top \mathbf{x} + w_0 < 0. \end{cases}$$

$$\mathbf{x}'_j = s_j \begin{bmatrix} 1 \\ \mathbf{x}_j \end{bmatrix}, \quad \mathbf{w}' = \begin{bmatrix} w_0 \\ \mathbf{w} \end{bmatrix}$$

for all  $\mathbf{x}'$

$$\mathbf{w}'^\top \mathbf{x}' > 0$$

drop the dashes to avoid notation clutter.

## Two classes set-up

$|S| = 2$ , i.e. two states (typically also classes)

$$g(\mathbf{x}) = \begin{cases} s = 1, & \text{if } \mathbf{w}^\top \mathbf{x} + w_0 > 0, \\ s = -1, & \text{if } \mathbf{w}^\top \mathbf{x} + w_0 < 0. \end{cases}$$

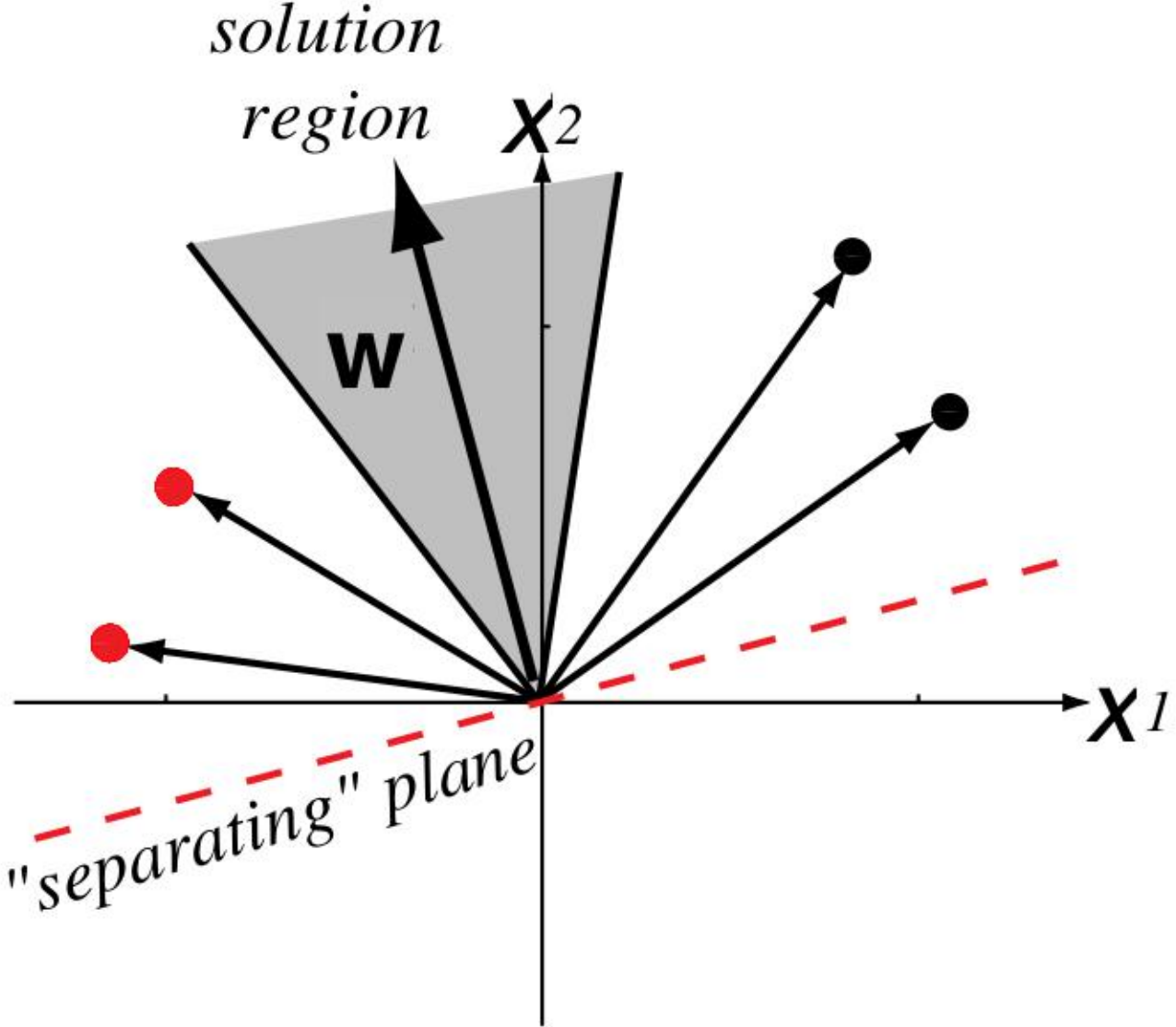
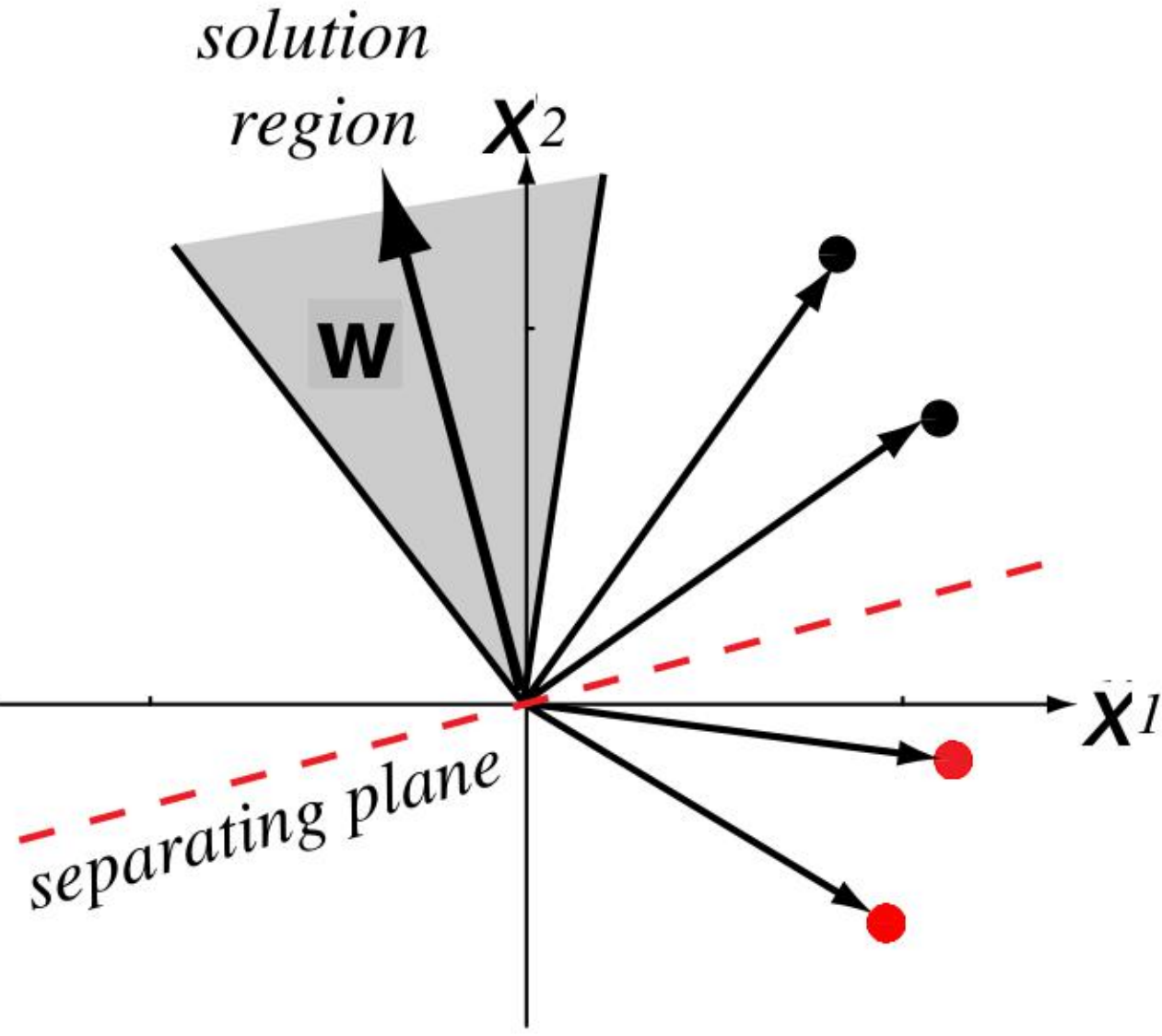
$$\mathbf{x}'_j = s_j \begin{bmatrix} 1 \\ \mathbf{x}_j \end{bmatrix}, \quad \mathbf{w}' = \begin{bmatrix} w_0 \\ \mathbf{w} \end{bmatrix}$$

for all  $\mathbf{x}'$

$$\mathbf{w}'^\top \mathbf{x}' > 0$$

drop the dashes to avoid notation clutter.

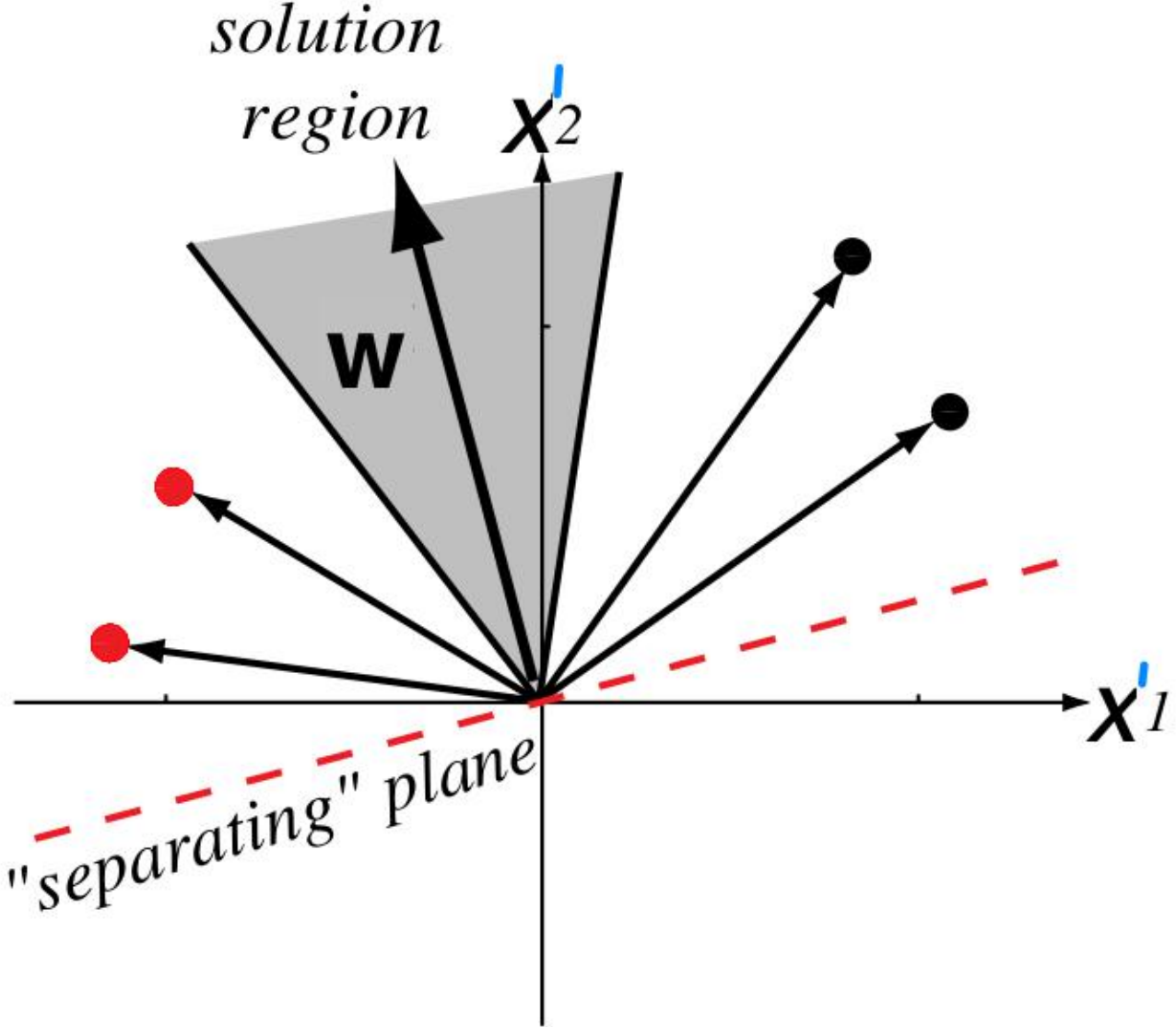
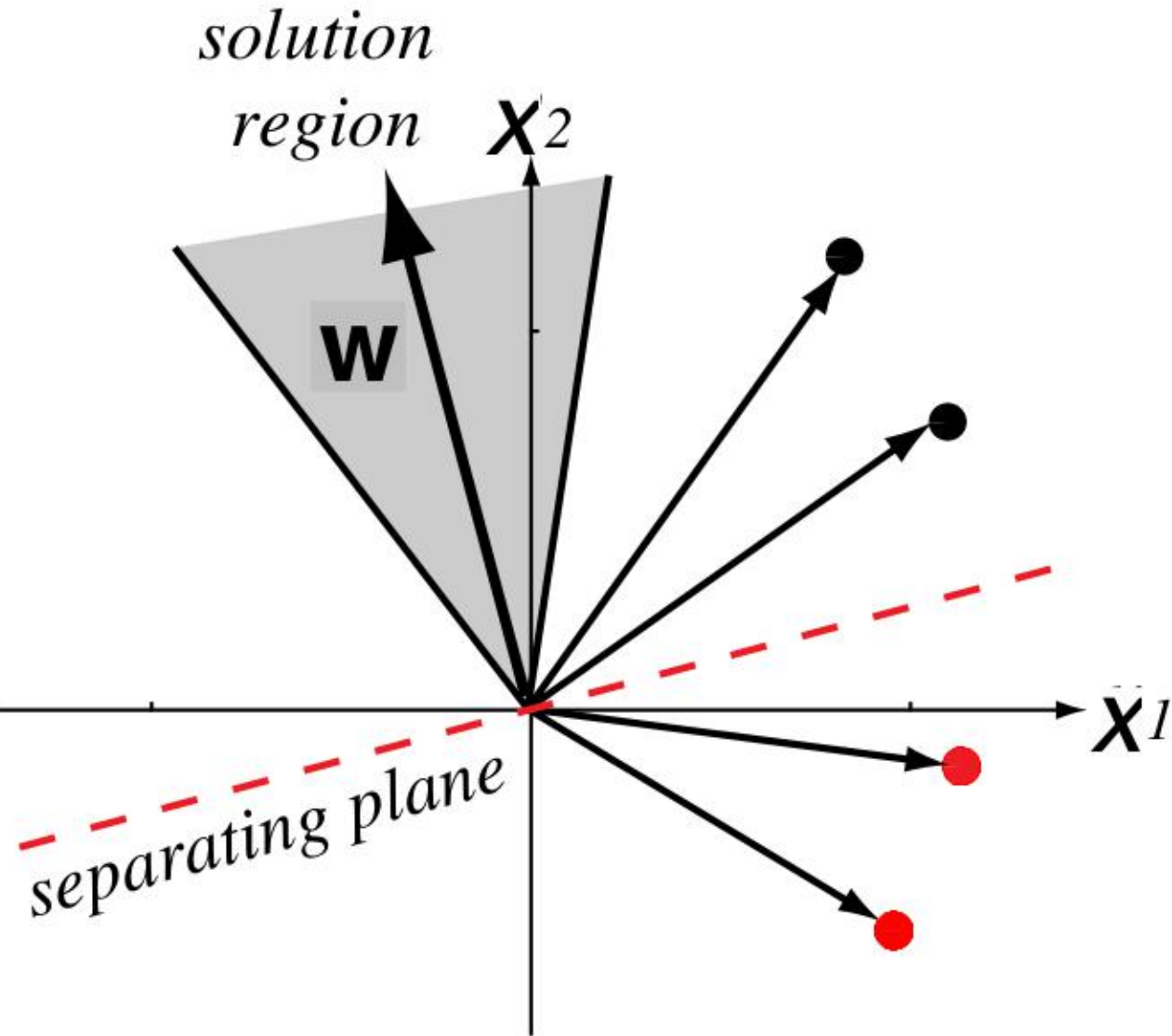
# Solution (graphically)



Four training samples. Left: original, Right: sign corrected

Figure from [2] (notation changed)

# Solution (graphically)



Four training samples. Left: original, Right: sign corrected

Figure from [2] (notation changed)



# Learning $\mathbf{w}$ , gradient descent

A criterion to be minimized  $J(\mathbf{w})$ ; assume to be known

Initialize  $\mathbf{w}$ , threshold  $\theta$ , learning rate  $\alpha$

$k \leftarrow 0$

**repeat**

$k \leftarrow k + 1$

$\mathbf{w} \leftarrow \mathbf{w} - \alpha(k)\nabla J(\mathbf{w})$

**until**  $|\alpha(k)\nabla J(\mathbf{w})| < \theta$

return  $\mathbf{w}$

# Learning $\mathbf{w}$ , gradient descent

?

A criterion to be minimized  $J(\mathbf{w})$ ; assume to be known

Initialize  $\mathbf{w}$ , threshold  $\theta$ , learning rate  $\alpha$

$k \leftarrow 0$

**repeat**

$k \leftarrow k + 1$

$\mathbf{w} \leftarrow \mathbf{w} - \alpha(k) \nabla J(\mathbf{w})$

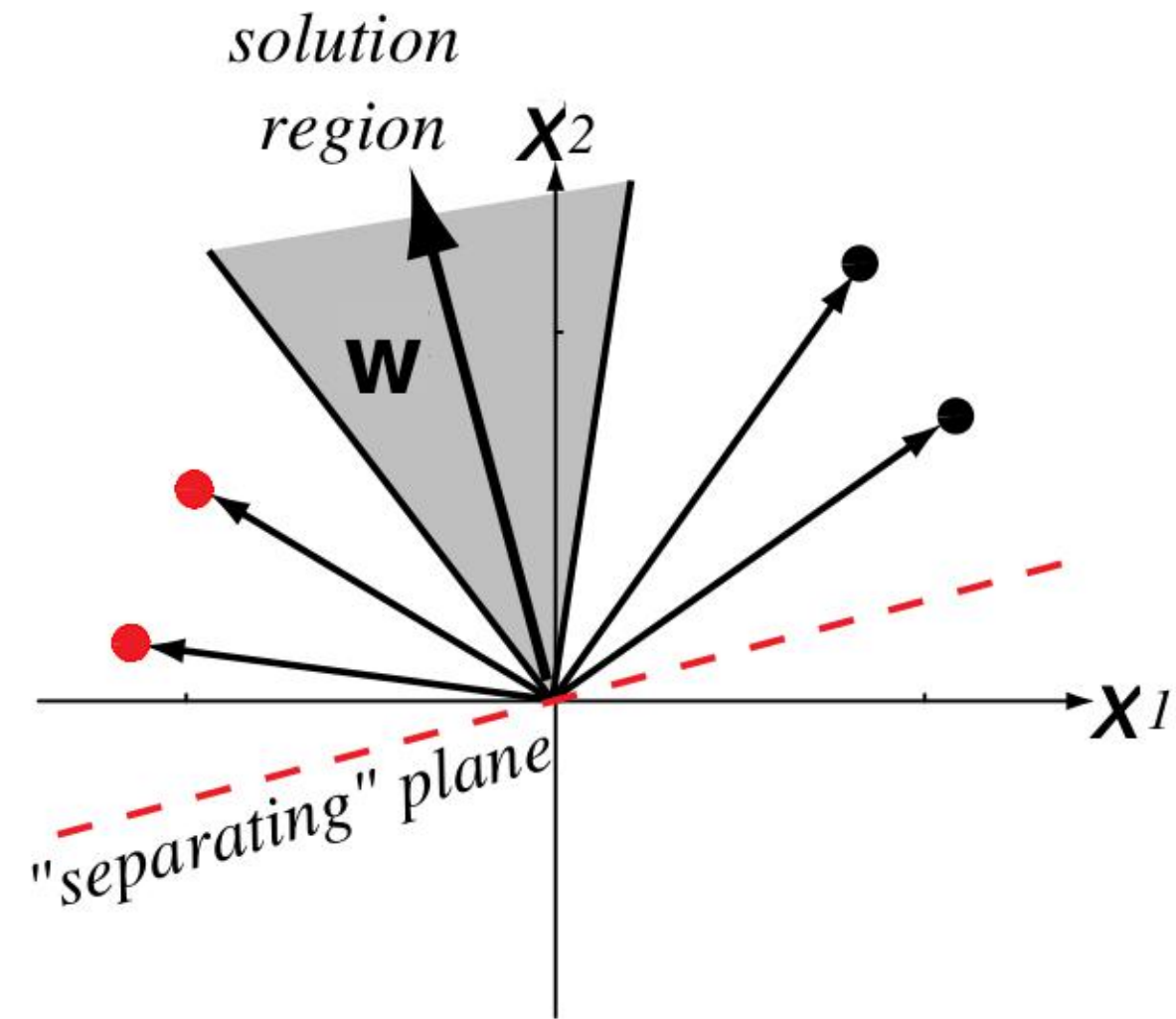
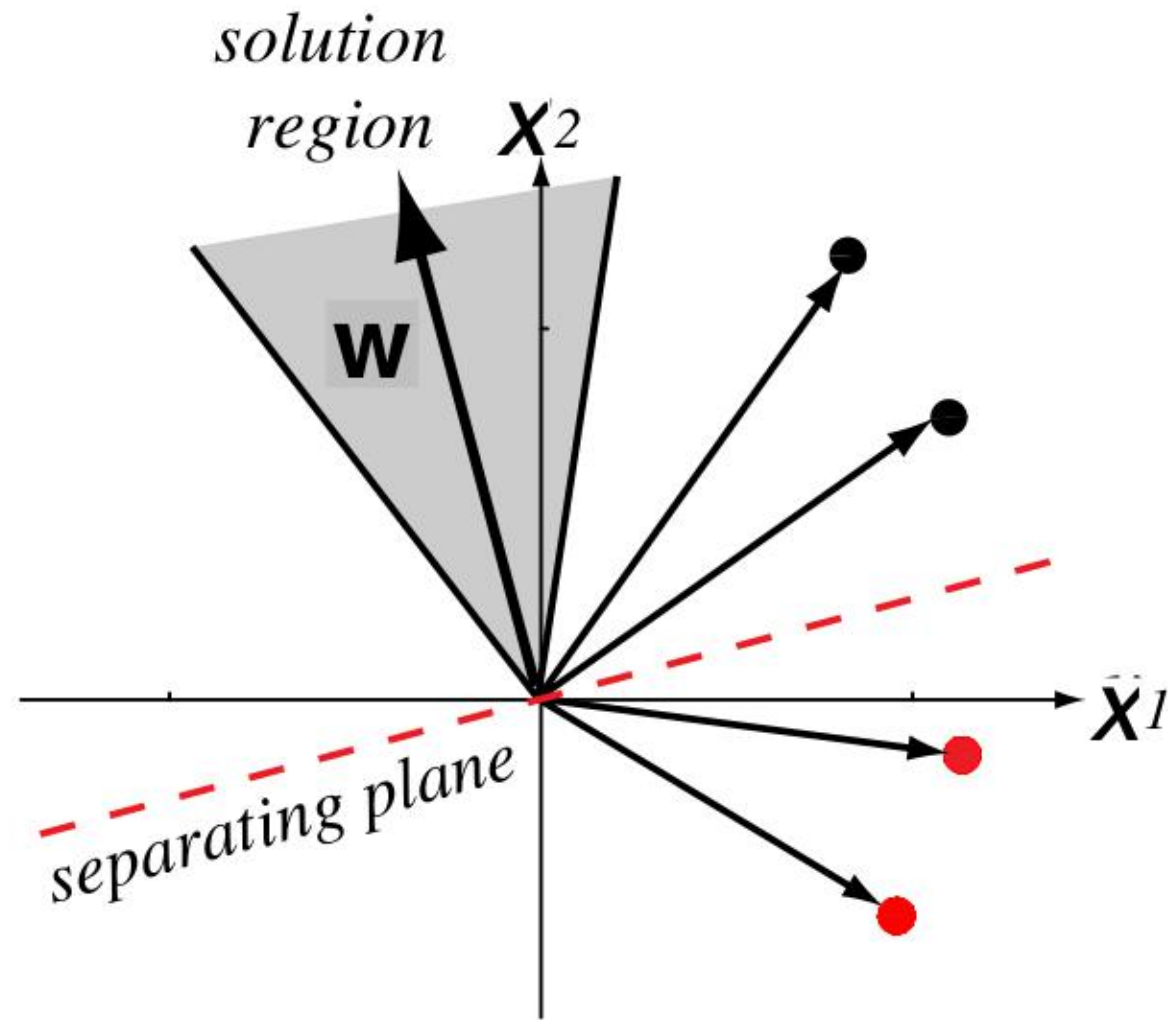
**until**  $|\alpha(k) \nabla J(\mathbf{w})| < \theta$

return  $\mathbf{w}$

# Learning $\mathbf{w}$ - Perceptron criterion

**Goal:** Find a weight vector  $\mathbf{w} \in \mathbb{R}^{D+1}$  (original feature space dimensionality is  $D$ ) such that:

$$\mathbf{w}^\top \mathbf{x}_j > 0 \quad (\forall j \in \{1, 2, \dots, m\})$$



(Perceptron) Criterion to be minimized:

*solution  
region*  $\mathbf{x}_2$

# Learning $\mathbf{w}$ - Perceptron criterion

**Goal:** Find a weight vector  $\mathbf{w} \in \mathbb{R}^{D+1}$  (original feature space dimensionality is  $D$ ) such that:

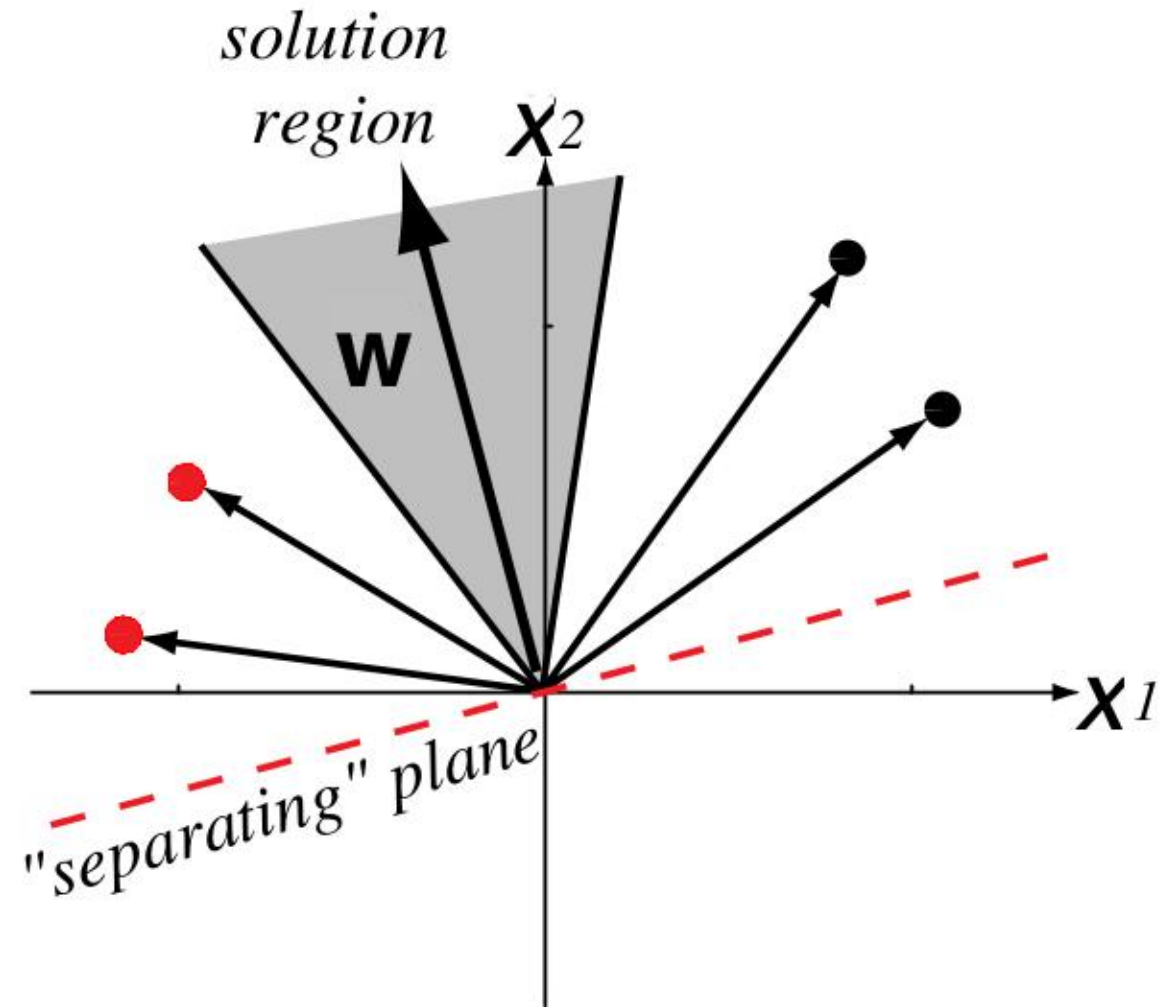
$$\mathbf{w}^\top \mathbf{x}_j > 0 \quad (\forall j \in \{1, 2, \dots, m\})$$

(Perceptron) Criterion to be minimized:

$$J(\mathbf{w}) = \sum_{\mathbf{x} \in \mathcal{X}} -\mathbf{w}^\top \mathbf{x}$$

where  $\mathcal{X}$  is a set of misclassified  $\mathbf{x}$ .

$$\nabla J(\mathbf{w}) = \sum_{\mathbf{x} \in \mathcal{X}} -\mathbf{x}$$





# Learning $\mathbf{w}$ - Perceptron criterion

**Goal:** Find a weight vector  $\mathbf{w} \in \mathbb{R}^{D+1}$  (original feature space dimensionality is  $D$ ) such that:

$$\mathbf{w}^\top \mathbf{x}_j > 0 \quad (\forall j \in \{1, 2, \dots, m\})$$

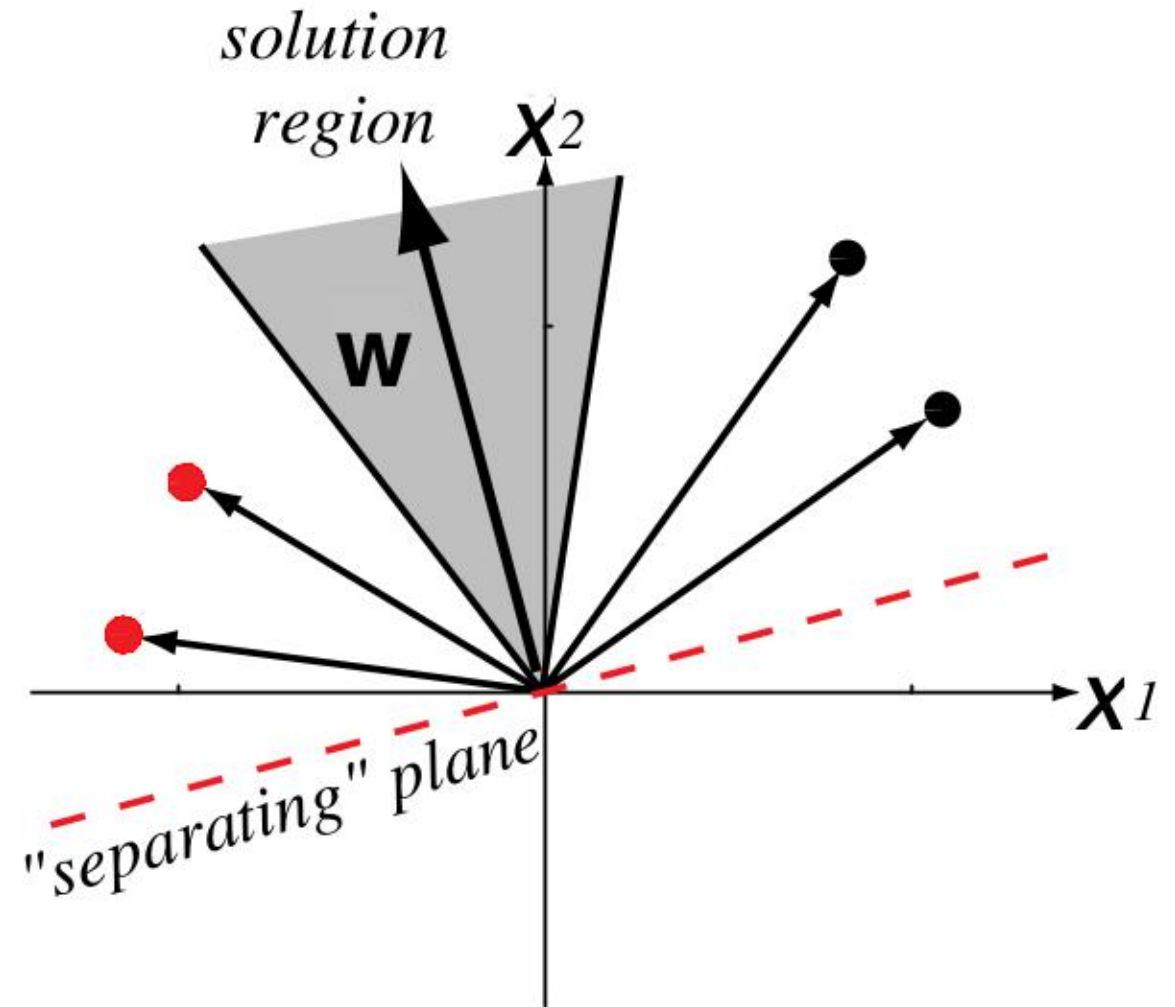
(Perceptron) Criterion to be minimized:

$$J(\mathbf{w}) = \sum_{\mathbf{x} \in \mathcal{X}} -\mathbf{w}^\top \mathbf{x}$$

where  $\mathcal{X}$  is a set of misclassified  $\mathbf{x}$ .



$$\nabla J(\mathbf{w}) = \sum_{\mathbf{x} \in \mathcal{X}} -\mathbf{x}$$



# (Batch) Perceptron algorithm

Initialize  $\mathbf{w}$ , threshold  $\theta$ , learning rate  $\alpha$

$k \leftarrow 0$

**repeat**

$k \leftarrow k + 1$

$\mathbf{w} \leftarrow \mathbf{w} + \alpha(k) \sum_{\mathbf{x} \in \mathcal{X}(k)} \mathbf{x}$

**until**  $|\alpha(k) \sum_{\mathbf{x} \in \mathcal{X}(k)} \mathbf{x}| < \theta$

return  $\mathbf{w}$

## (Batch) Perceptron algorithm

Initialize  $\mathbf{w}$ , threshold  $\theta$ , learning rate  $\alpha$

$k \leftarrow 0$

**repeat**

$k \leftarrow k + 1$

$\mathbf{w} \leftarrow \mathbf{w} + \alpha(k) \sum_{\mathbf{x} \in \mathcal{X}(k)} \mathbf{x}$

**until**  $|\alpha(k) \sum_{\mathbf{x} \in \mathcal{X}(k)} \mathbf{x}| < \theta$

return  $\mathbf{w}$

# Fixed-increment single-sample Perceptron

$n$  patterns/samples, we are looping over all patterns repeatedly

Initialize  $\mathbf{w}$

$k \leftarrow 0$

**repeat**

$k \leftarrow (k + 1) \bmod n$

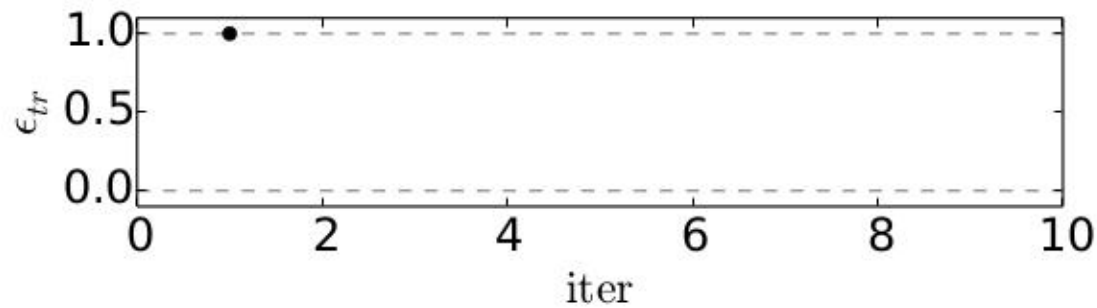
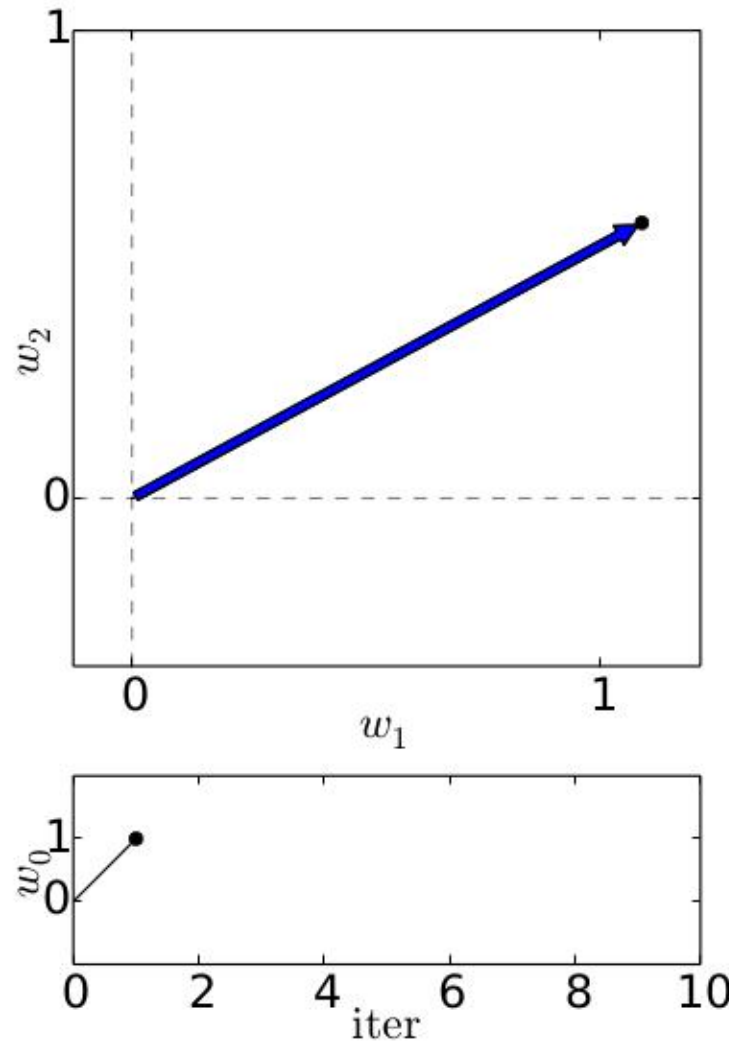
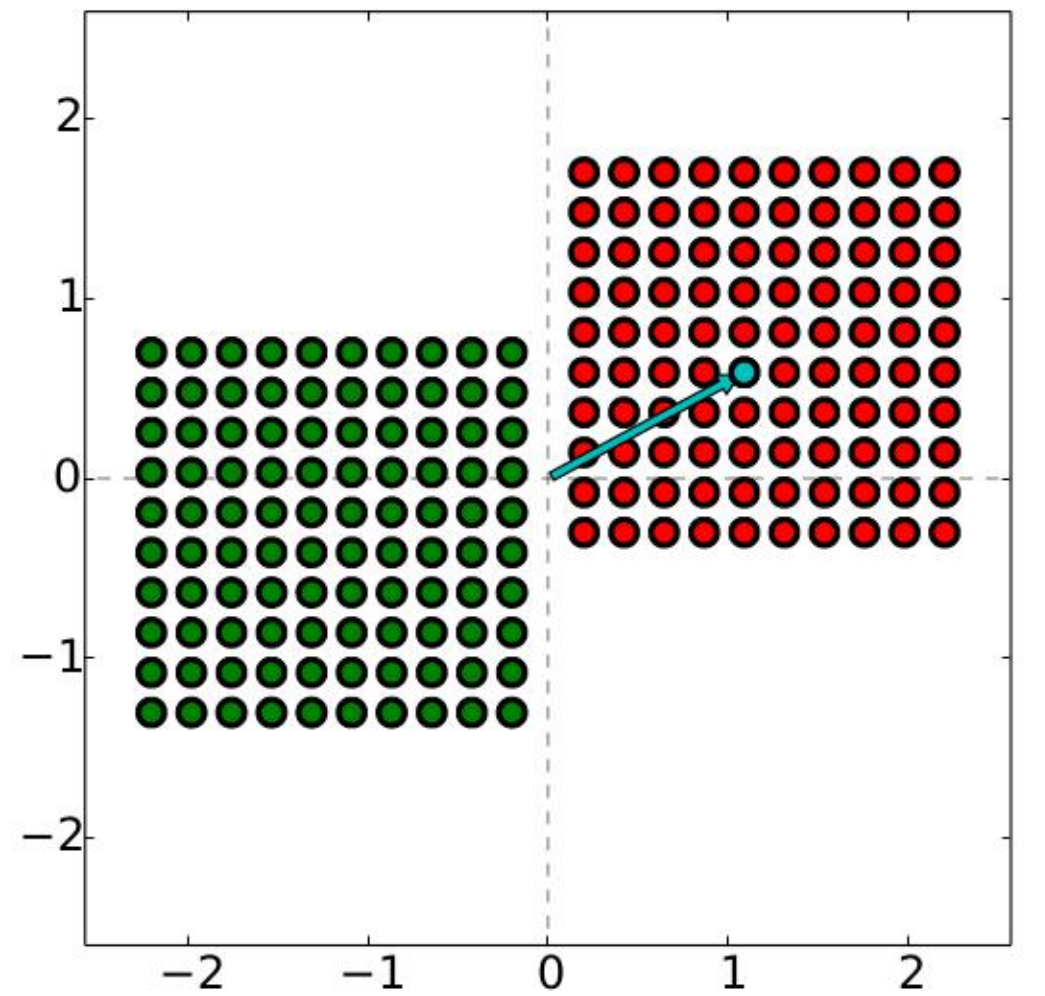
**if**  $\mathbf{x}^k$  missclassified, **then**  $\mathbf{w} \leftarrow \mathbf{w} + \mathbf{x}^k$

**until** all  $\mathbf{x}$  correctly classified

return  $\mathbf{w}$



# Perceptron iterations/loops



$n$  patterns/samples, we are looping over all patterns repeatedly:

Initialize  $\mathbf{w}$

$k \leftarrow 0$

**repeat**

$k \leftarrow (k + 1) \bmod n$

**if**  $\mathbf{x}^k$  misclassified, **then**

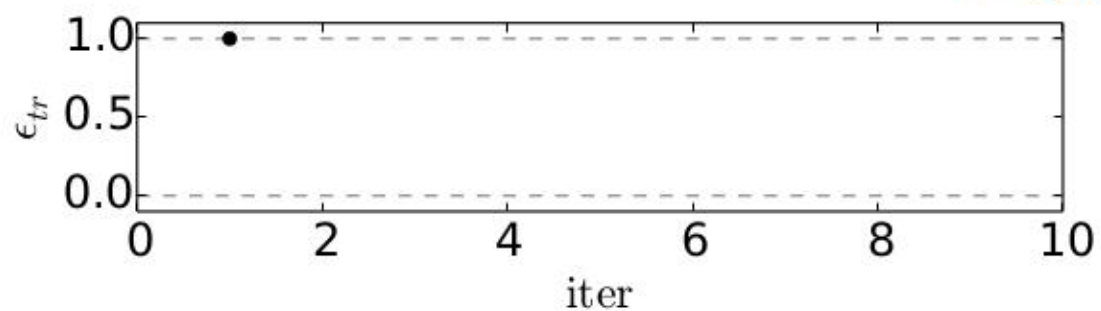
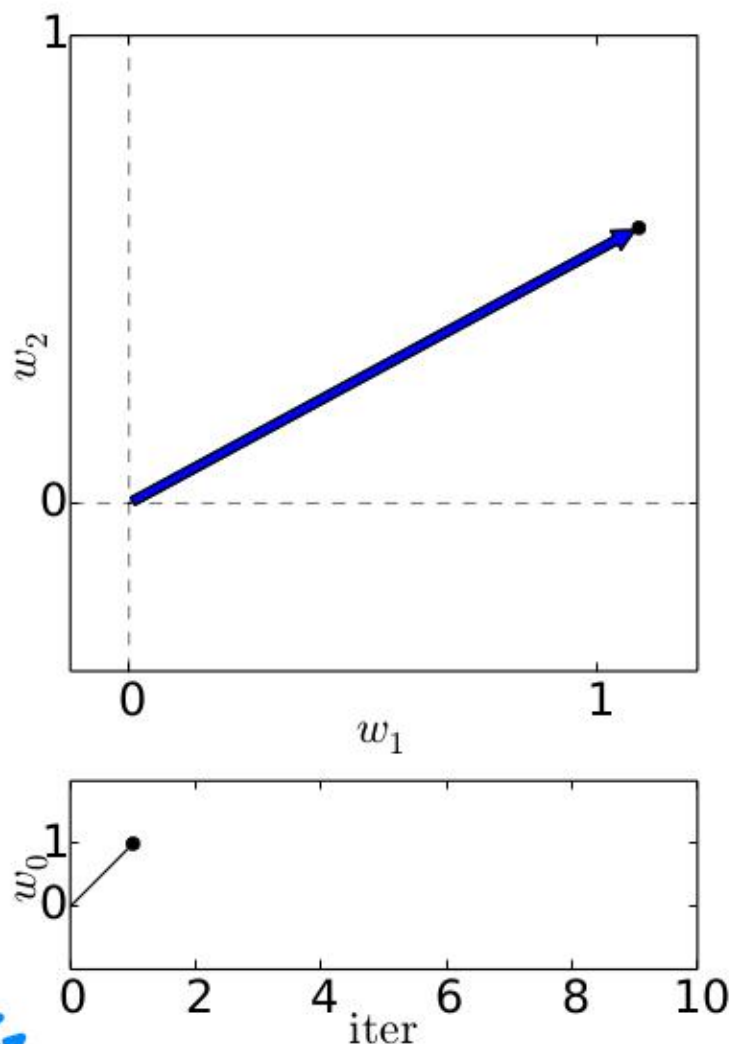
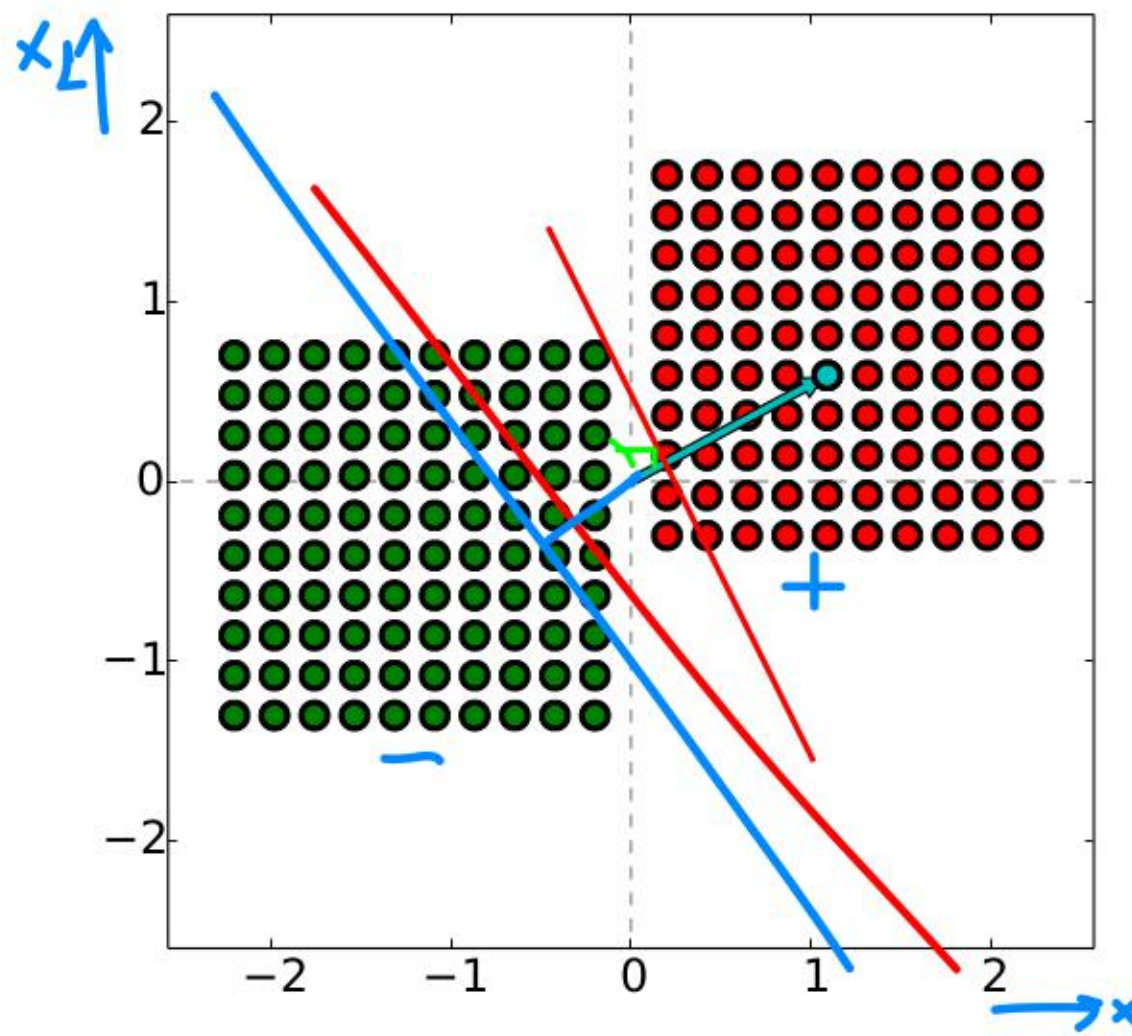
$\mathbf{w} \leftarrow \mathbf{w} + \mathbf{x}^k$

**until** all  $\mathbf{x}$  correctly classified

return  $\mathbf{w}$

(Dark) Blue is  $\mathbf{w}$  after update step. Reds are  $+$ , Greens  $-$ .

# Perceptron iterations/loops



$n$  patterns/samples, we are looping over all patterns repeatedly:

Initialize  $\mathbf{w}$

$k \leftarrow 0$

**repeat**

$k \leftarrow (k + 1) \bmod n$

**if**  $\mathbf{x}^k$  misclassified, **then**

$$\mathbf{w} \leftarrow \mathbf{w} + \mathbf{x}^k$$

**until** all  $\mathbf{x}$  correctly classified

return  $\mathbf{w}$

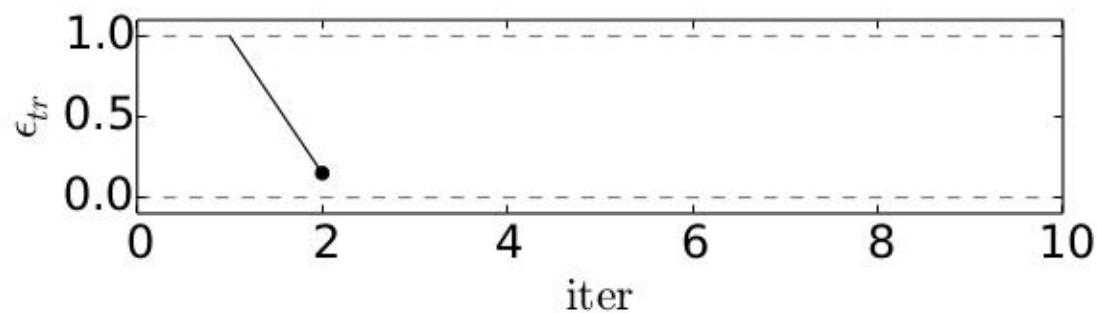
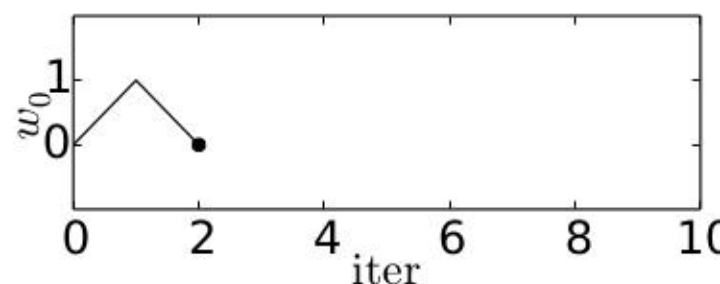
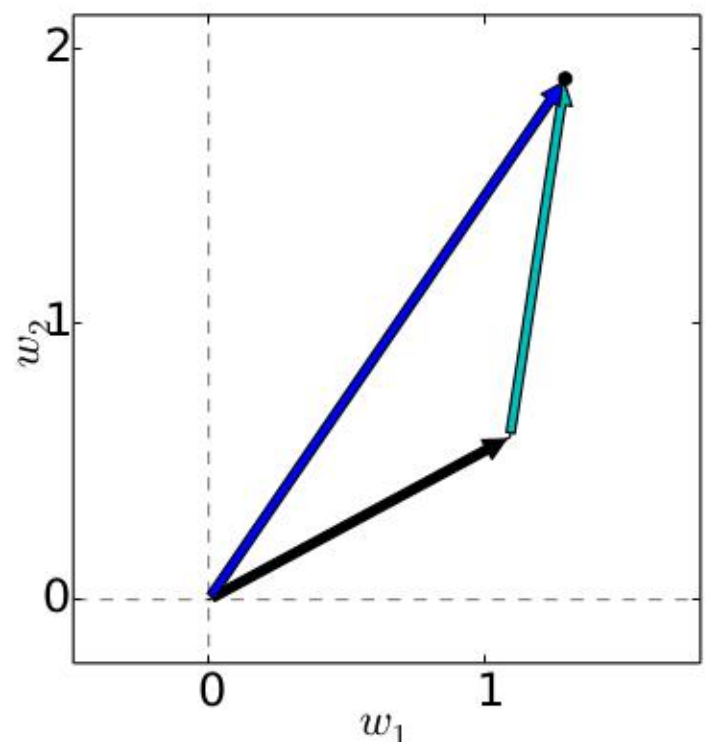
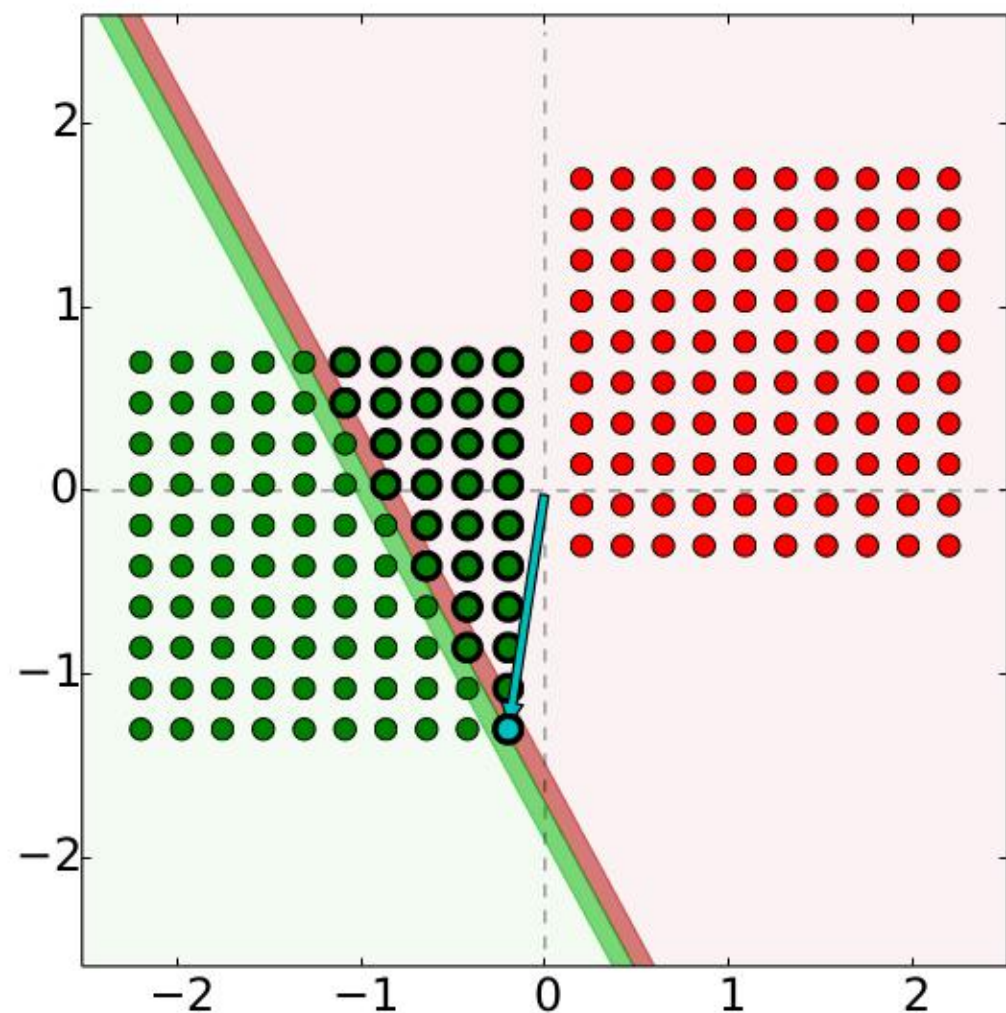
$$\vec{w}^T \vec{x} = \phi$$

$$1.2x_1 + 0.6x_2 + 1 = \phi$$

(Dark) Blue is  $\mathbf{w}$  after update step. Reds are  $+$ , Greens  $-$ .



# Perceptron iterations/loops



$n$  patterns/samples, we are looping over all patterns repeatedly:

Initialize  $\mathbf{w}$

$k \leftarrow 0$

**repeat**

$k \leftarrow (k + 1) \bmod n$

**if**  $\mathbf{x}^k$  misclassified, **then**

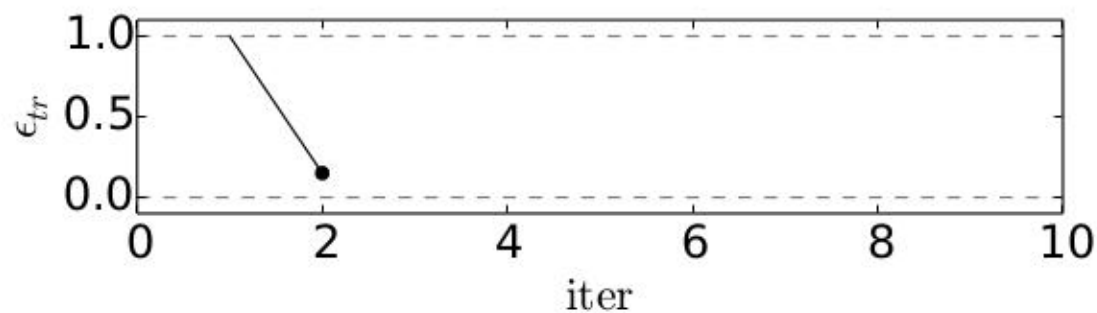
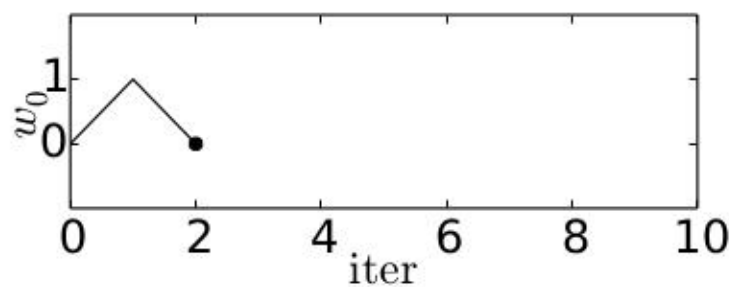
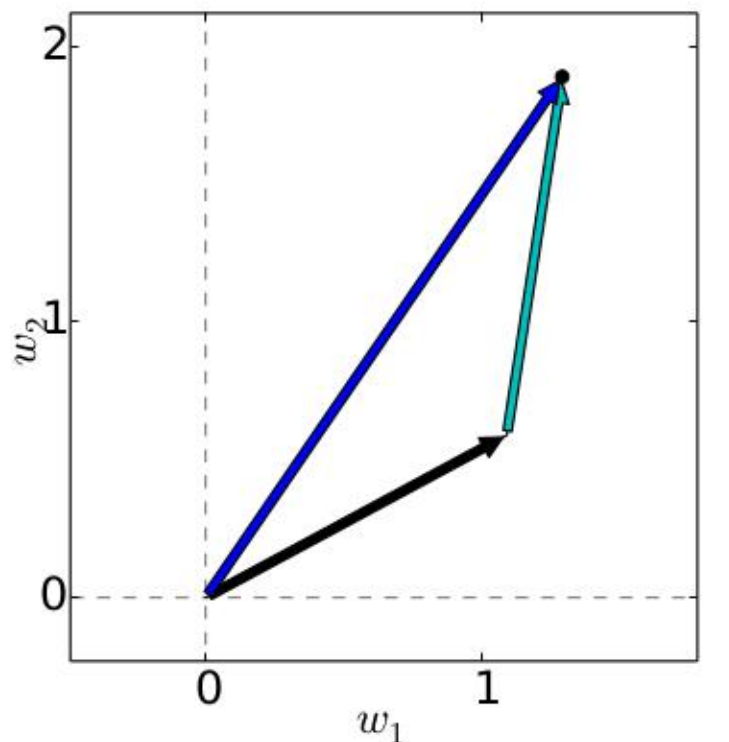
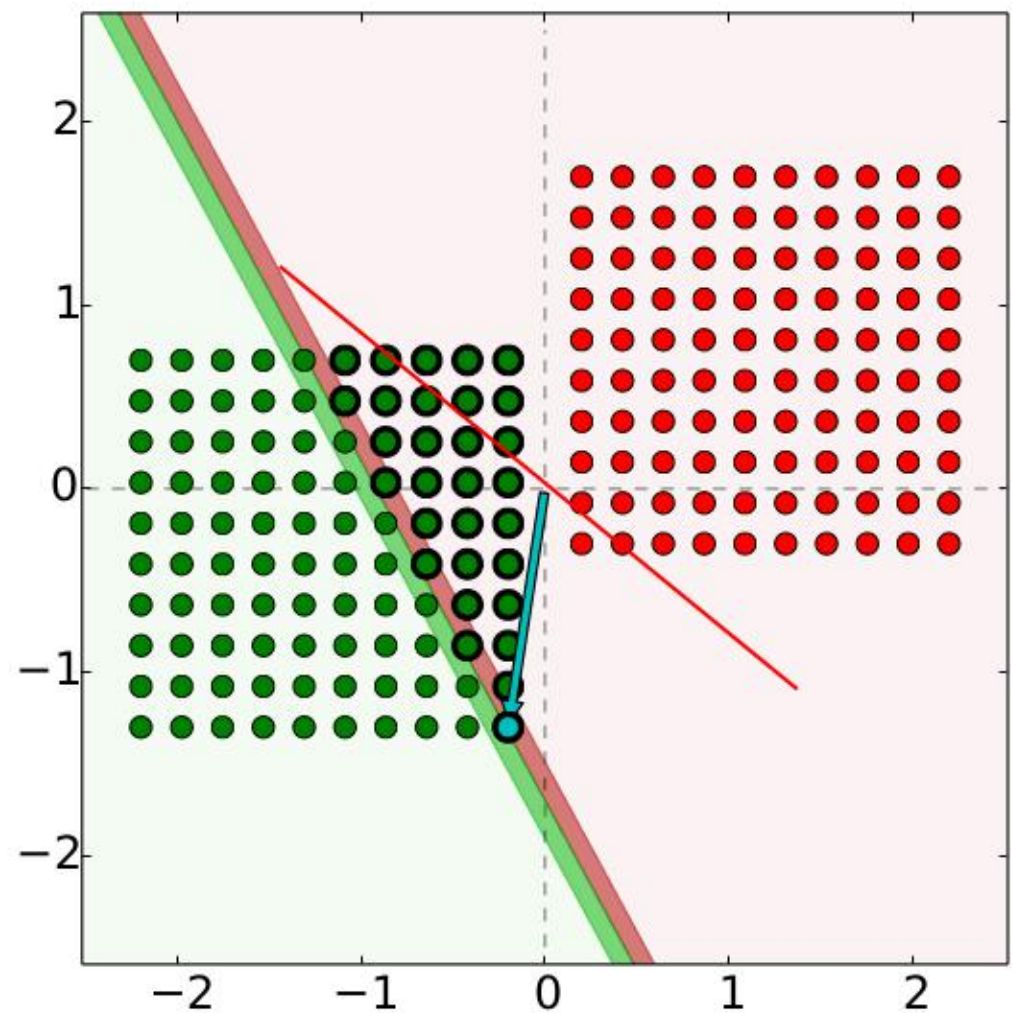
$\mathbf{w} \leftarrow \mathbf{w} + \mathbf{x}^k$

**until** all  $\mathbf{x}$  correctly classified

return  $\mathbf{w}$

(Dark) Blue is  $\mathbf{w}$  after update step. Reds are  $+$ , Greens  $-$ .

# Perceptron iterations/loops



$n$  patterns/samples, we are looping over all patterns repeatedly:

Initialize  $\mathbf{w}$

$k \leftarrow 0$

**repeat**

$k \leftarrow (k + 1) \bmod n$

**if**  $\mathbf{x}^k$  misclassified, **then**

$\mathbf{w} \leftarrow \mathbf{w} + \mathbf{x}^k$

**until** all  $\mathbf{x}$  correctly classified

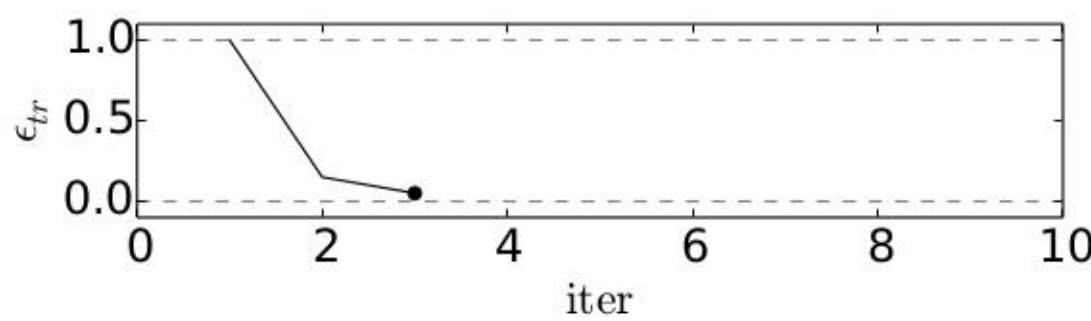
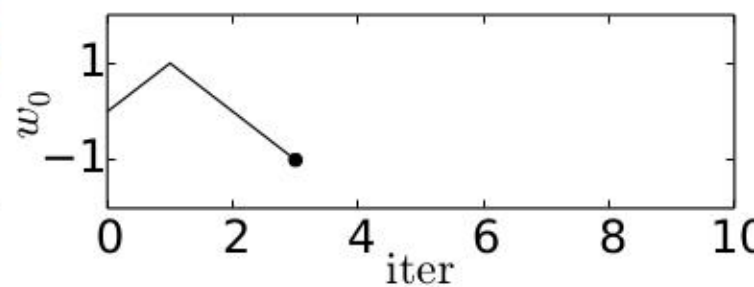
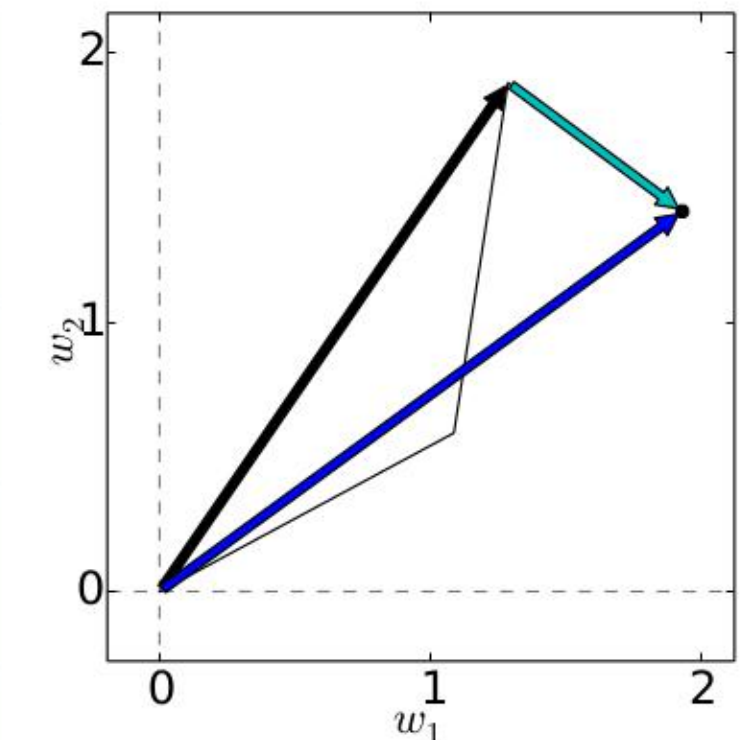
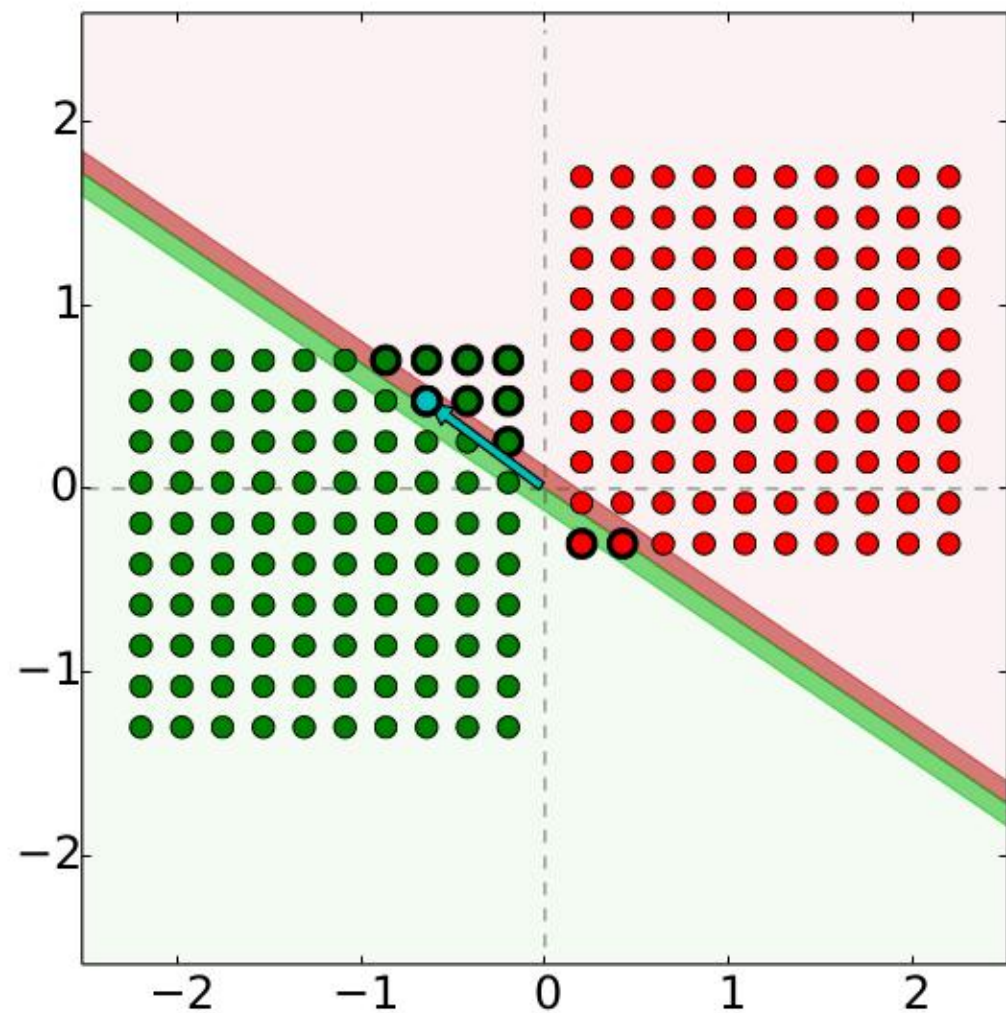
return  $\mathbf{w}$



(Dark) Blue is  $\mathbf{w}$  after update step. Reds are +, Greens -.



# Perceptron iterations/loops



$n$  patterns/samples, we are looping over all patterns repeatedly:

Initialize  $\mathbf{w}$

$k \leftarrow 0$

**repeat**

$k \leftarrow (k + 1) \bmod n$

**if**  $\mathbf{x}^k$  misclassified, **then**

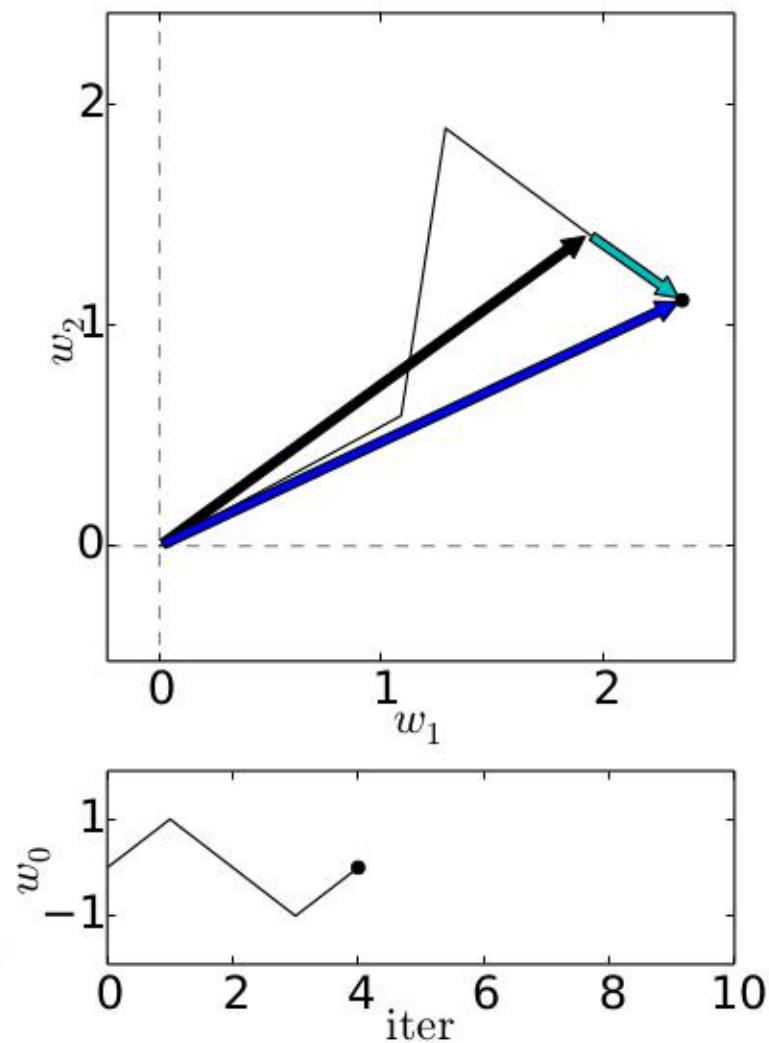
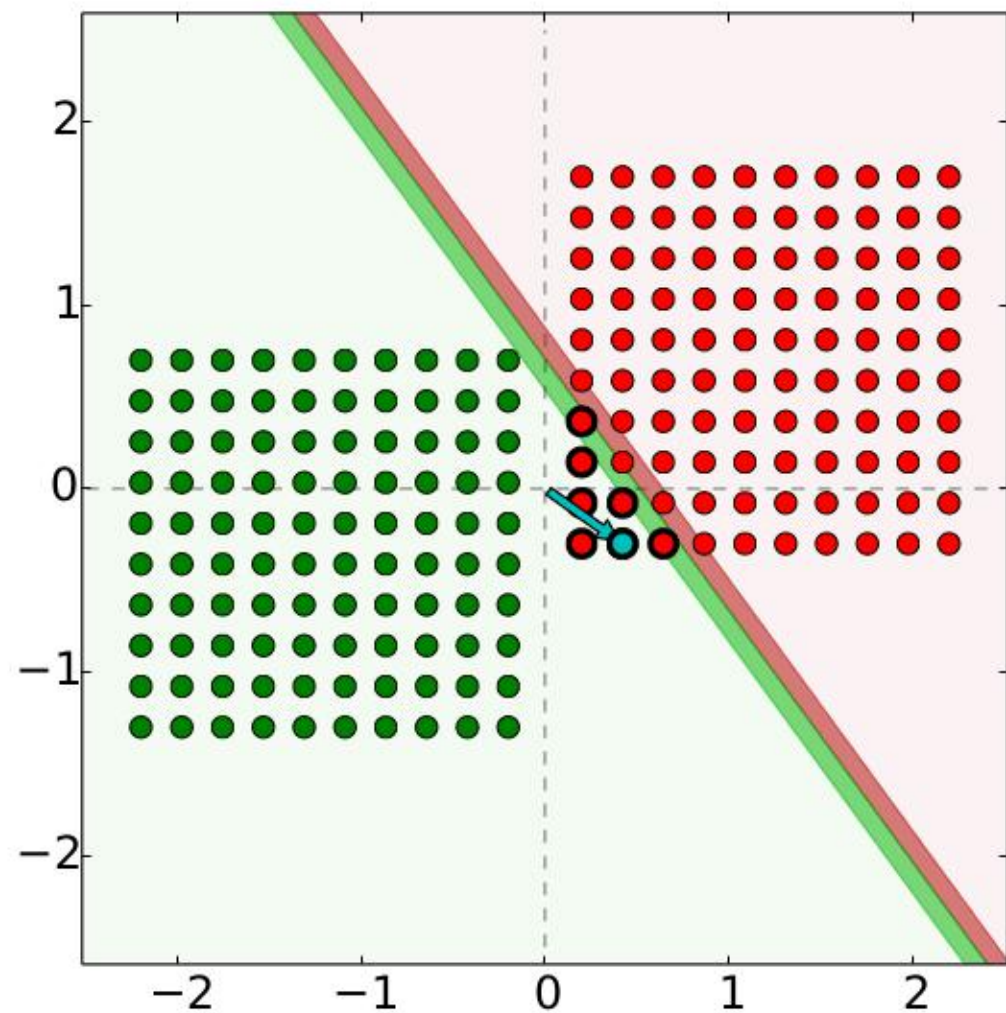
$$\mathbf{w} \leftarrow \mathbf{w} + \mathbf{x}^k$$

**until** all  $\mathbf{x}$  correctly classified

return  $\mathbf{w}$

(Dark) Blue is  $\mathbf{w}$  after update step. Reds are  $+$ , Greens  $-$ .

# Perceptron iterations/loops



$n$  patterns/samples, we are looping over all patterns repeatedly:

Initialize  $\mathbf{w}$

$k \leftarrow 0$

**repeat**

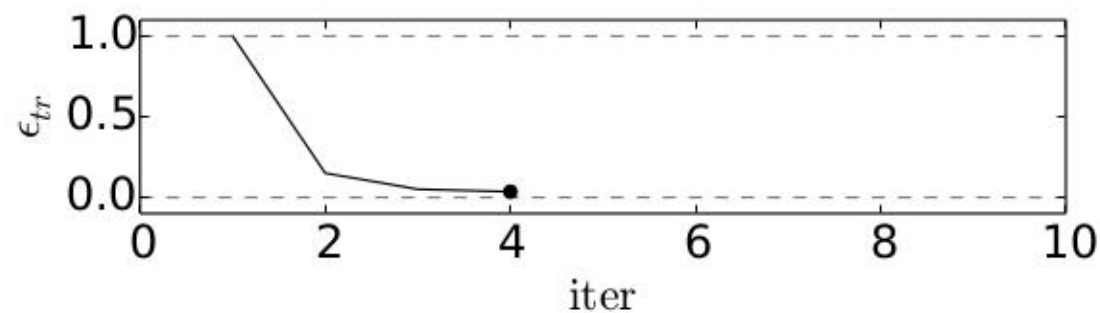
$k \leftarrow (k + 1) \bmod n$

**if**  $\mathbf{x}^k$  misclassified, **then**

$\mathbf{w} \leftarrow \mathbf{w} + \mathbf{x}^k$

**until** all  $\mathbf{x}$  correctly classified

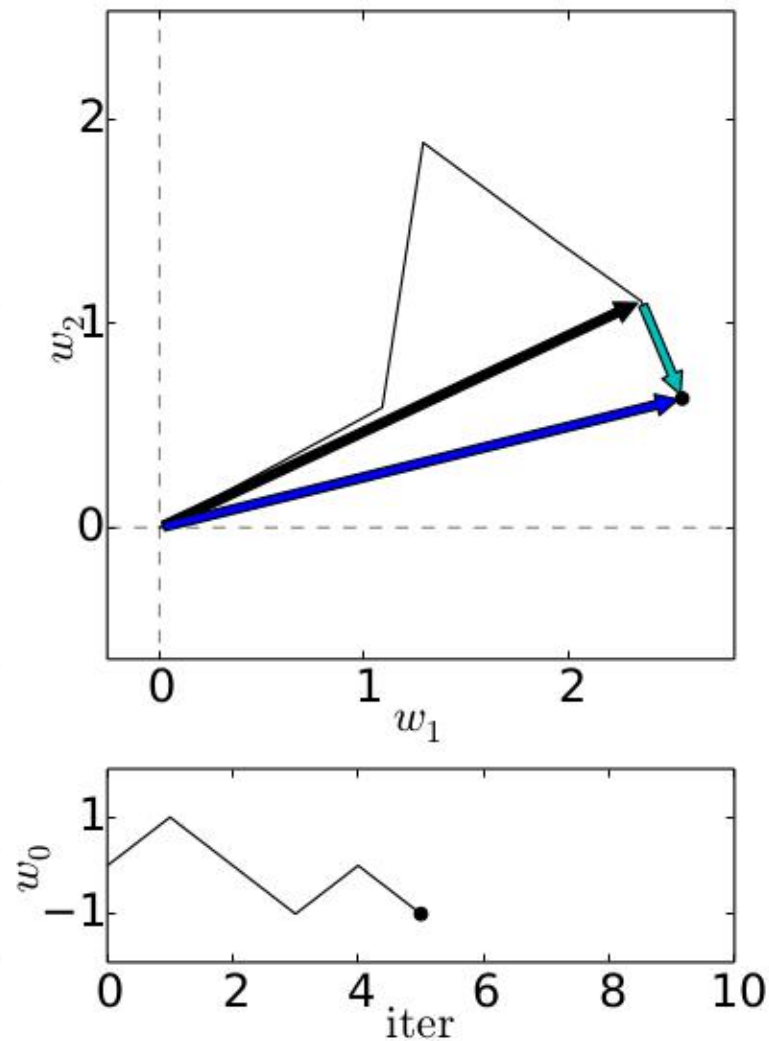
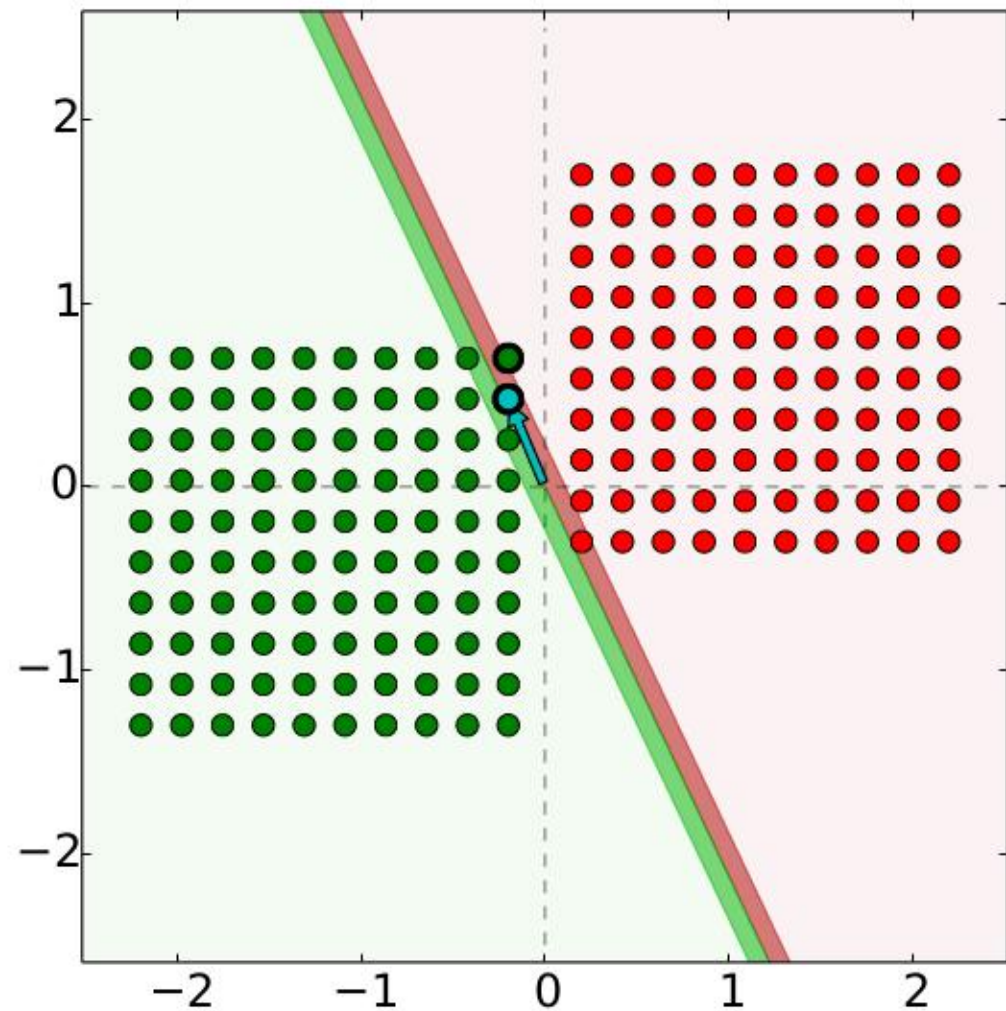
return  $\mathbf{w}$



(Dark) Blue is  $\mathbf{w}$  after update step. Reds are  $+$ , Greens  $-$ .



# Perceptron iterations/loops



$n$  patterns/samples, we are looping over all patterns repeatedly:

Initialize  $\mathbf{w}$

$k \leftarrow 0$

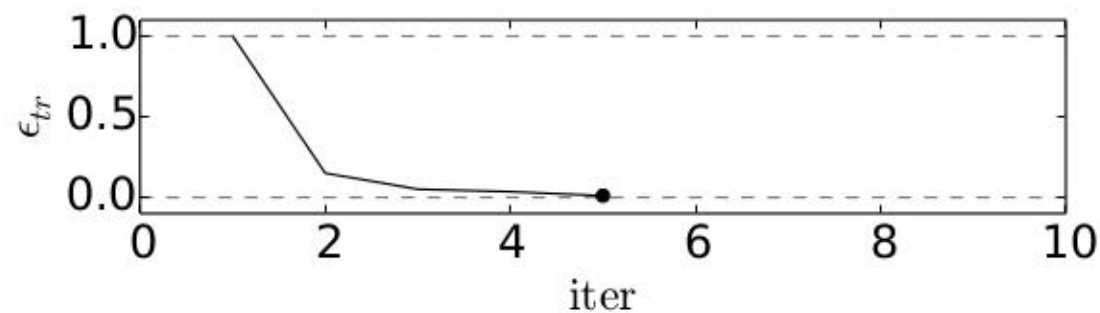
**repeat**

$k \leftarrow (k + 1) \bmod n$

**if**  $\mathbf{x}^k$  misclassified, **then**

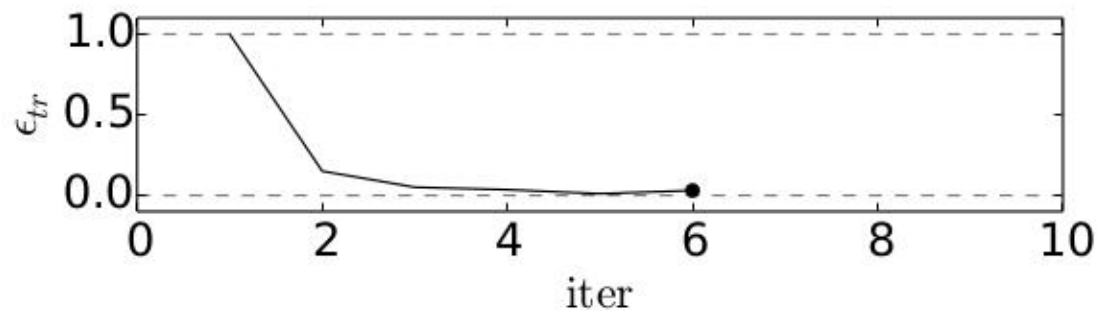
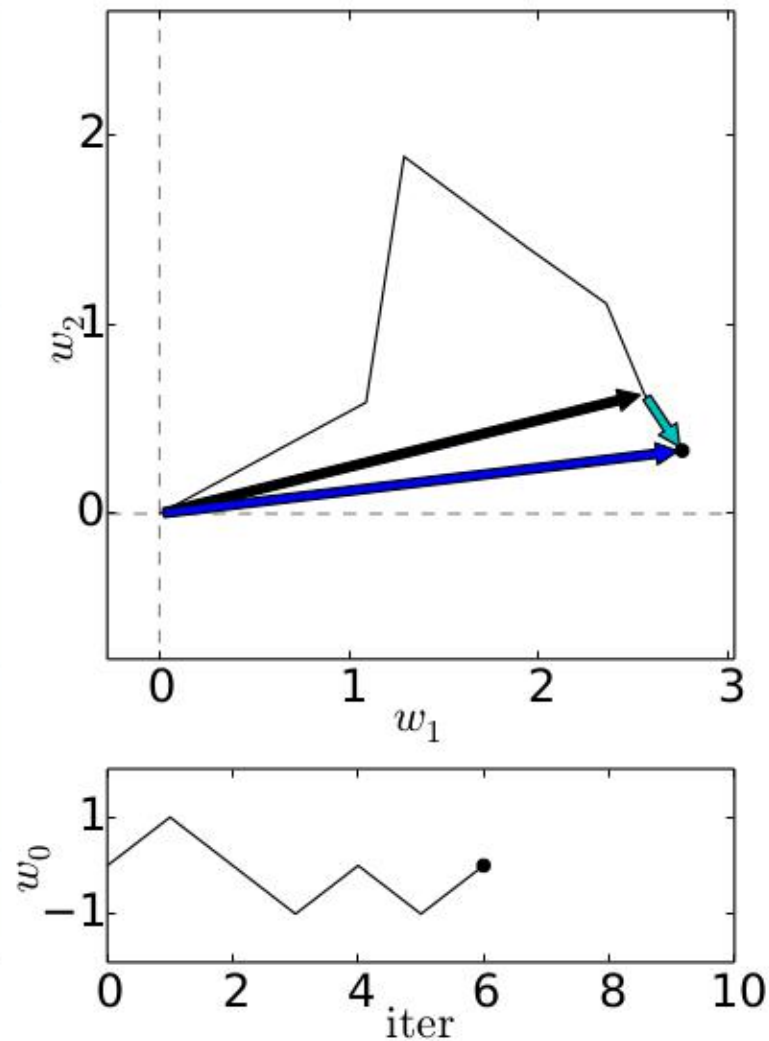
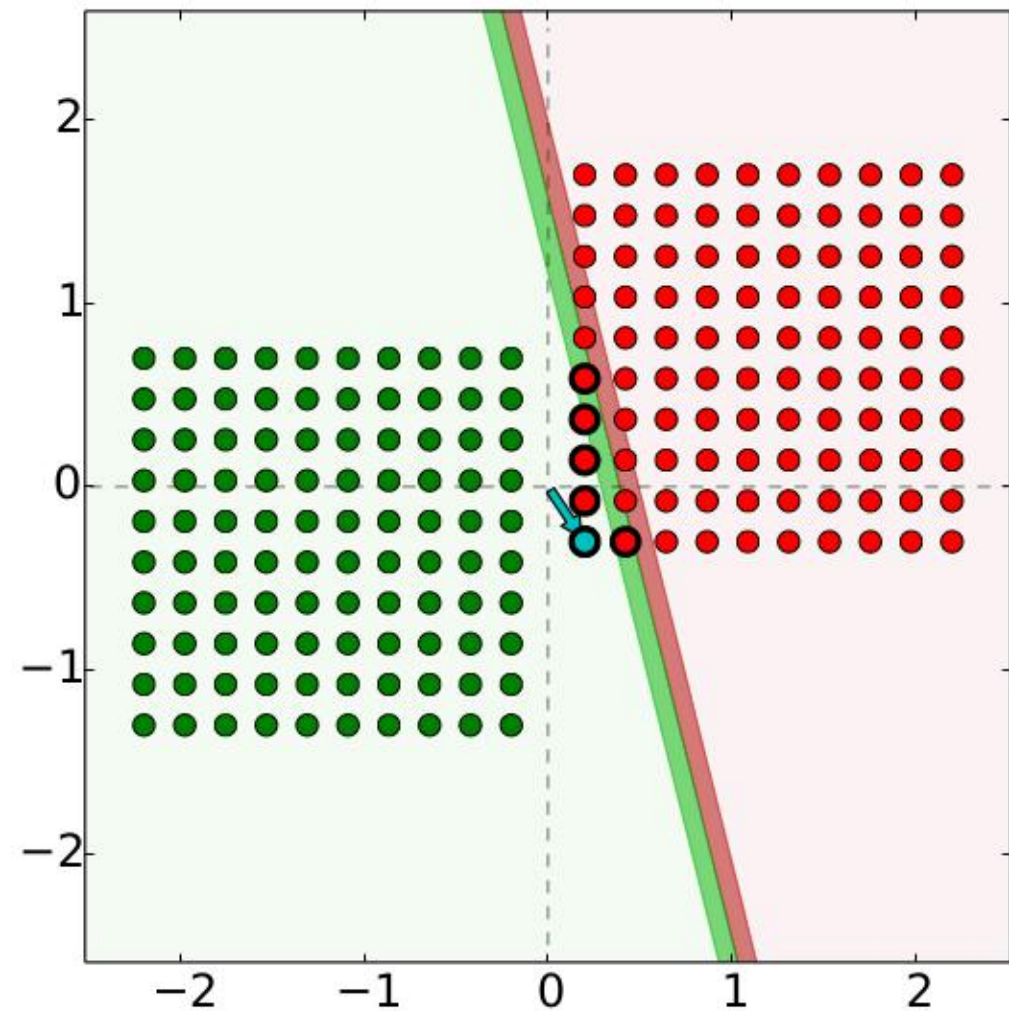
$$\mathbf{w} \leftarrow \mathbf{w} + \mathbf{x}^k$$

**until** all  $\mathbf{x}$  correctly classified  
return  $\mathbf{w}$



(Dark) Blue is  $\mathbf{w}$  after update step. Reds are  $+$ , Greens  $-$ .

# Perceptron iterations/loops



$n$  patterns/samples, we are looping over all patterns repeatedly:

Initialize  $\mathbf{w}$

$k \leftarrow 0$

**repeat**

$k \leftarrow (k + 1) \bmod n$

**if**  $\mathbf{x}^k$  misclassified, **then**

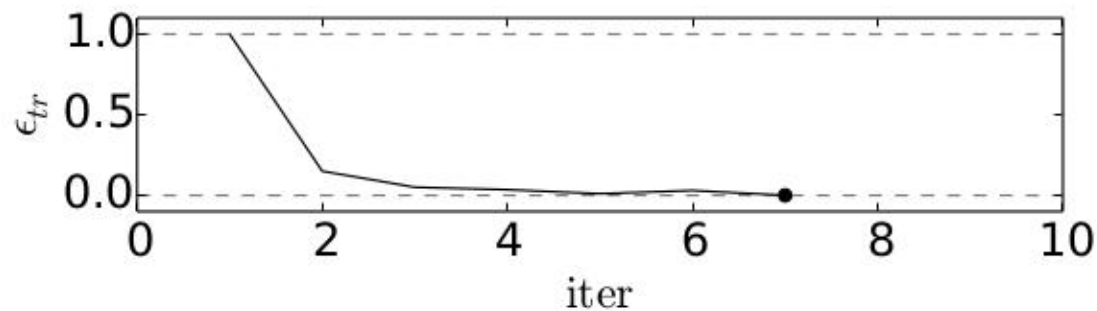
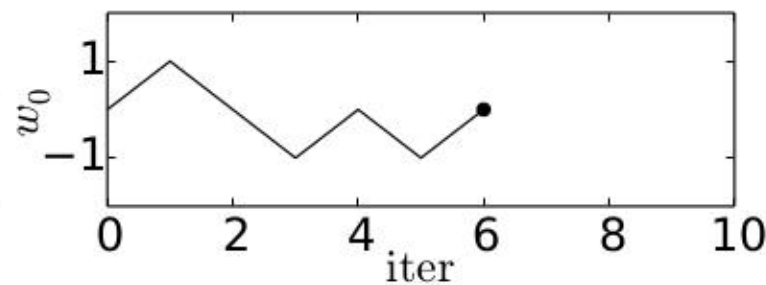
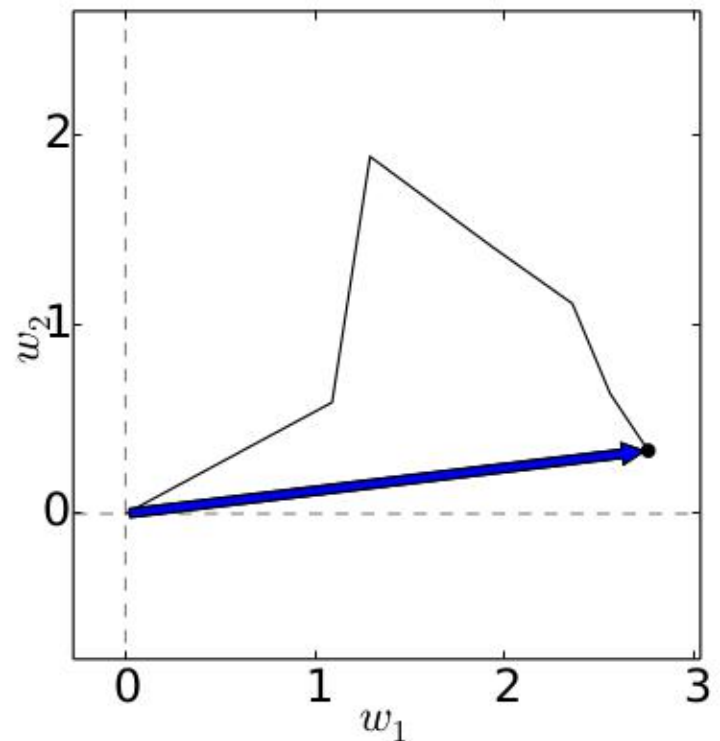
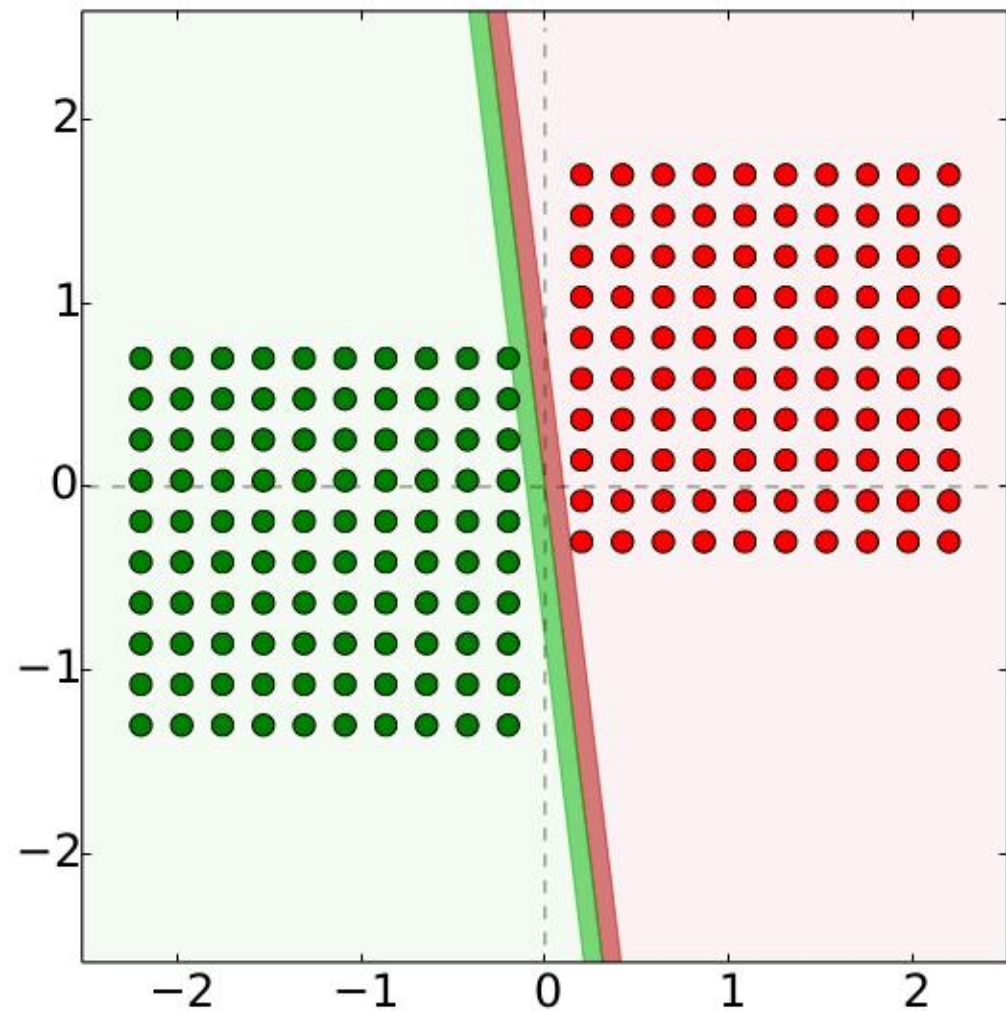
$\mathbf{w} \leftarrow \mathbf{w} + \mathbf{x}^k$

**until** all  $\mathbf{x}$  correctly classified  
**return**  $\mathbf{w}$

(Dark) Blue is  $\mathbf{w}$  after update step. Reds are  $+$ , Greens  $-$ .



# Perceptron iterations/loops



$n$  patterns/samples, we are looping over all patterns repeatedly:

Initialize  $\mathbf{w}$

$k \leftarrow 0$

**repeat**

$k \leftarrow (k + 1) \bmod n$

**if**  $\mathbf{x}^k$  misclassified, **then**

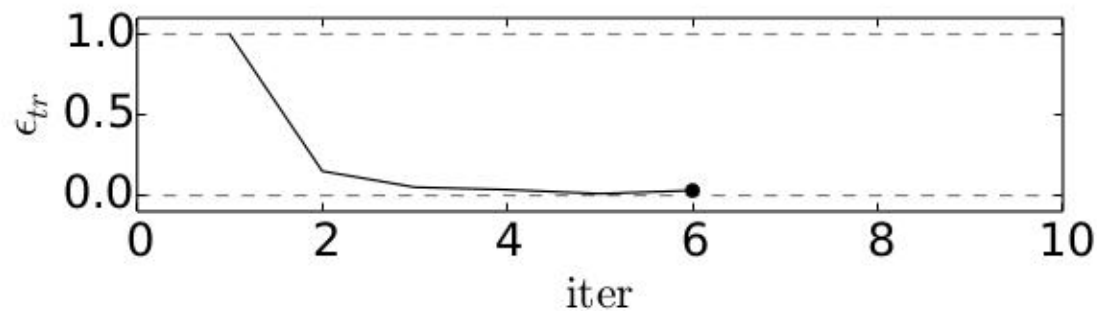
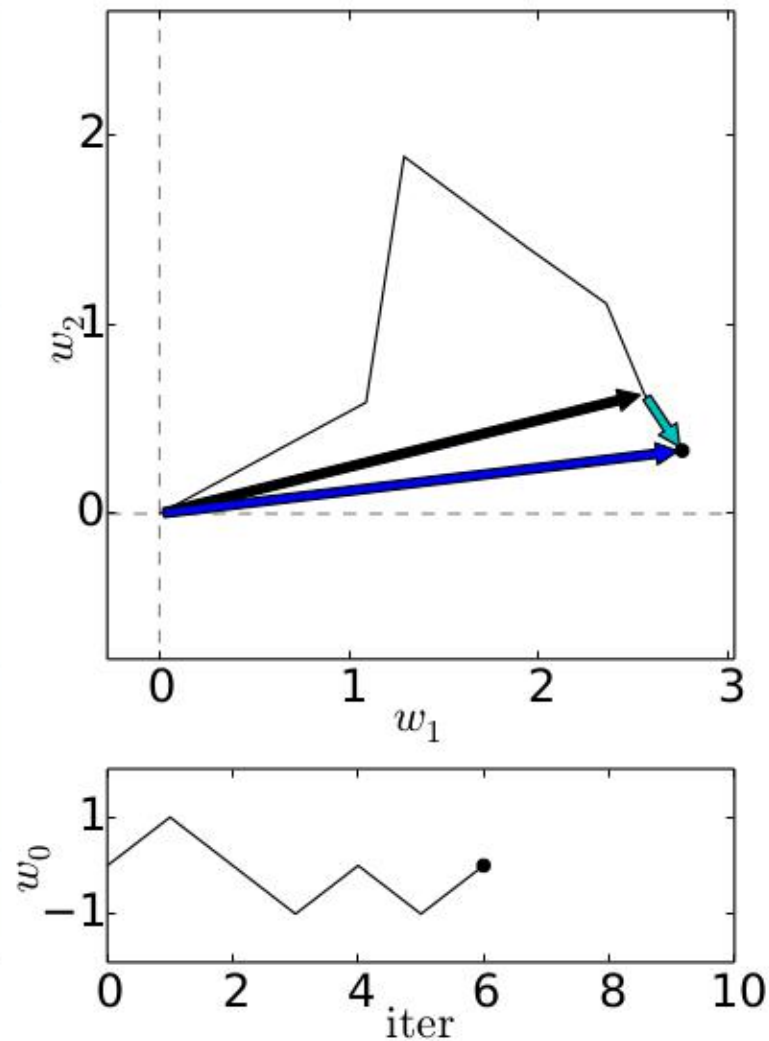
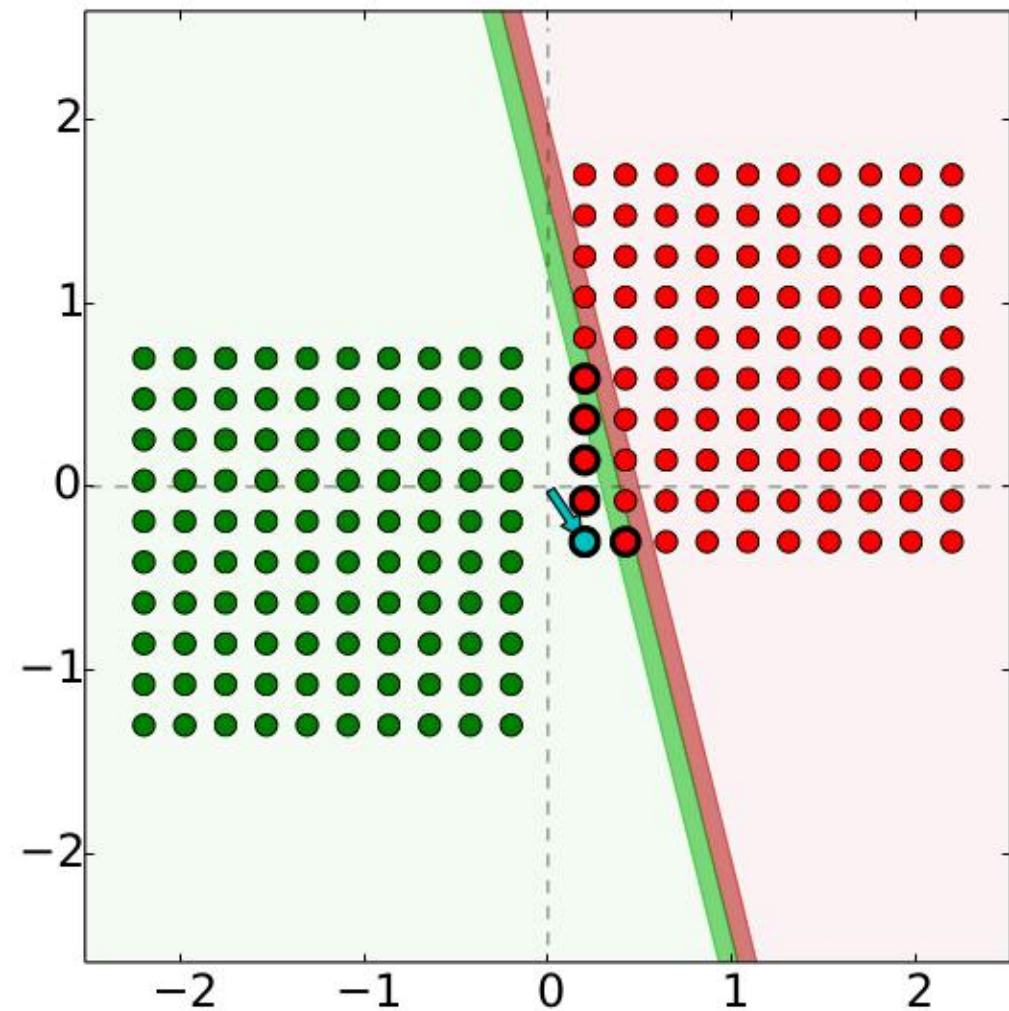
$\mathbf{w} \leftarrow \mathbf{w} + \mathbf{x}^k$

**until** all  $\mathbf{x}$  correctly classified  
**return**  $\mathbf{w}$

(Dark) Blue is  $\mathbf{w}$  after update step. Reds are  $+$ , Greens  $-$ .



# Perceptron iterations/loops



$n$  patterns/samples, we are looping over all patterns repeatedly:

Initialize  $\mathbf{w}$

$k \leftarrow 0$

repeat

$k \leftarrow (k + 1) \bmod n$

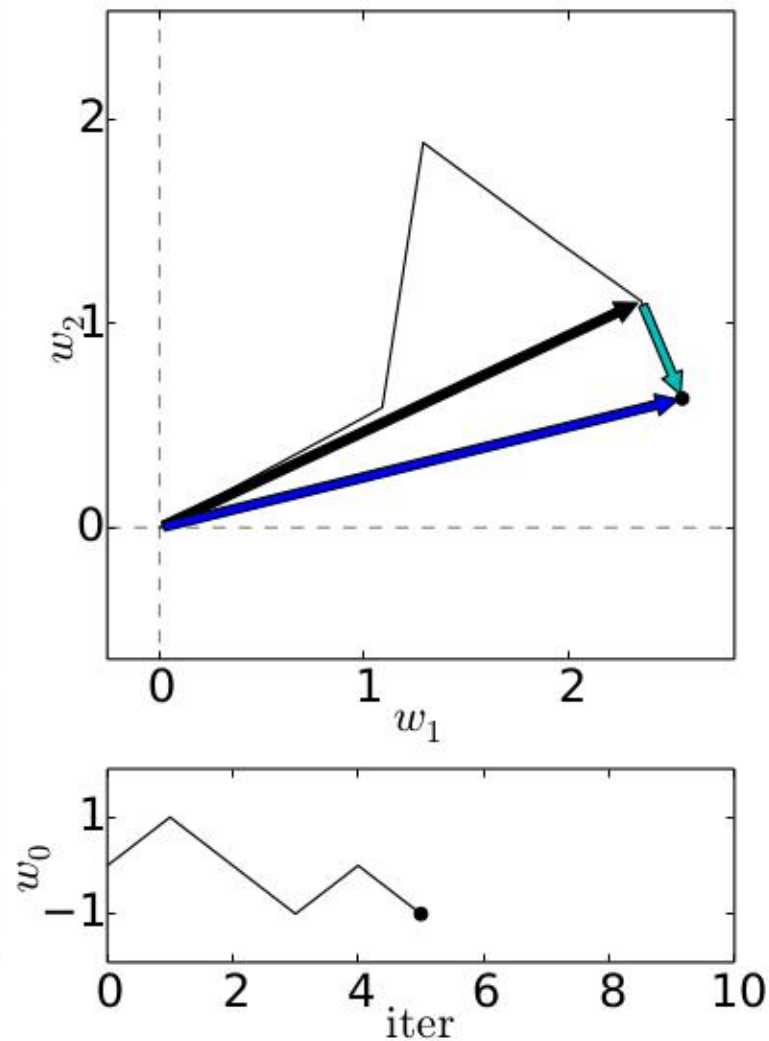
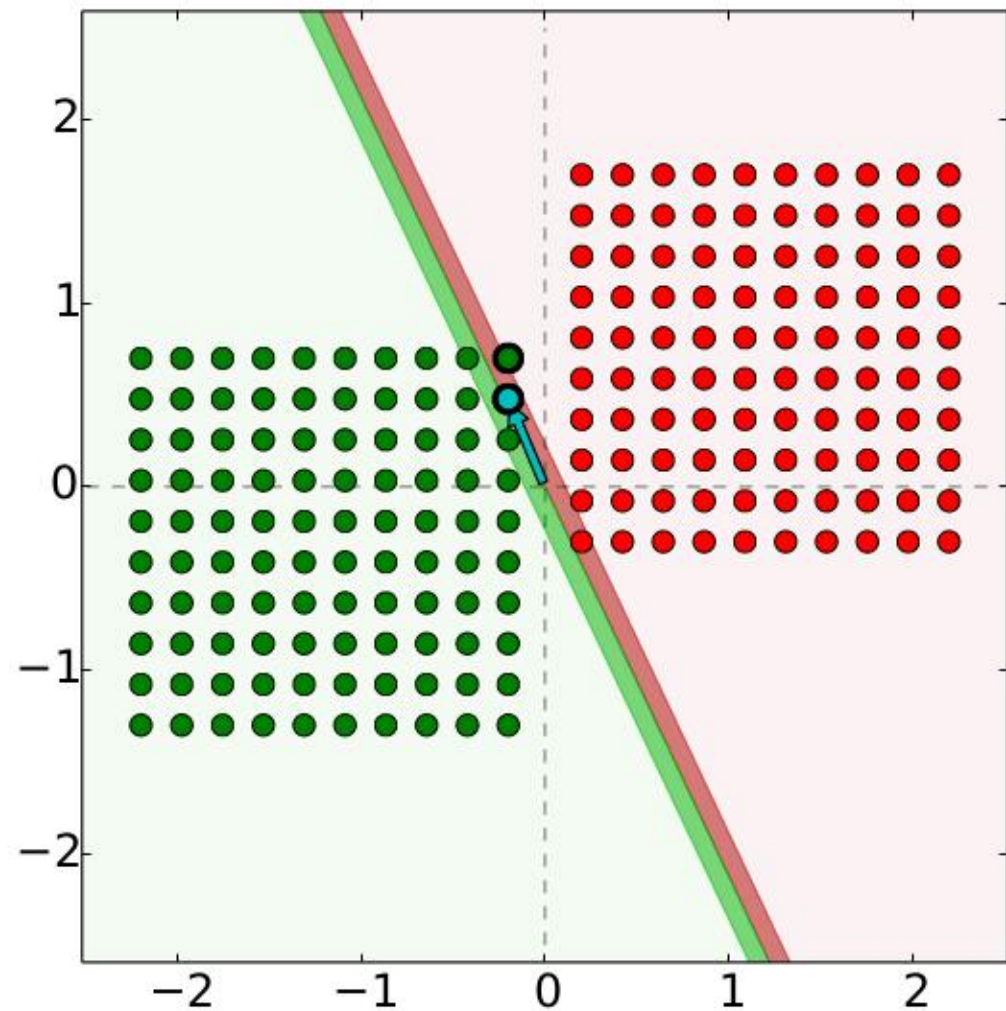
if  $\mathbf{x}^k$  misclassified, then

$$\mathbf{w} \leftarrow \mathbf{w} + \mathbf{x}^k$$

until all  $\mathbf{x}$  correctly classified  
return  $\mathbf{w}$

(Dark) Blue is  $\mathbf{w}$  after update step. Reds are  $+$ , Greens  $-$ .

# Perceptron iterations/loops



$n$  patterns/samples, we are looping over all patterns repeatedly:

Initialize  $\mathbf{w}$

$k \leftarrow 0$

**repeat**

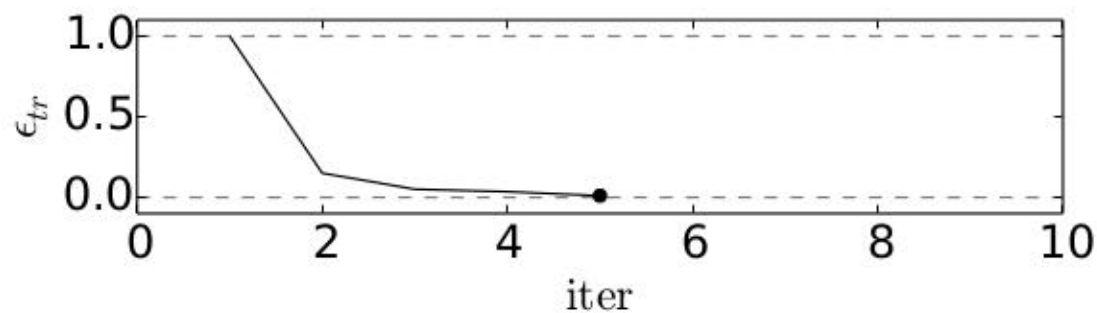
$k \leftarrow (k + 1) \bmod n$

**if**  $\mathbf{x}^k$  misclassified, **then**

$$\mathbf{w} \leftarrow \mathbf{w} + \mathbf{x}^k$$

**until** all  $\mathbf{x}$  correctly classified

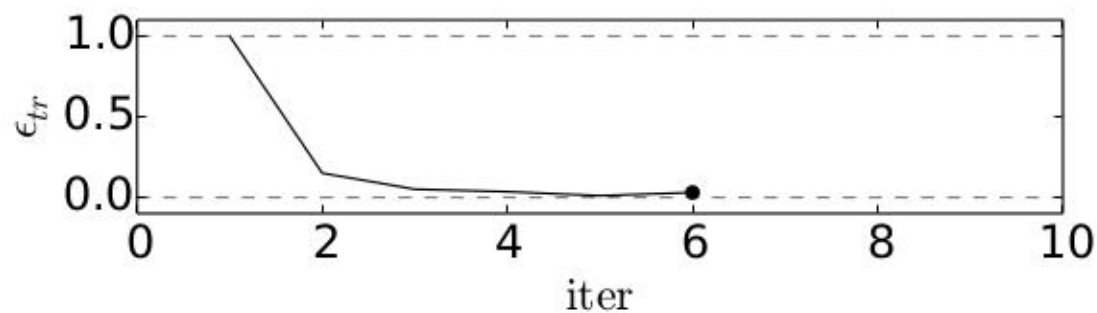
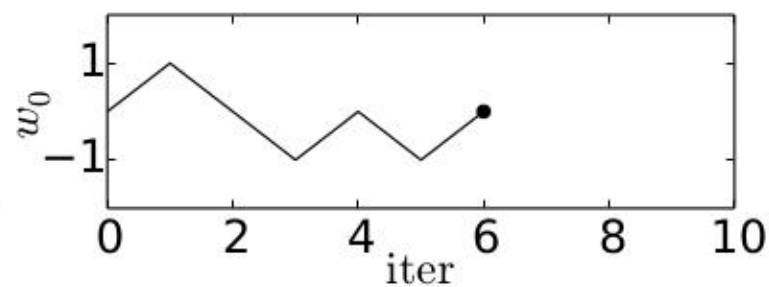
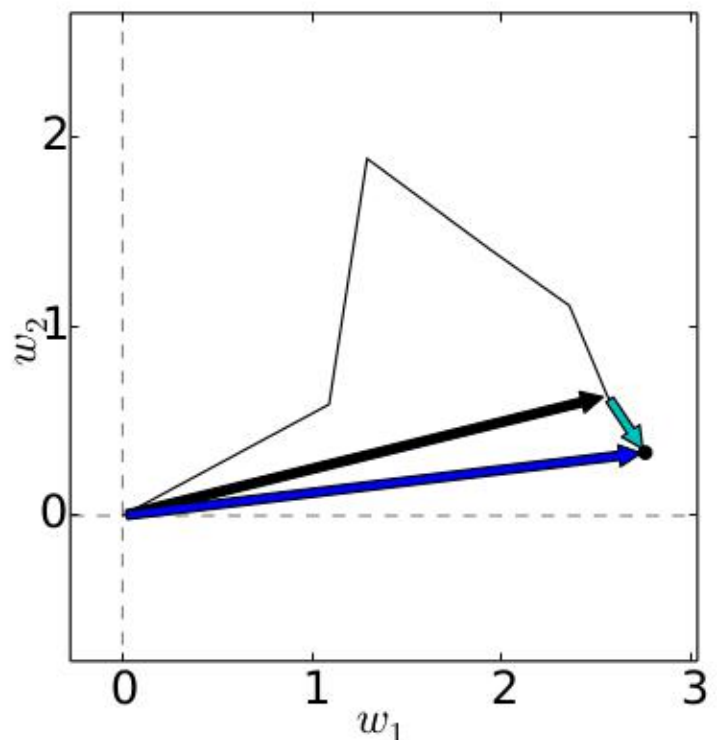
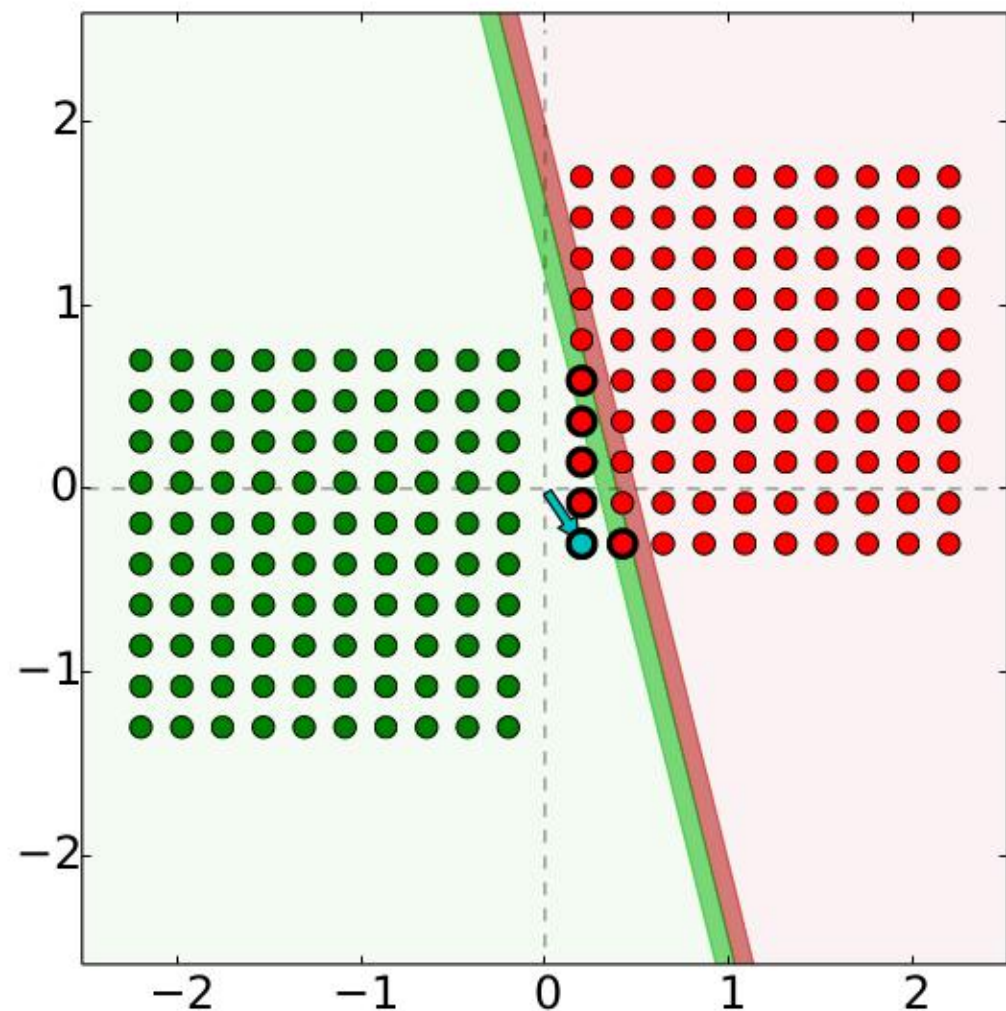
return  $\mathbf{w}$



(Dark) Blue is  $\mathbf{w}$  after update step. Reds are  $+$ , Greens  $-$ .



# Perceptron iterations/loops



$n$  patterns/samples, we are looping over all patterns repeatedly:

Initialize  $\mathbf{w}$

$k \leftarrow 0$

**repeat**

$k \leftarrow (k + 1) \bmod n$

**if**  $\mathbf{x}^k$  missclassified, **then**

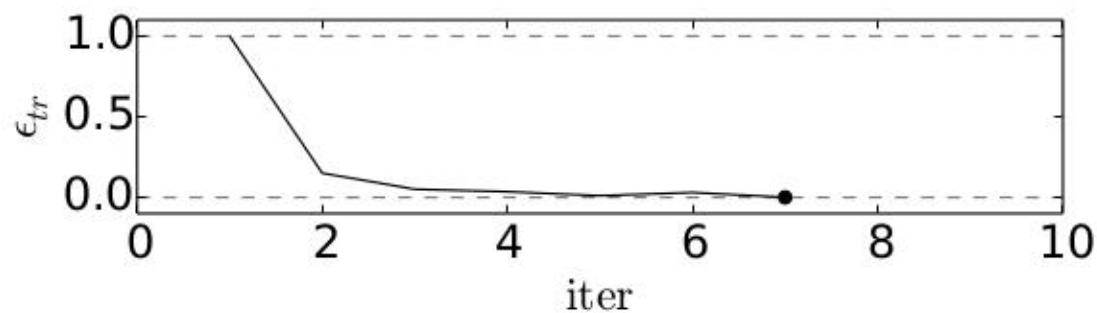
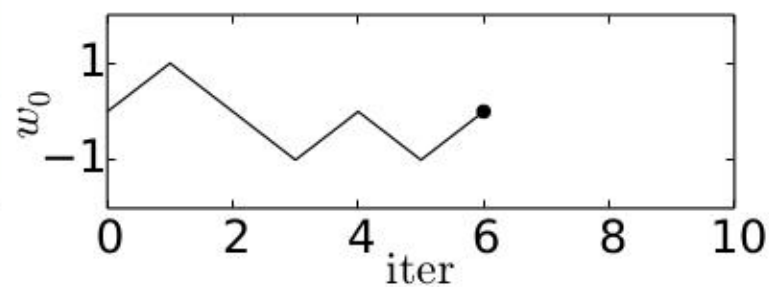
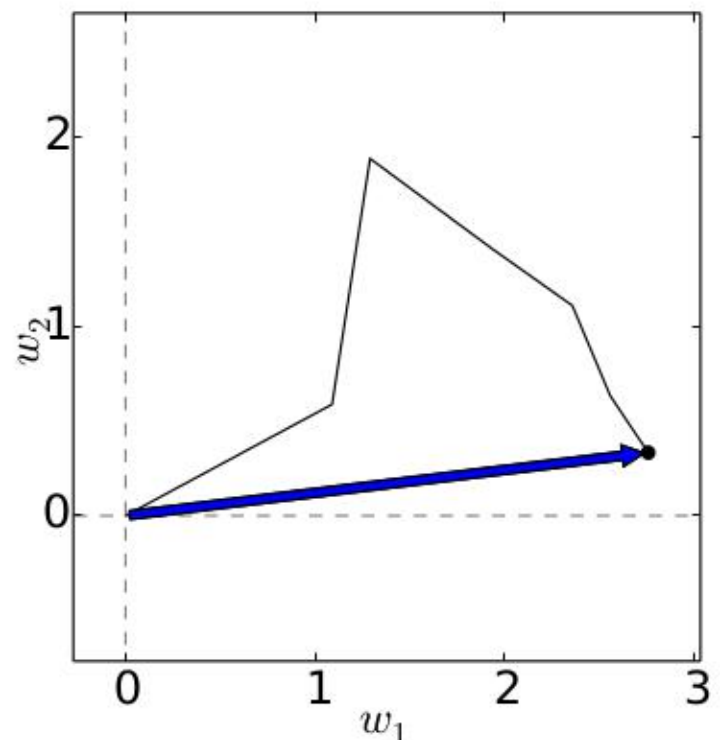
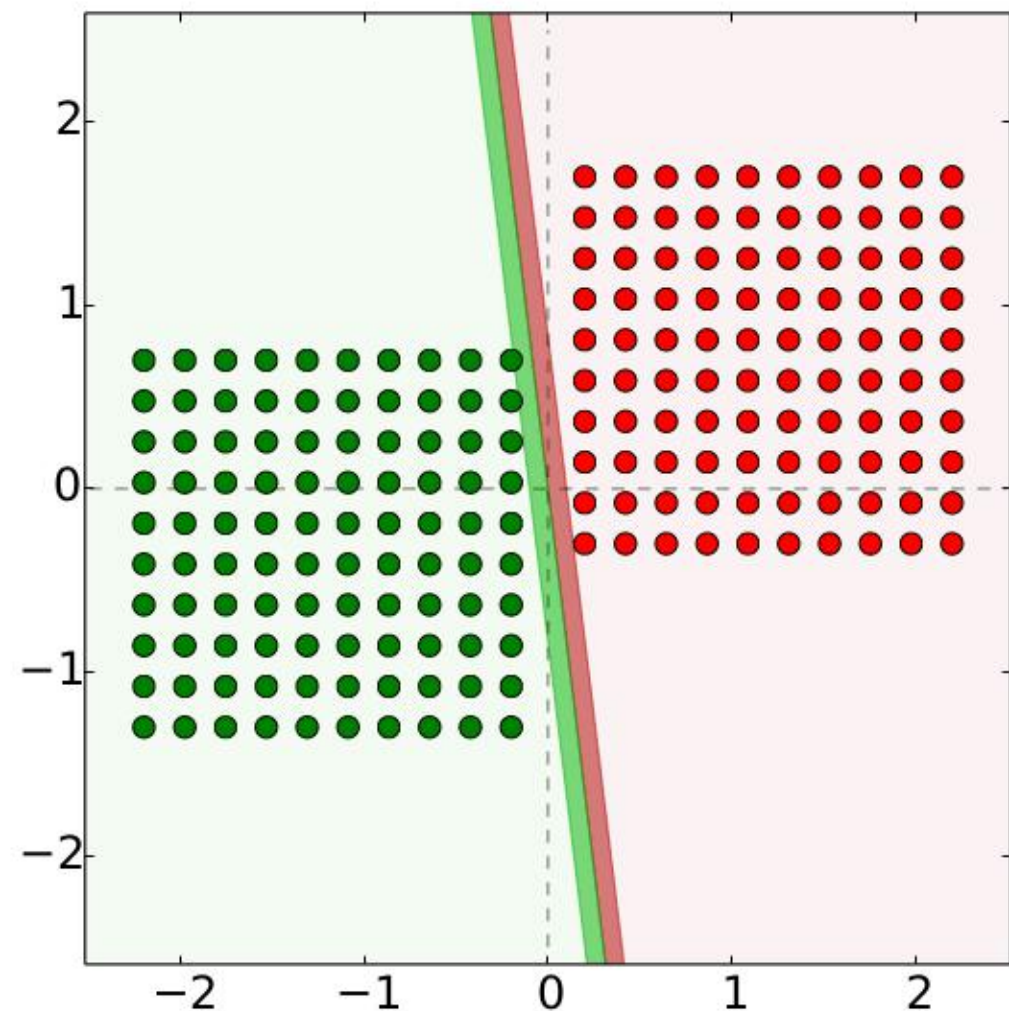
$\mathbf{w} \leftarrow \mathbf{w} + \mathbf{x}^k$

**until** all  $\mathbf{x}$  correctly classified

return  $\mathbf{w}$

(Dark) Blue is  $\mathbf{w}$  after update step. Reds are  $+$ , Greens  $-$ .

# Perceptron iterations/loops



$n$  patterns/samples, we are looping over all patterns repeatedly:

Initialize  $\mathbf{w}$

$k \leftarrow 0$

**repeat**

$k \leftarrow (k + 1) \bmod n$

**if**  $\mathbf{x}^k$  misclassified, **then**

$\mathbf{w} \leftarrow \mathbf{w} + \mathbf{x}^k$

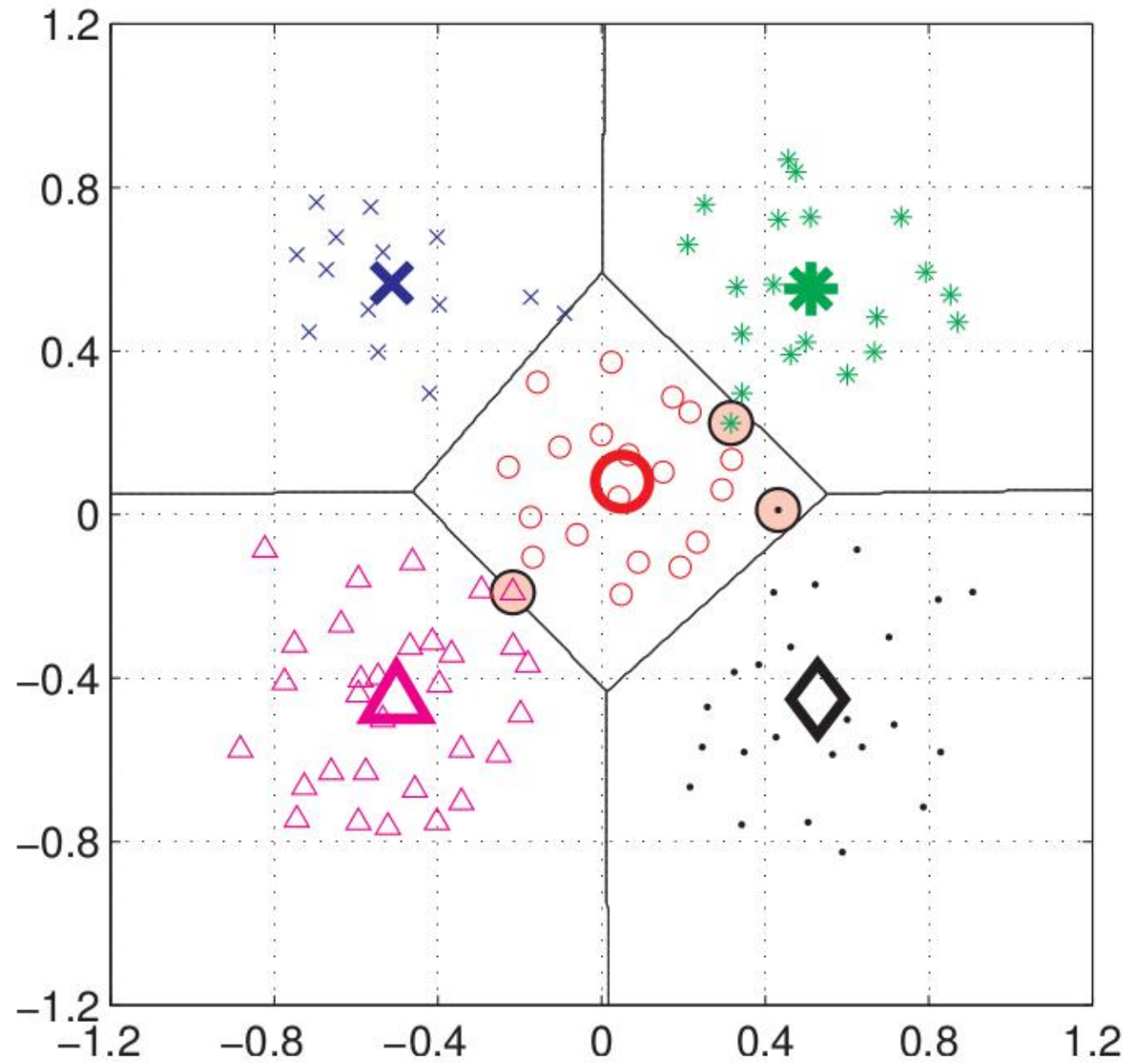
**until** all  $\mathbf{x}$  correctly classified  
**return**  $\mathbf{w}$

(Dark) Blue is  $\mathbf{w}$  after update step. Reds are  $+$ , Greens  $-$ .

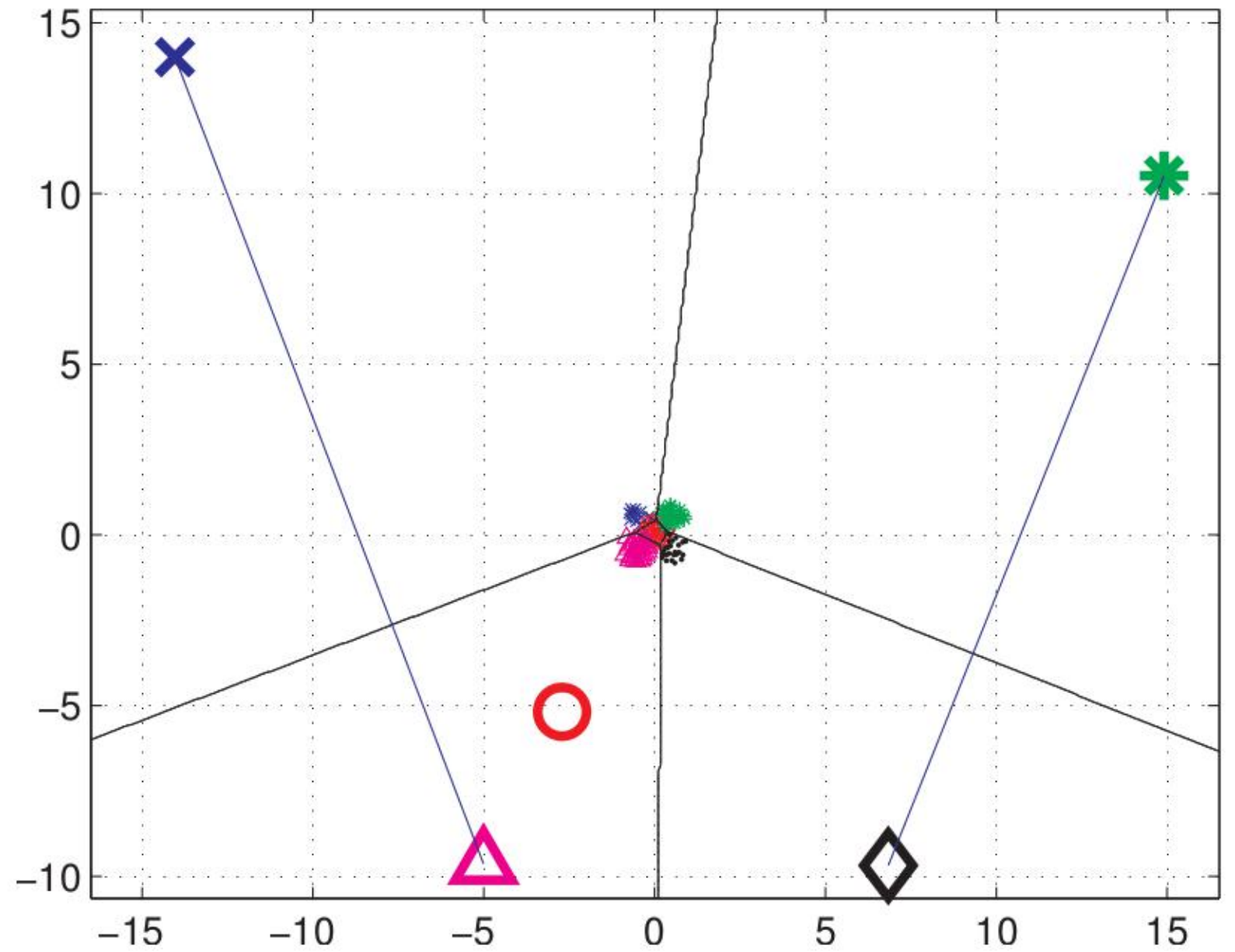


# Etalons: means vs. found by perceptron

minimum distance from etalons



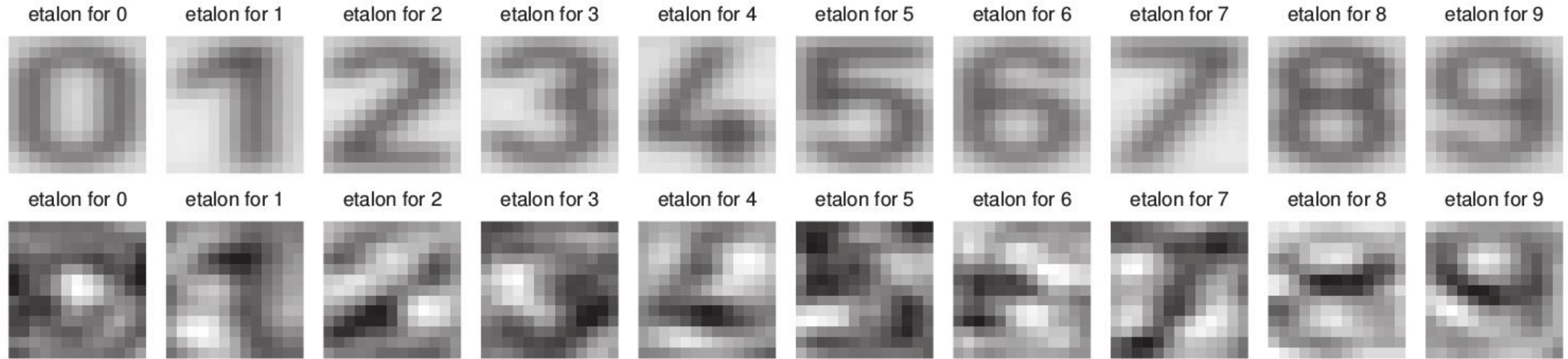
Etalons and separating hyperplanes found by perceptron



Figures from [5]

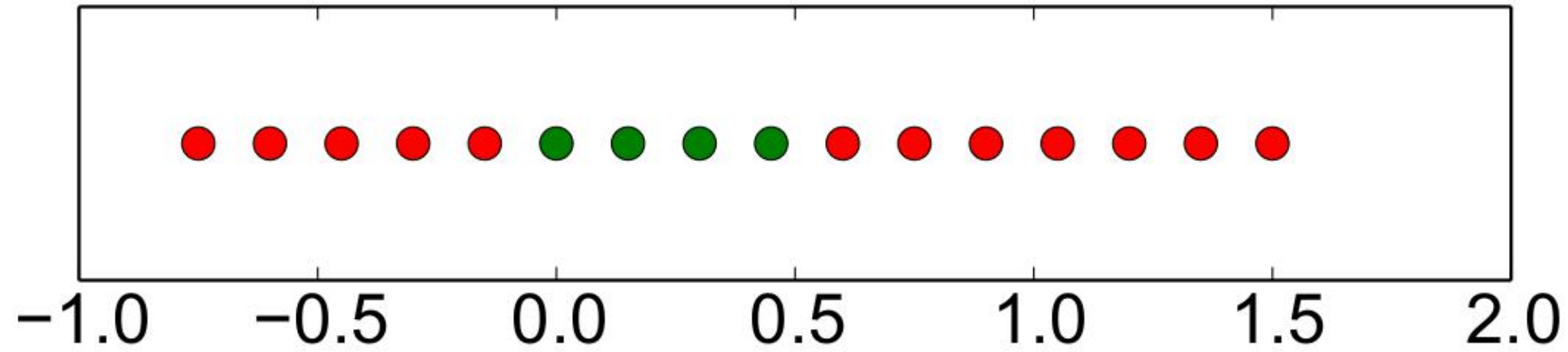


# Digit recognition - etalons means vs. perceptron



Figures from [5]

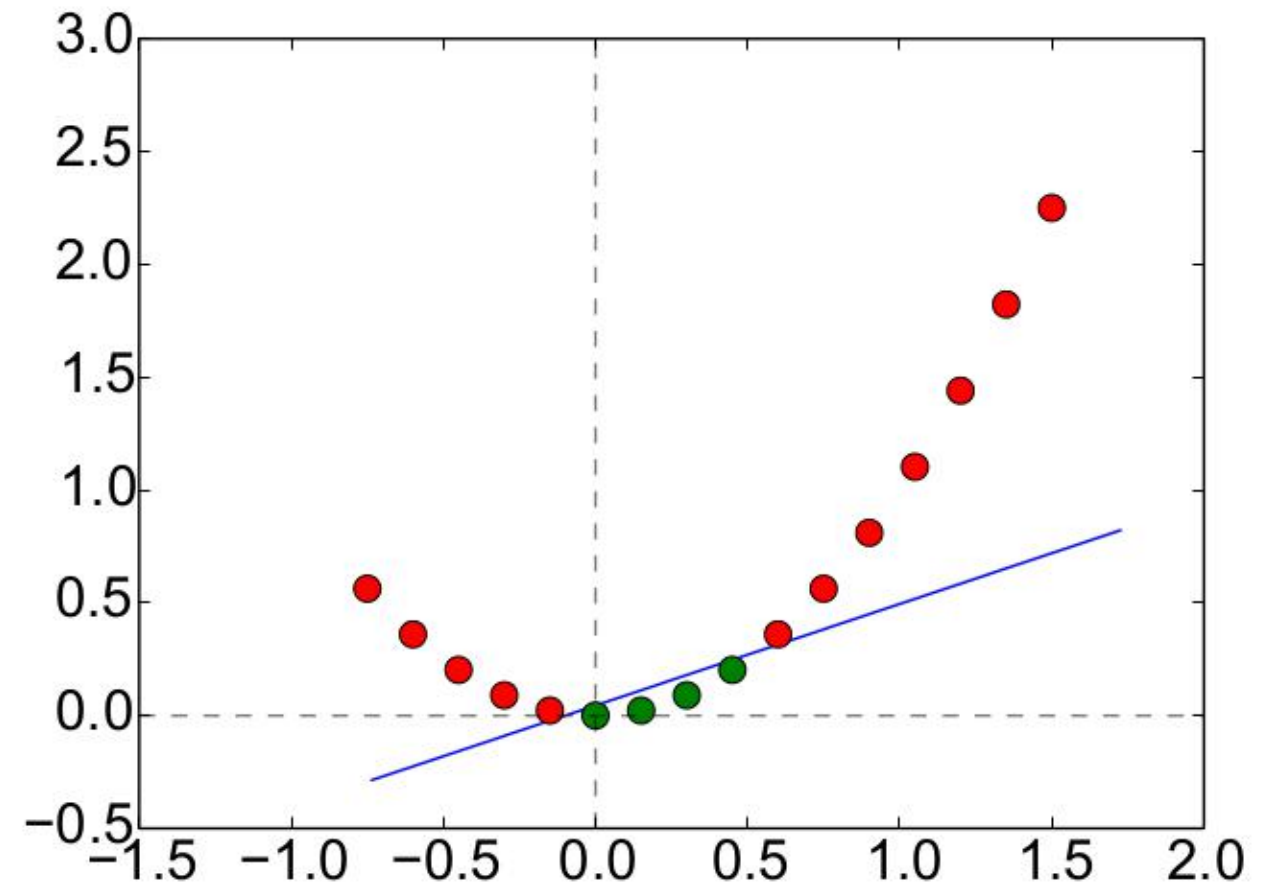
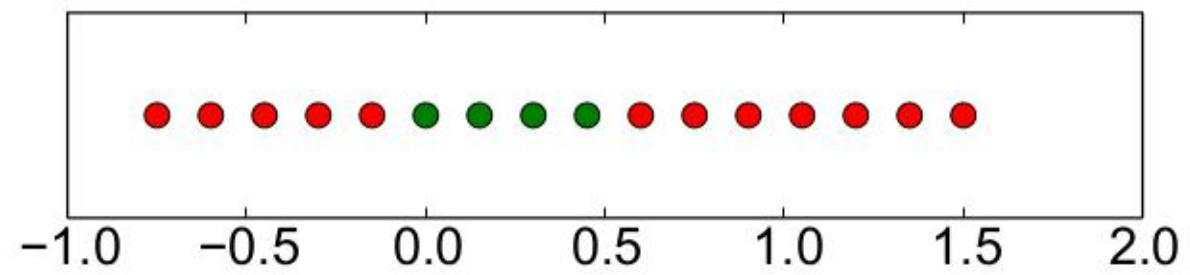
# What if not lin separable?



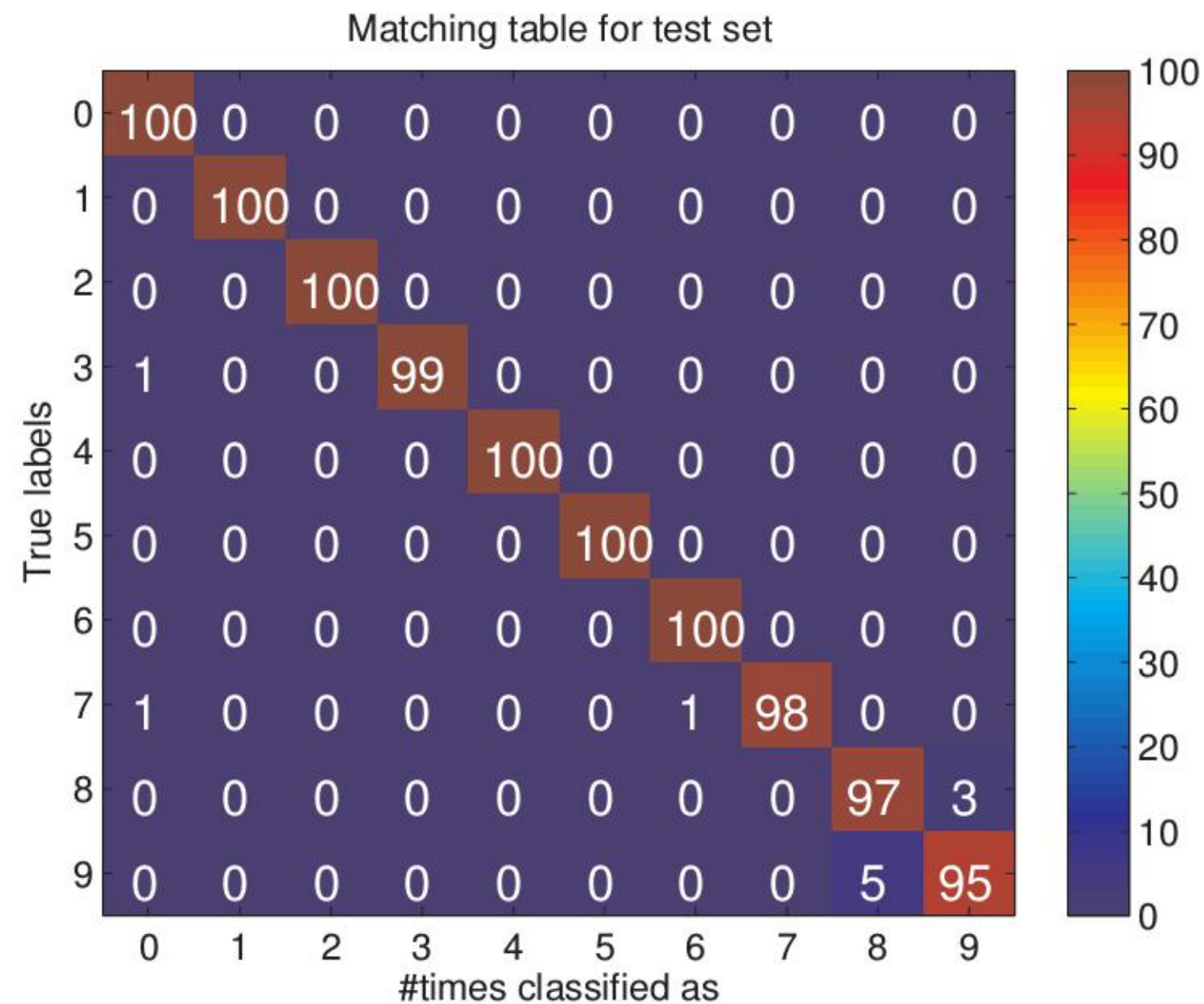
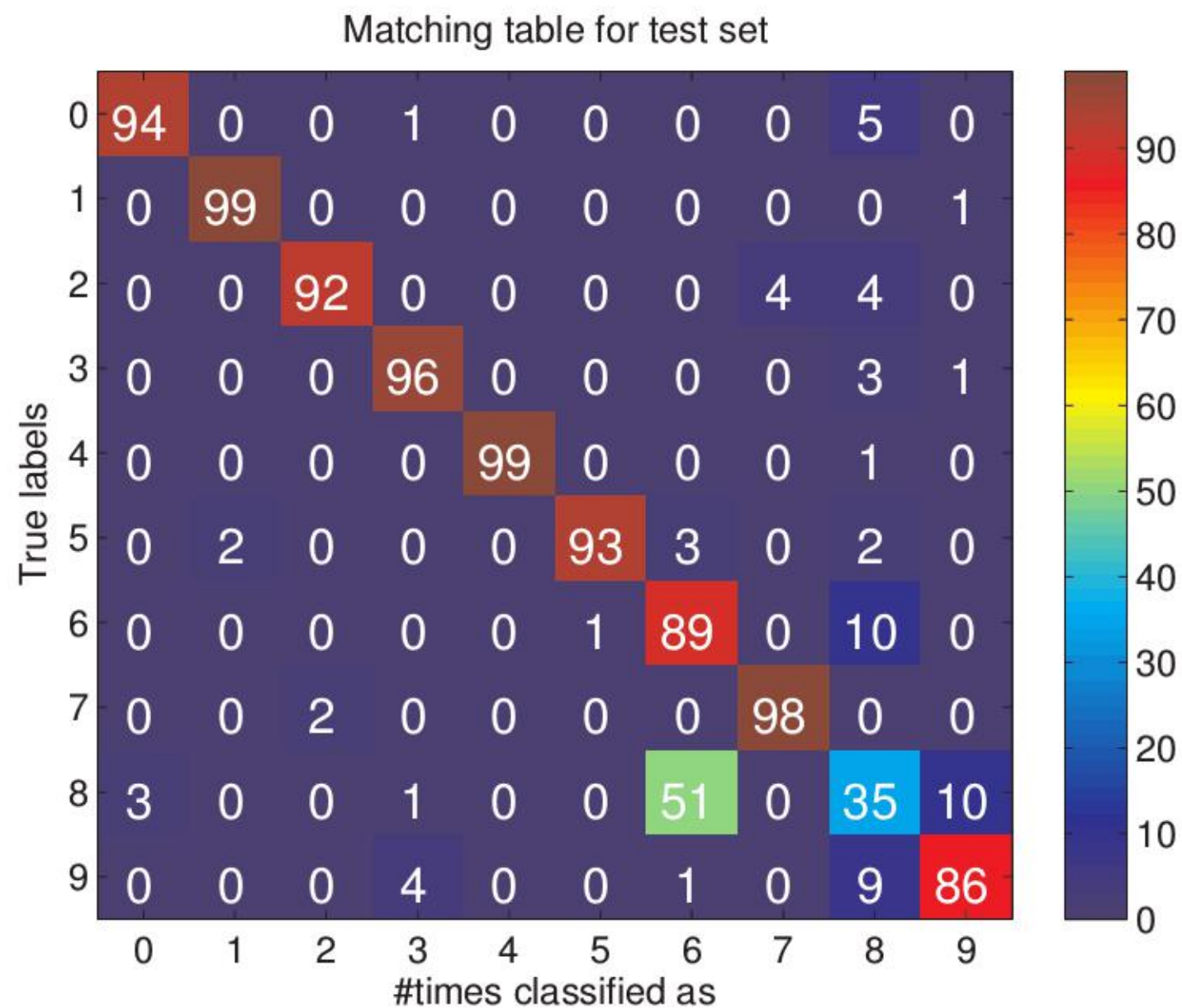
Dimension lifting

$$\mathbf{x} = [x, x^2]^T$$

Dimension lifting,  $\mathbf{x} = [x, x^2]^\top$



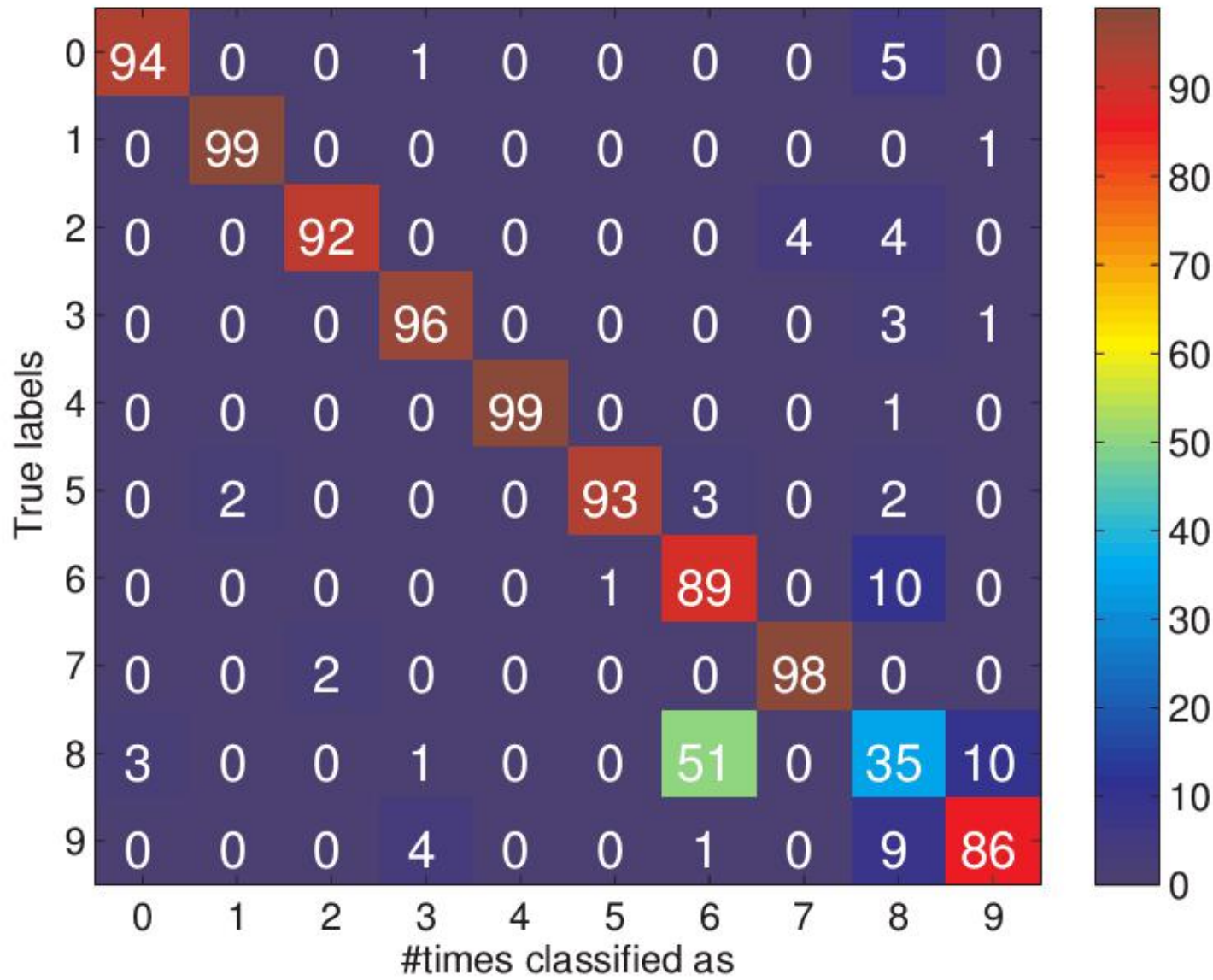
# Performance comparison, parameters fixed



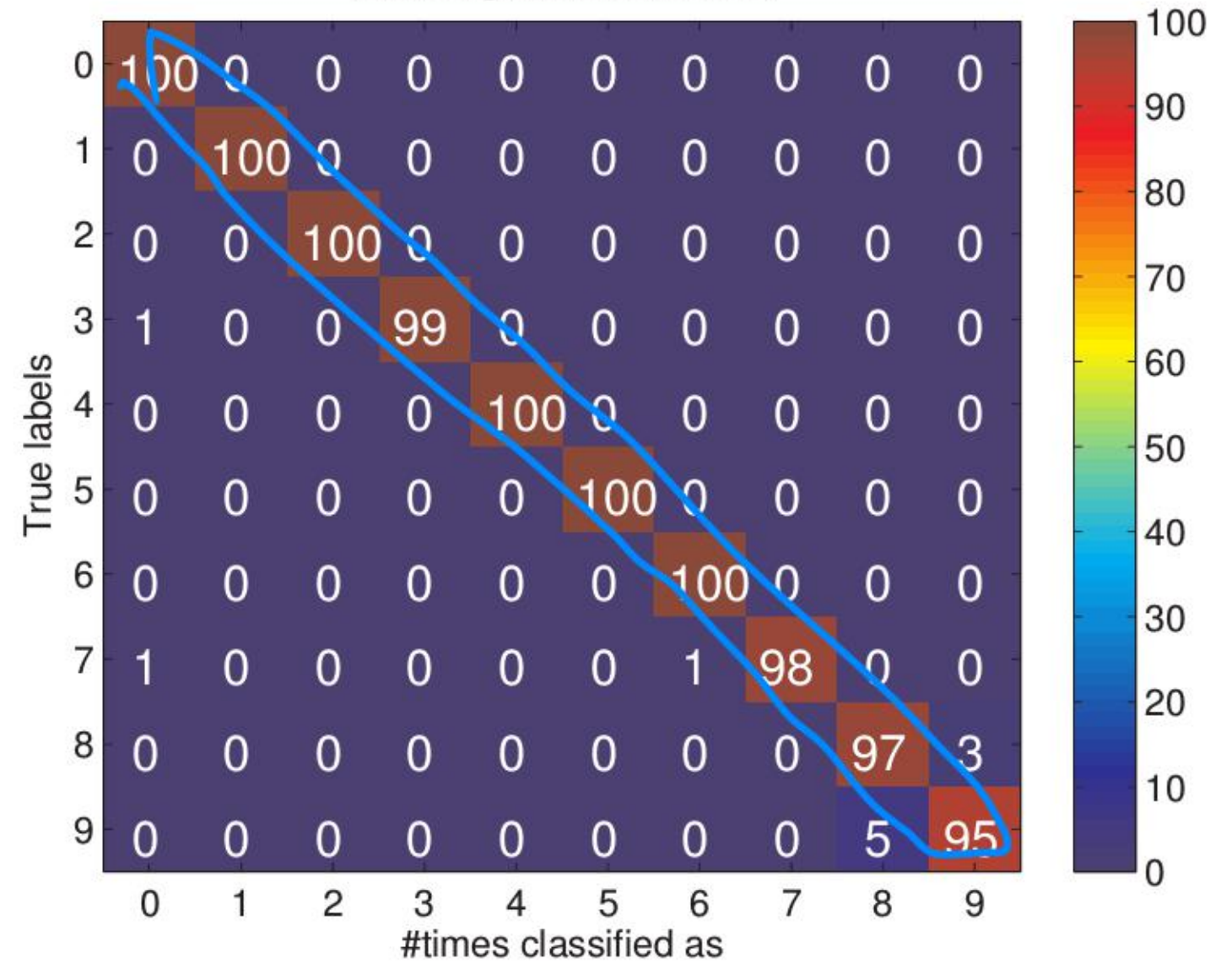


# Performance comparison, parameters fixed

Matching table for test set

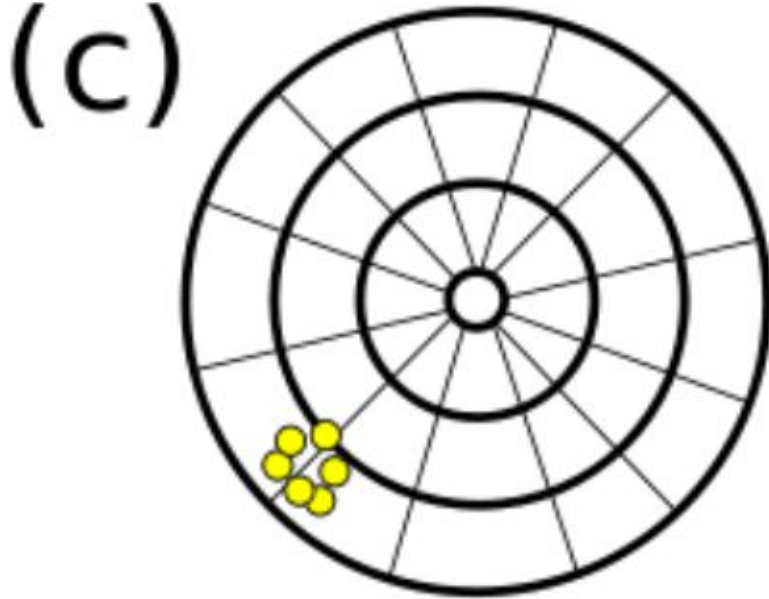
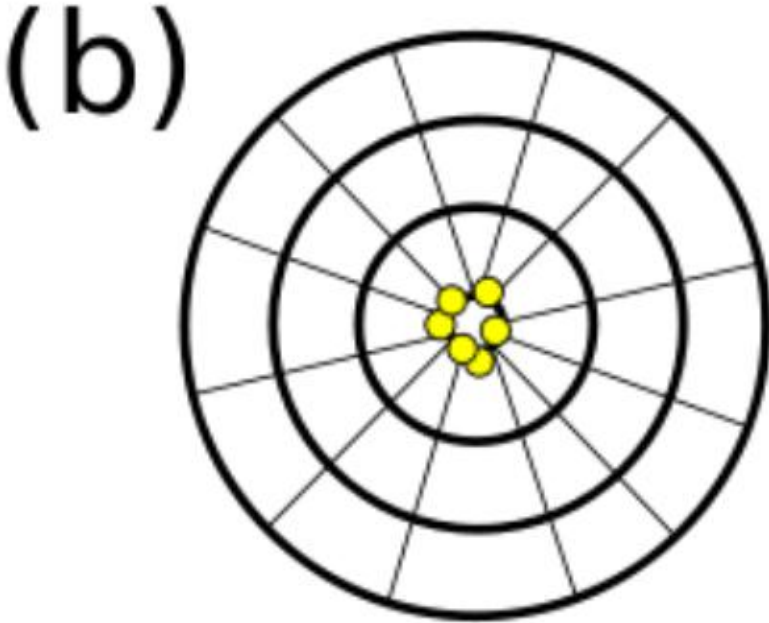
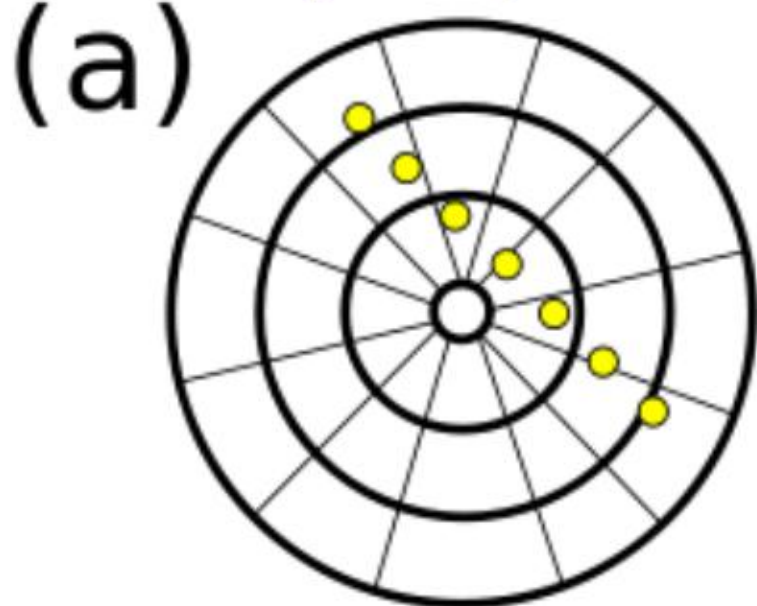


Matching table for test set

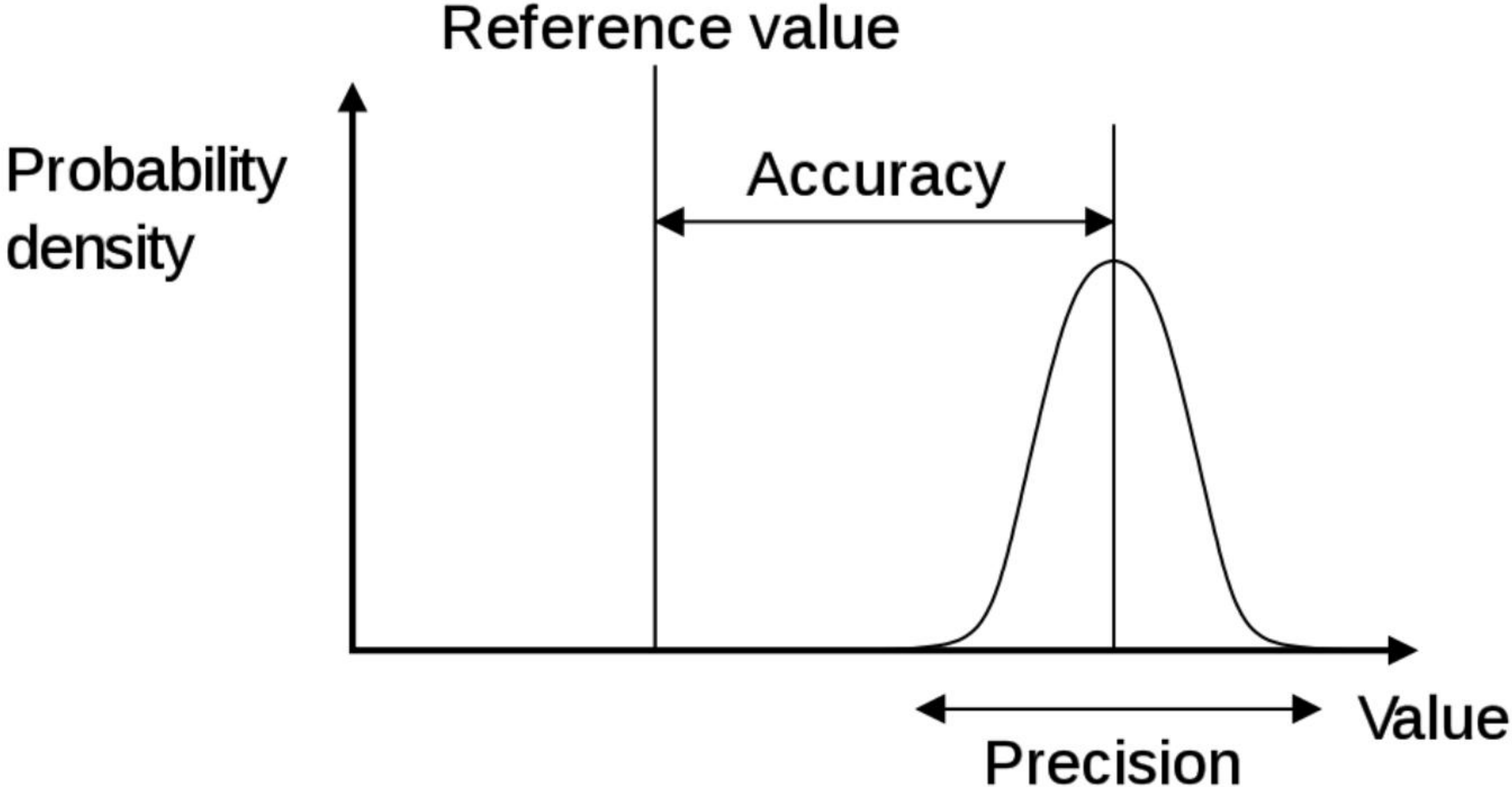




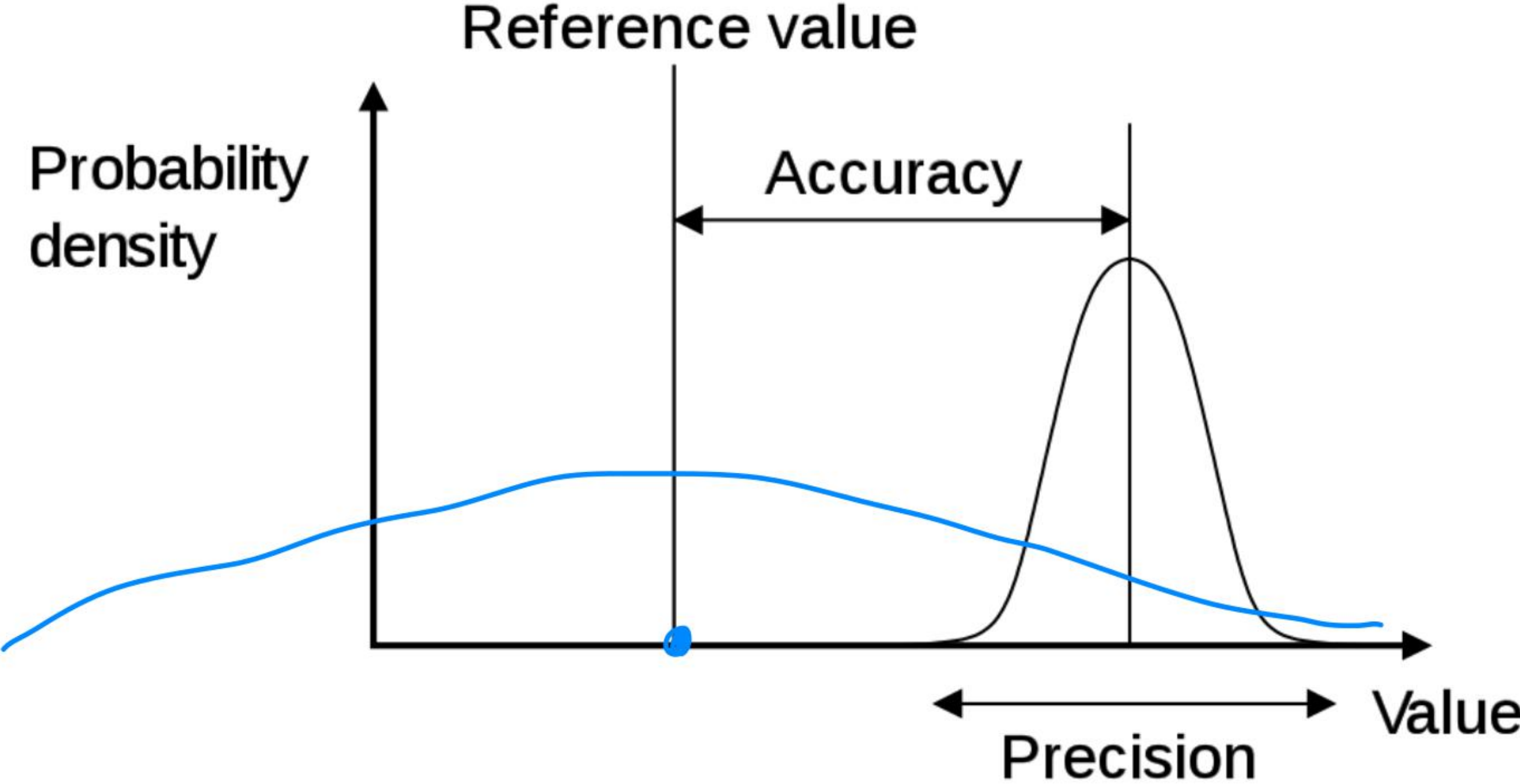
# Accuracy vs precision



# Accuracy vs precision



# Accuracy vs precision



# References I

Further reading: Chapter 18 of [4], or chapter 4 of [1], or chapter 5 of [2]. Many Matlab figures created with the help of [3]. You may also play with demo functions from [5].

[1] Christopher M. Bishop.

*Pattern Recognition and Machine Learning.*

Springer Science+Business Media, New York, NY, 2006.

PDF freely downloadable.

[2] Richard O. Duda, Peter E. Hart, and David G. Stork.

*Pattern Classification.*

John Wiley & Sons, 2nd edition, 2001.

[3] Votjěch Franc and Václav Hlaváč.

Statistical pattern recognition toolbox.

<http://cmp.felk.cvut.cz/cmp/software/stprtool/index.html>.



## References II

- [4] Stuart Russell and Peter Norvig.  
*Artificial Intelligence: A Modern Approach*.  
Prentice Hall, 3rd edition, 2010.  
<http://aima.cs.berkeley.edu/>.
  
- [5] Tomáš Svoboda, Jan Kybic, and Hlaváč Václav.  
*Image Processing, Analysis and Machine Vision — A MATLAB Companion*.  
Thomson, Toronto, Canada, 1<sup>st</sup> edition, September 2007.  
<http://visionbook.felk.cvut.cz/>.

# References I

Further reading: Chapter 18 of [4], or chapter 4 of [1], or chapter 5 of [2]. Many Matlab figures created with the help of [3]. You may also play with demo functions from [5].

[1] Christopher M. Bishop.

*Pattern Recognition and Machine Learning.*

Springer Science+Business Media, New York, NY, 2006.

PDF freely downloadable.

[2] Richard O. Duda, Peter E. Hart, and David G. Stork.

*Pattern Classification.*

John Wiley & Sons, 2nd edition, 2001.

[3] Votjěch Franc and Václav Hlaváč.

Statistical pattern recognition toolbox.

<http://cmp.felk.cvut.cz/cmp/software/stprtool/index.html>.