Support Vector Machines

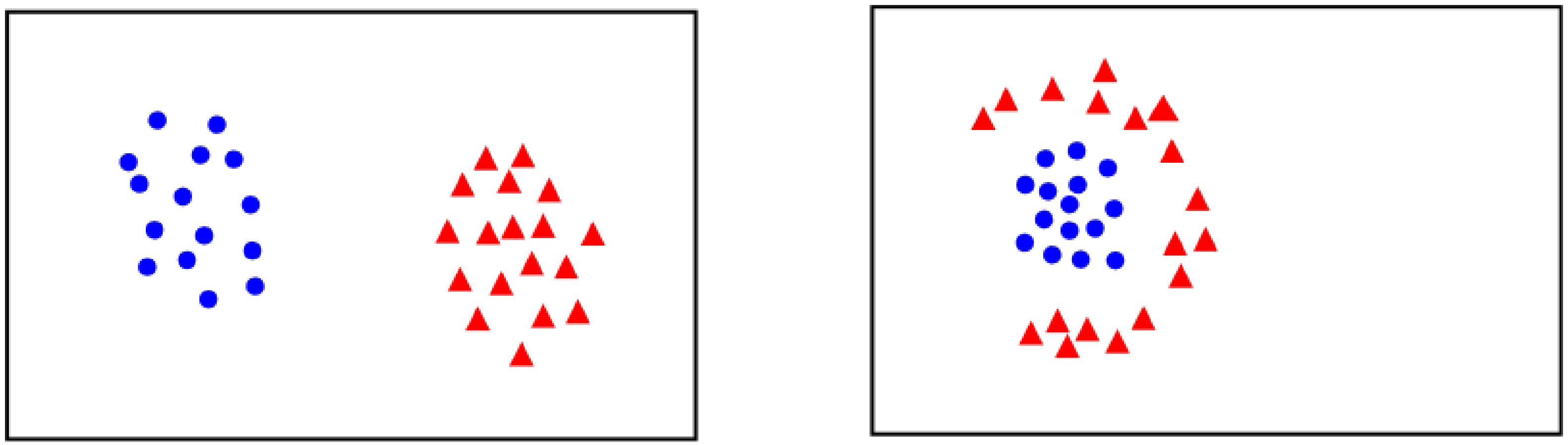
Slides are adapted from Andrew Moore (CMU), Andrew Zisserman (Oxford), Mingyue Tan (UBC)

Binary Classification

Given training data $(\mathbf{x}_i, y_i)$ for $i = 1 \ldots N$, with $\mathbf{x}_i \in \mathbb{R}^d$ and $y_i \in \{-1, 1\}$, learn a classifier $f(\mathbf{x})$ such that

$$f(\mathbf{x}_i) \begin{cases} \geq 0 & y_i = +1 \\ < 0 & y_i = -1 \end{cases}$$

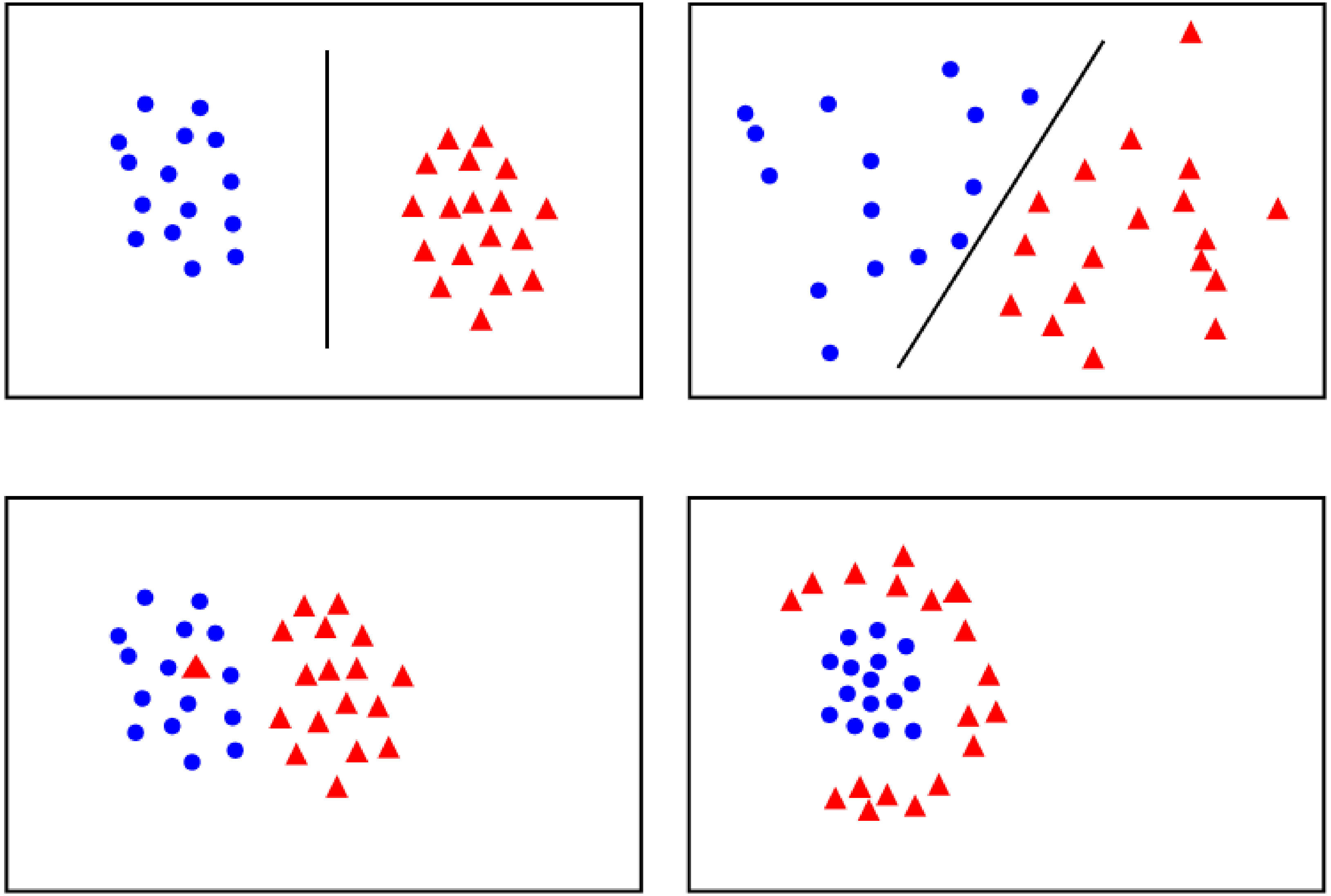
i.e. $y_i f(\mathbf{x}_i) > 0$ for a correct classification.



Linear separability

linearly separable

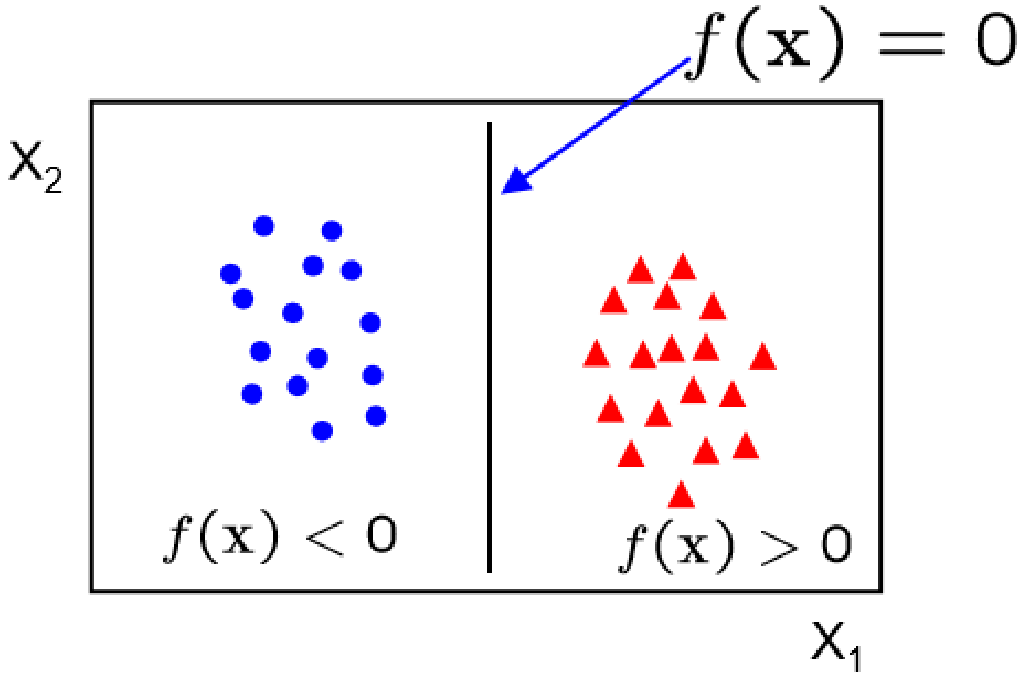
not linearly separable



Linear classifiers

A linear classifier has the form

$$f(\mathbf{x}) = \mathbf{w}^\top \mathbf{x} + b$$

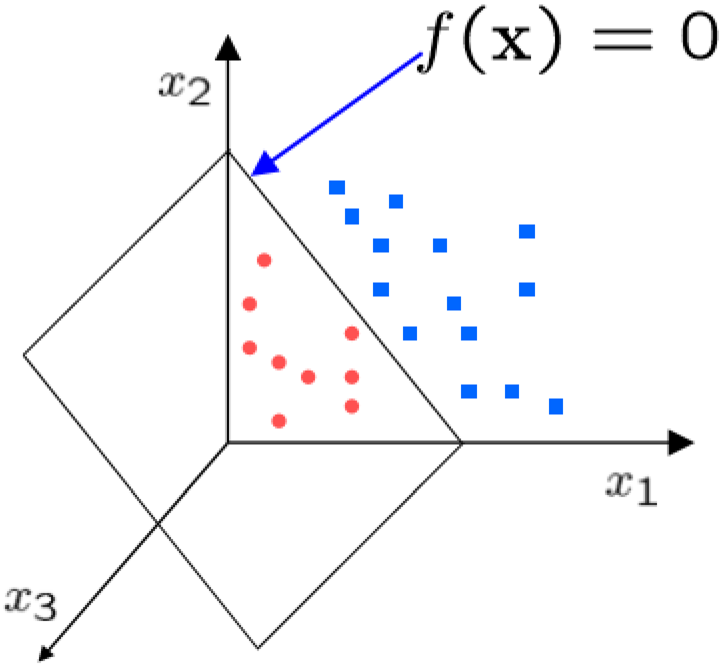


- in 2D the discriminant is a line
- $\mathbf{w}$ is the normal to the line, and b the bias
- $\mathbf{w}$ is known as the weight vector

Linear classifiers

A linear classifier has the form

$$f(\mathbf{x}) = \mathbf{w}^\top \mathbf{x} + b$$



- in 3D the discriminant is a plane, and in nD it is a hyperplane

For a K-NN classifier it was necessary to `carry' the training data

For a linear classifier, the training data is used to learn **w** and then discarded

Only **w** is needed for classifying new data

The Perceptron Classifier

Given linearly separable data $\mathbf{x}_i$ labelled into two categories $y_i = \{-1,1\}$, find a weight vector $\mathbf{w}$ such that the discriminant function

$$f(\mathbf{x}_i) = \mathbf{w}^\top \mathbf{x}_i + b$$

separates the categories for i = 1, .., N

- how can we find this separating hyperplane ?

The Perceptron Algorithm

Write classifier as $f(\mathbf{x}_i) = \tilde{\mathbf{w}}^\top \tilde{\mathbf{x}}_i + w_0 = \mathbf{w}^\top \mathbf{x}_i$

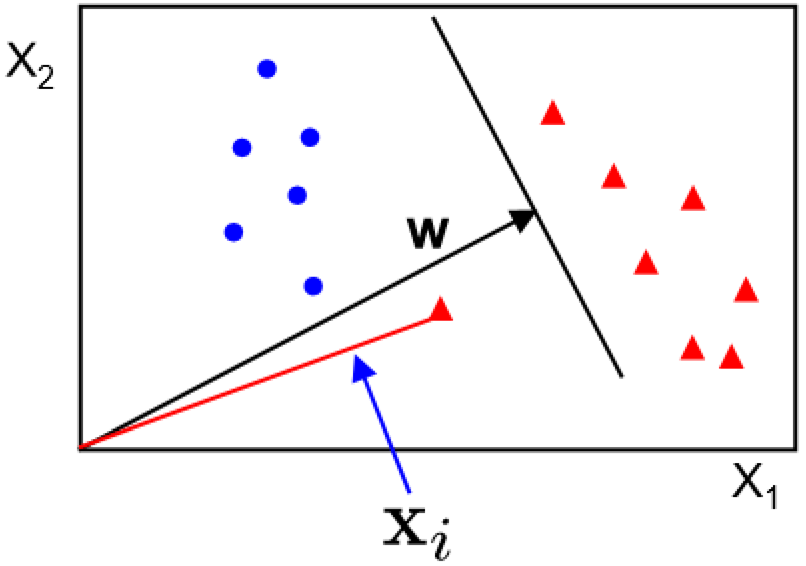
$$\text{where } \mathbf{w} = (\tilde{\mathbf{w}}, w_0), \mathbf{x}_i = (\tilde{\mathbf{x}}_i, 1)$$

- Initialize $\mathbf{w}$ = 0
- Cycle though the data points { $\mathbf{x}_i$, y_i }
  - if $\mathbf{x}_i$ is misclassified then $\mathbf{w} \leftarrow \mathbf{w} + \alpha\,\mathrm{sign}(f(\mathbf{x}_i))\,\mathbf{x}_i$
- Until all the data is correctly classified

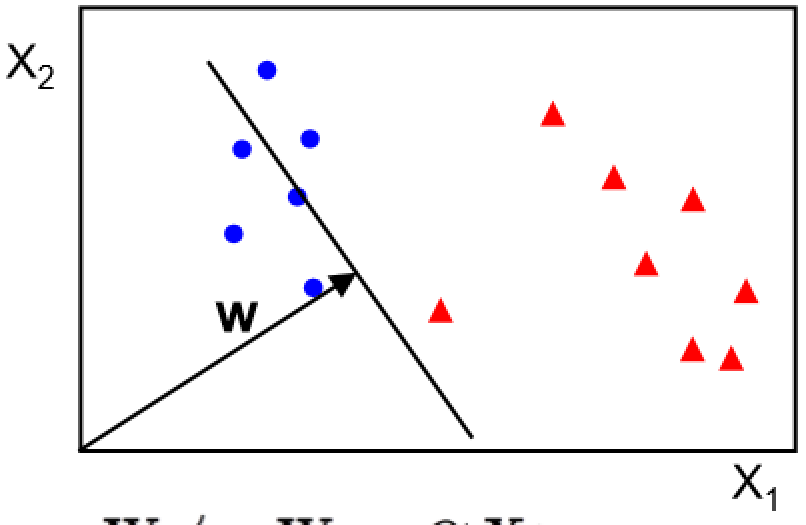
For example in 2D

- Initialize **w** = 0
- Cycle though the data points { $\mathbf{x}_i$, y_i }
  - if $\mathbf{x}_i$ is misclassified then $\mathbf{w} \leftarrow \mathbf{w} + \alpha \, \mathrm{sign}(f(\mathbf{x}_i)) \, \mathbf{x}_i$
- Until all the data is correctly classified

before update



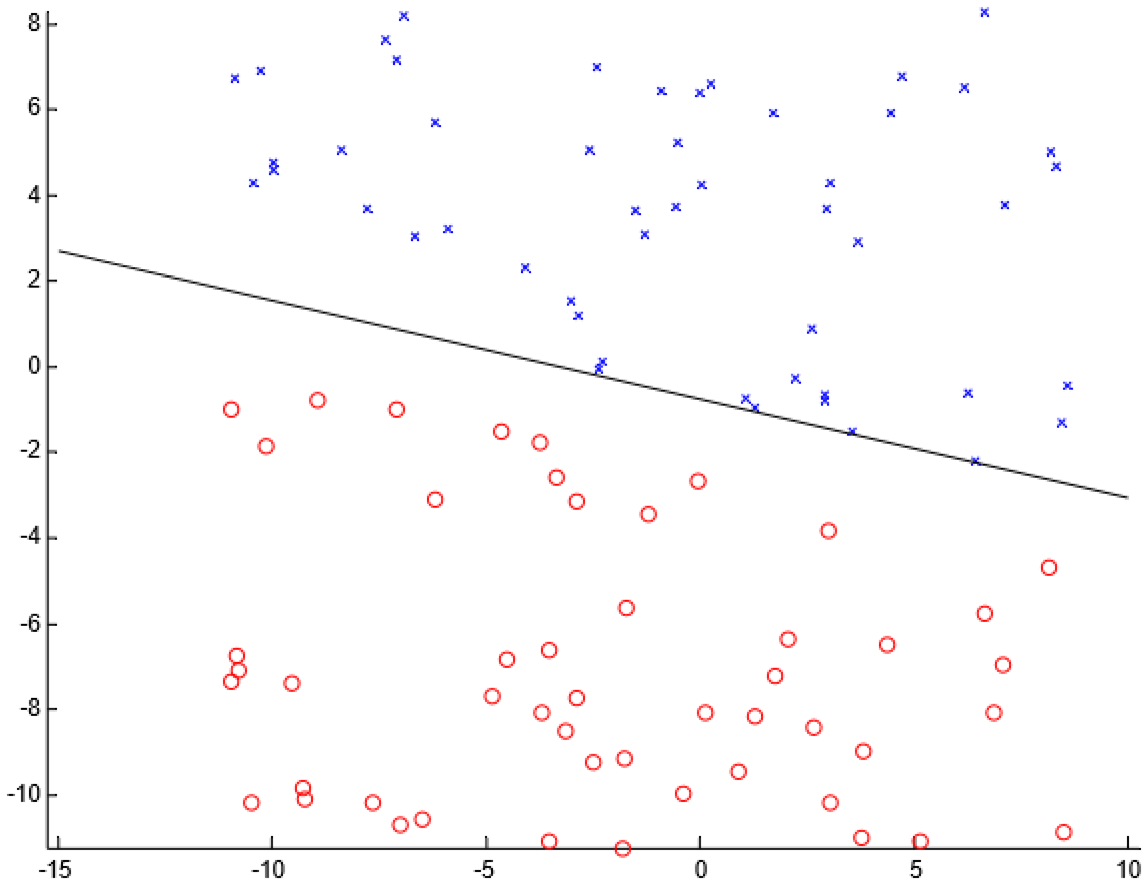
after update



$\mathbf{w} \leftarrow \mathbf{w} - \alpha \, \mathbf{x}_i$

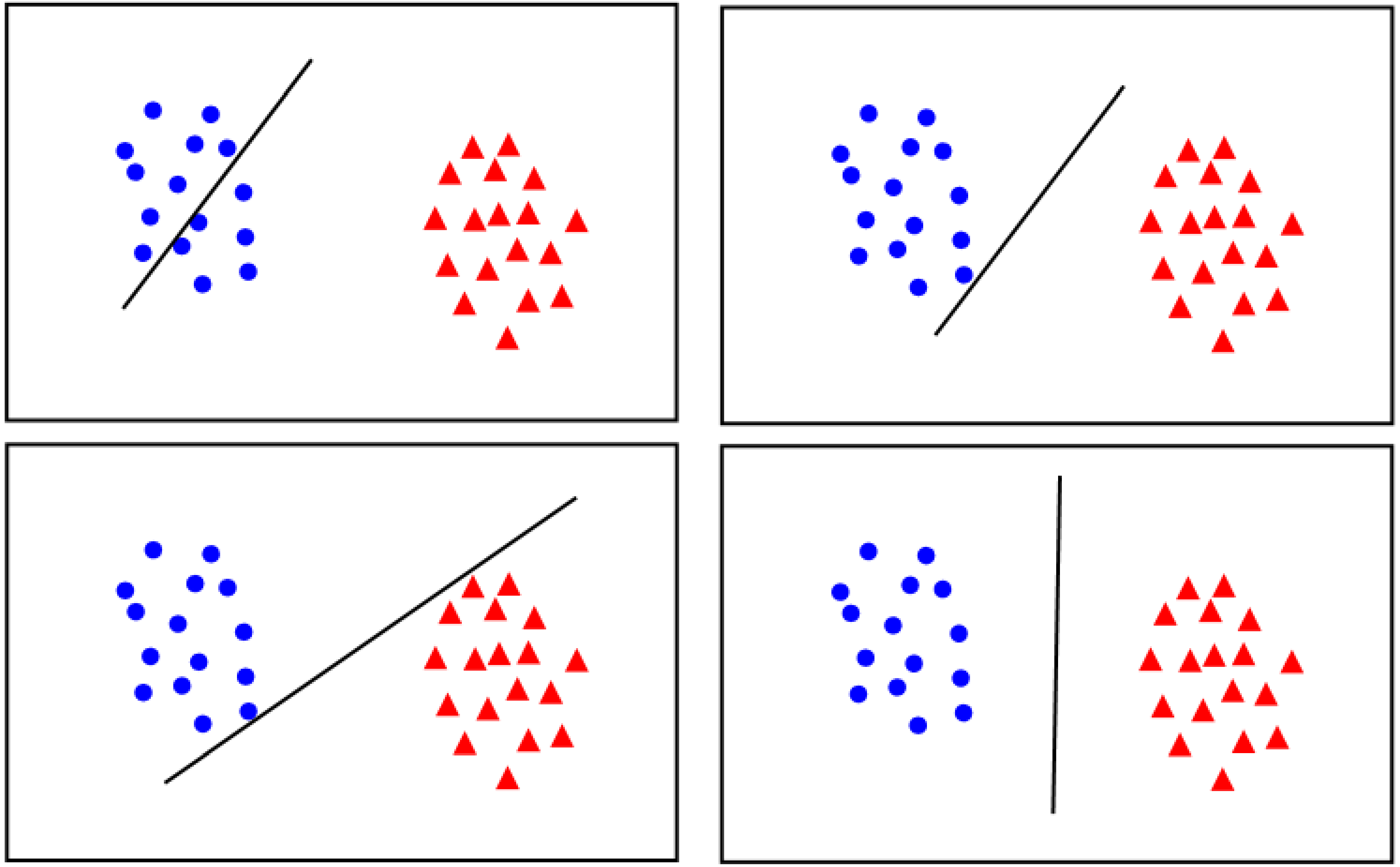
NB after convergence $\mathbf{w} = \sum_i^N \alpha_i \mathbf{x}_i$

Perceptron example



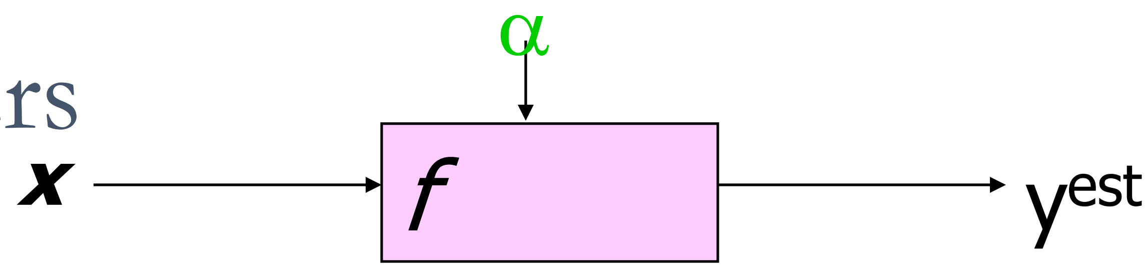
- if the data is linearly separable, then the algorithm will converge
- convergence can be slow …
- separating line close to training data
- we would prefer a larger margin for generalization

What is the best w?



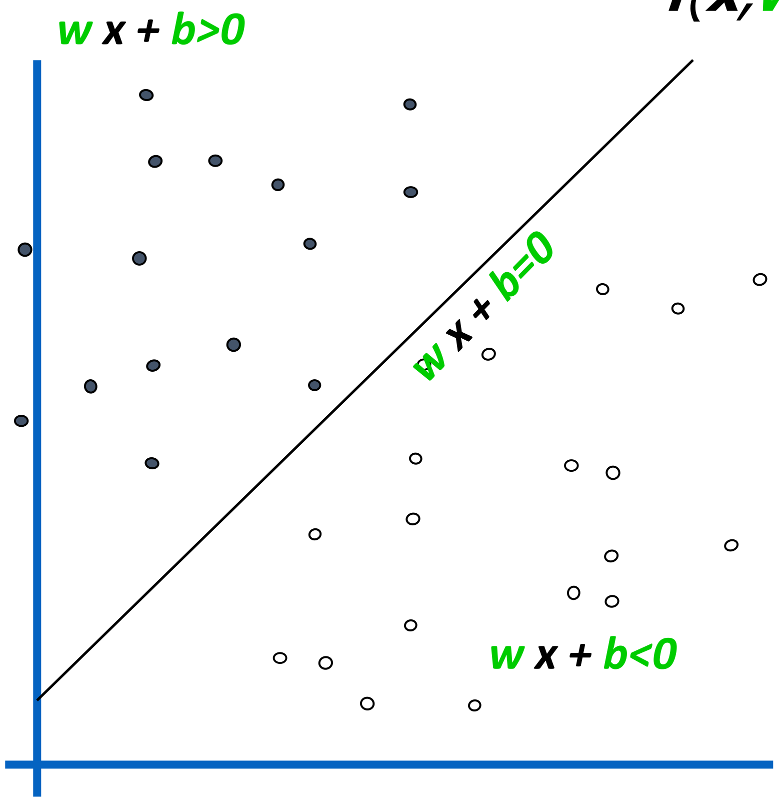
- maximum margin solution: most stable under perturbations of the inputs

Linear Classifiers



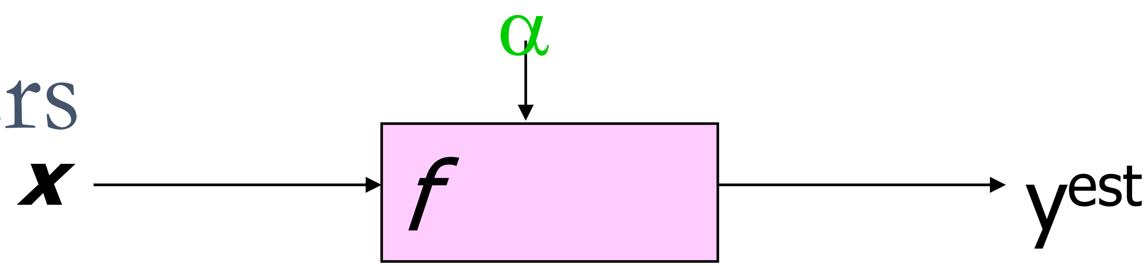
$$f(\mathbf{x}, \mathbf{w}, b) = sign(\mathbf{w}\,\mathbf{x} + b)$$

- denotes +1
- ○ denotes -1

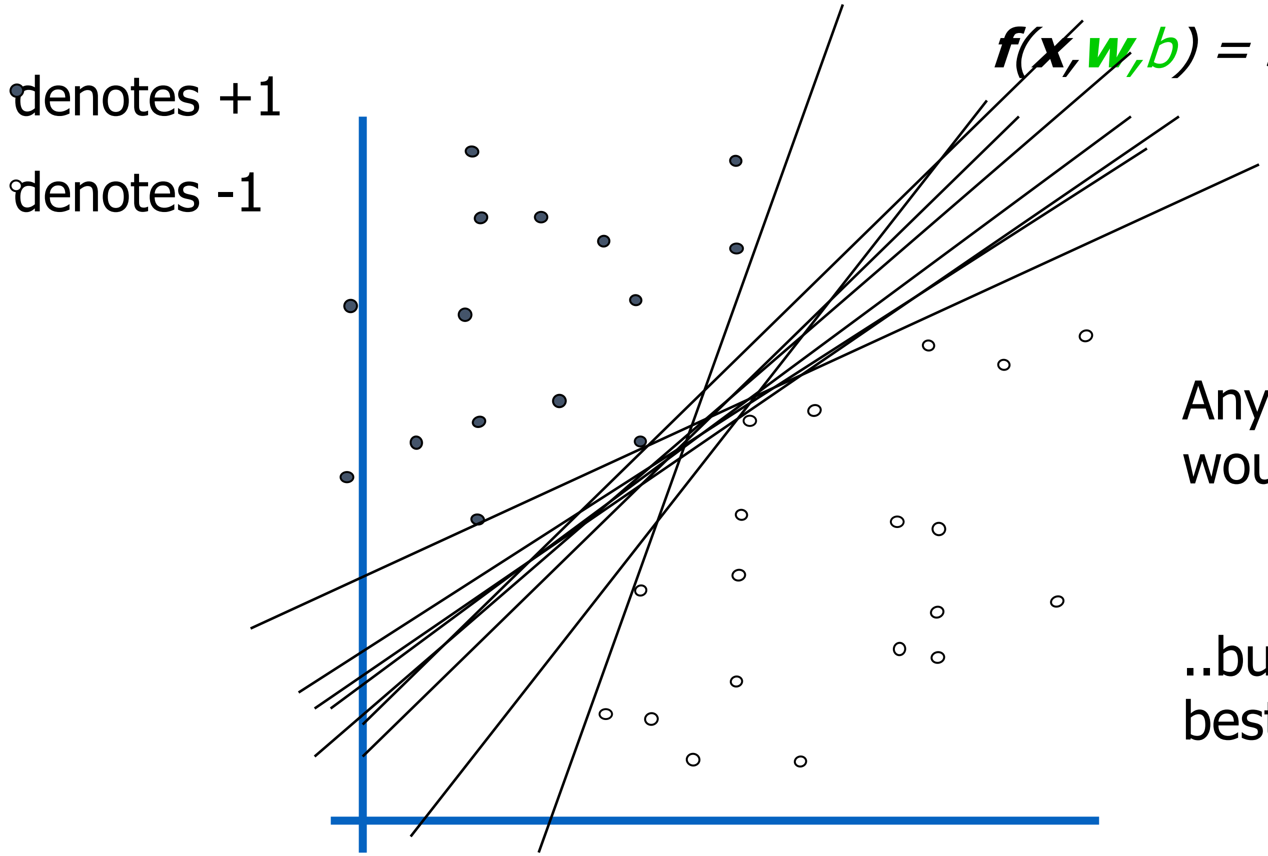


How would you classify this data?

Linear Classifiers



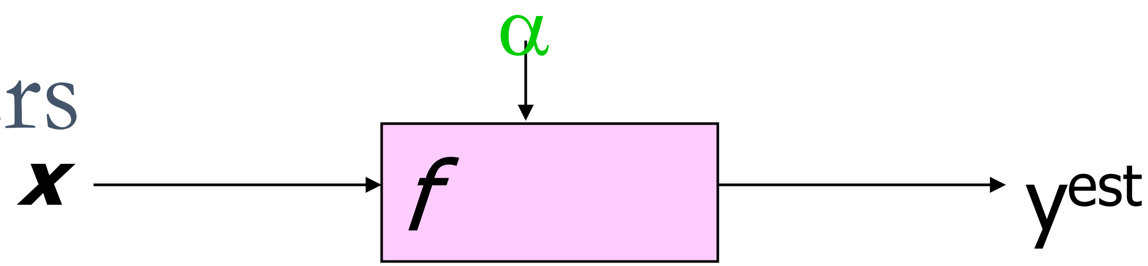
$f(\boldsymbol{x},\boldsymbol{w},b) = sign(\boldsymbol{w}\ \boldsymbol{x} + b)$



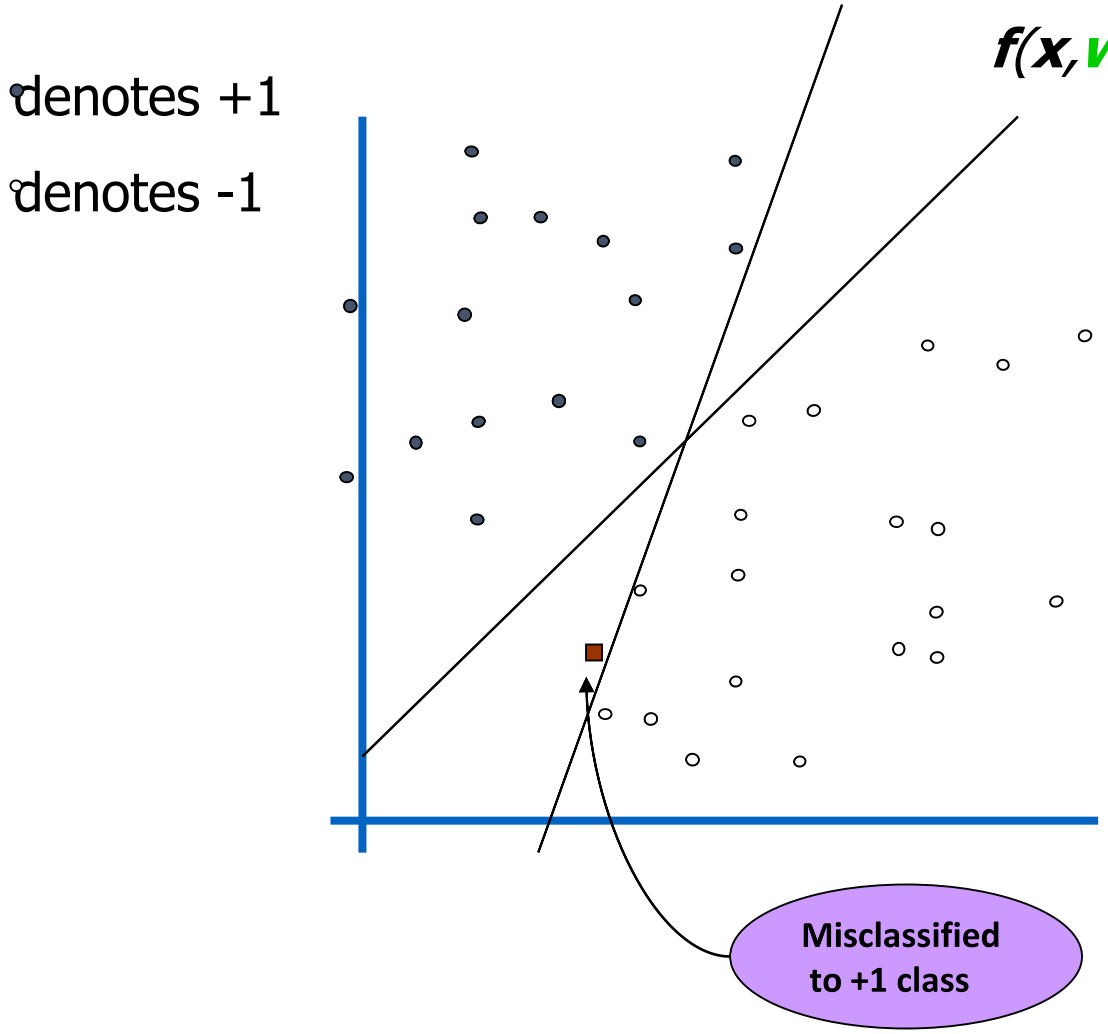
Any of these would be fine..

..but which is best?

Linear Classifiers

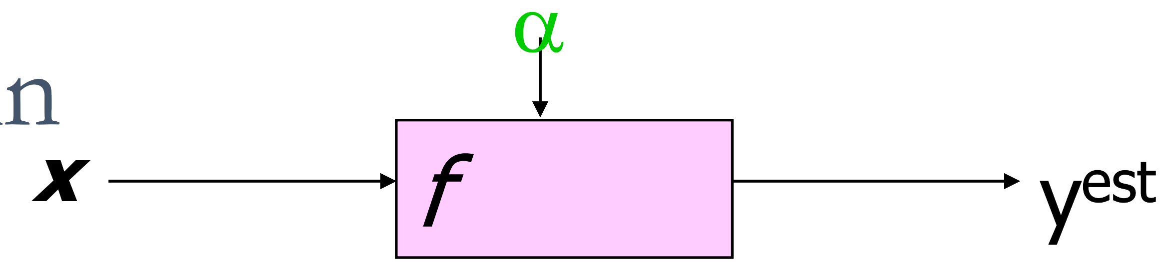


$$f(\mathbf{x},\mathbf{w},b) = sign(\mathbf{w}\ \mathbf{x} + b)$$

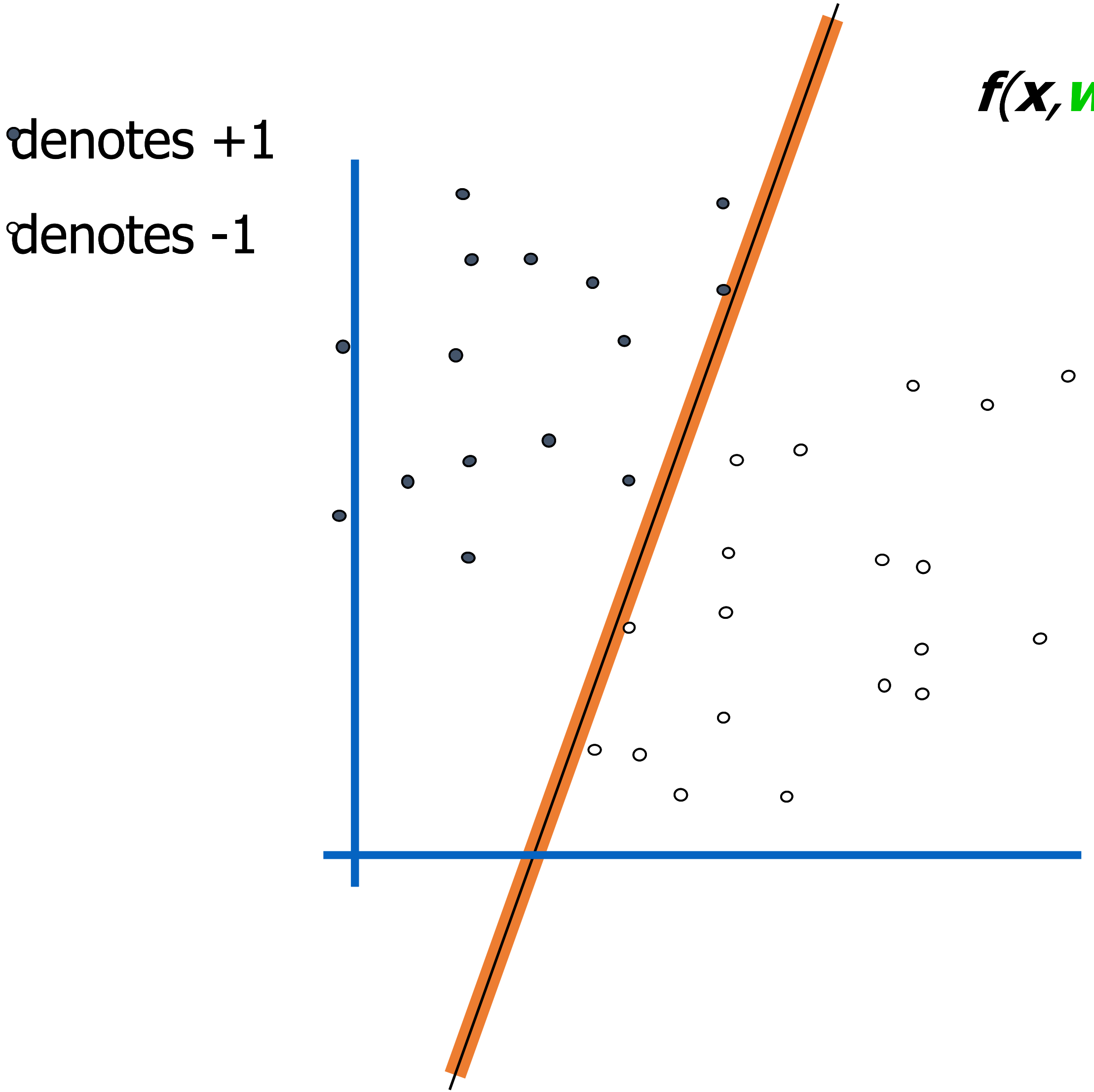


How would you classify this data?

Classifier Margin

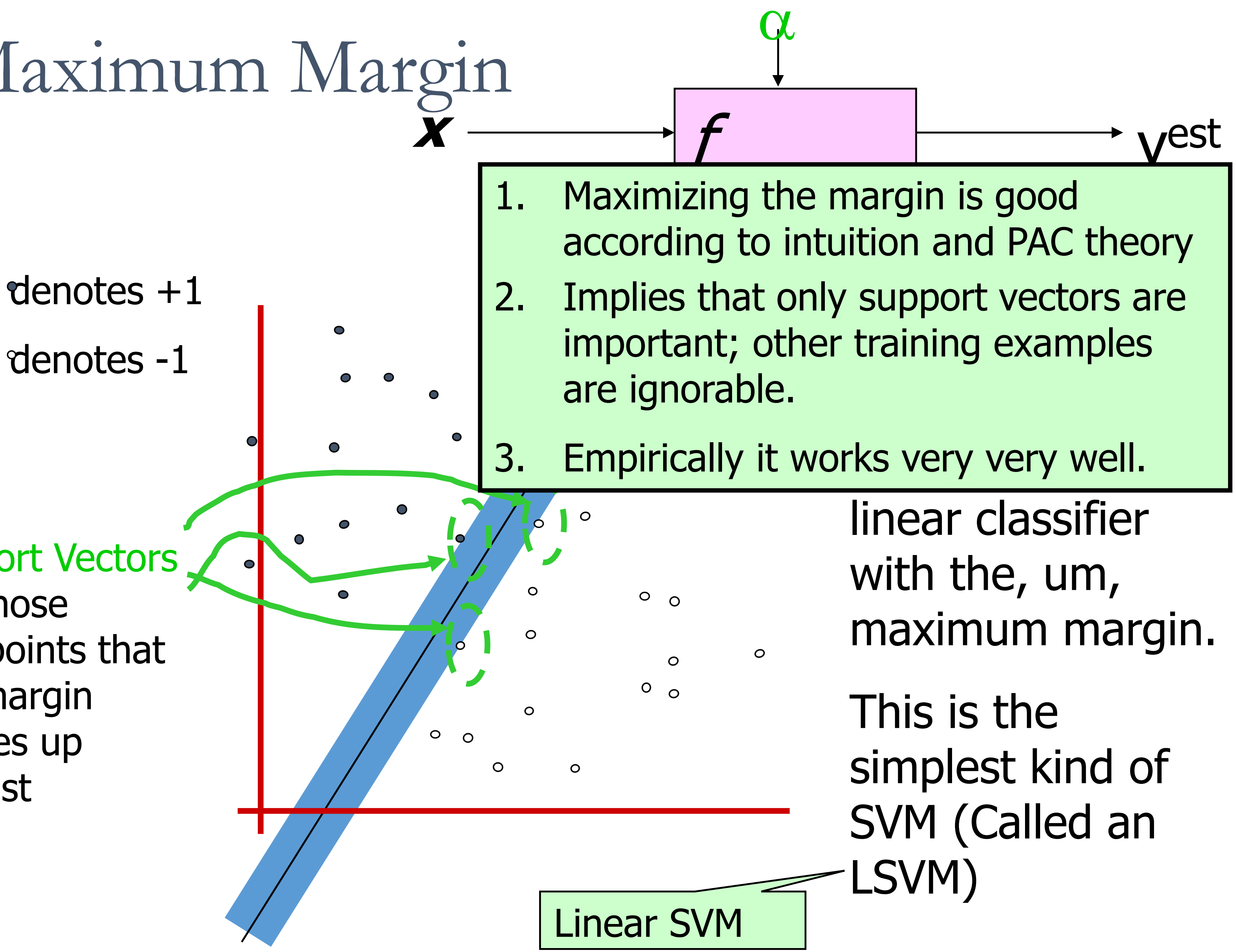


$f(\mathbf{x},\mathbf{w},b) = sign(\mathbf{w}\,\mathbf{x} + b)$



Define the margin of a linear classifier as the width that the boundary could be increased by before hitting a datapoint.

Maximum Margin



1. Maximizing the margin is good according to intuition and PAC theory
2. Implies that only support vectors are important; other training examples are ignorable.
3. Empirically it works very very well.

Support Vectors are those datapoints that the margin pushes up against

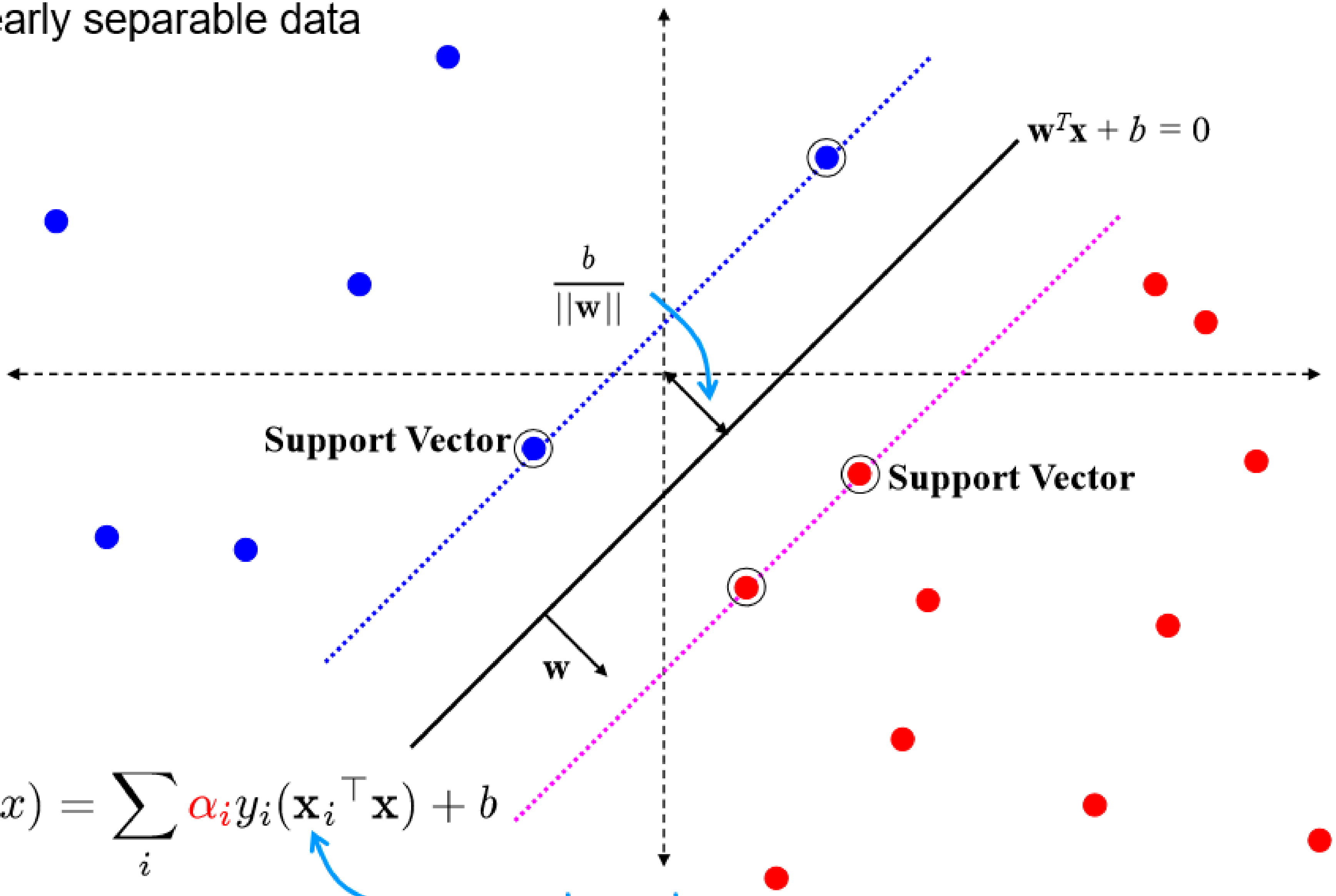
linear classifier with the, um, maximum margin.

This is the simplest kind of SVM (Called an LSVM)

Linear SVM

Support Vector Machine

linearly separable data



$$f(x) = \sum_i \alpha_i y_i (\mathbf{x}_i^\top \mathbf{x}) + b$$

support vectors

SVM – sketch derivation

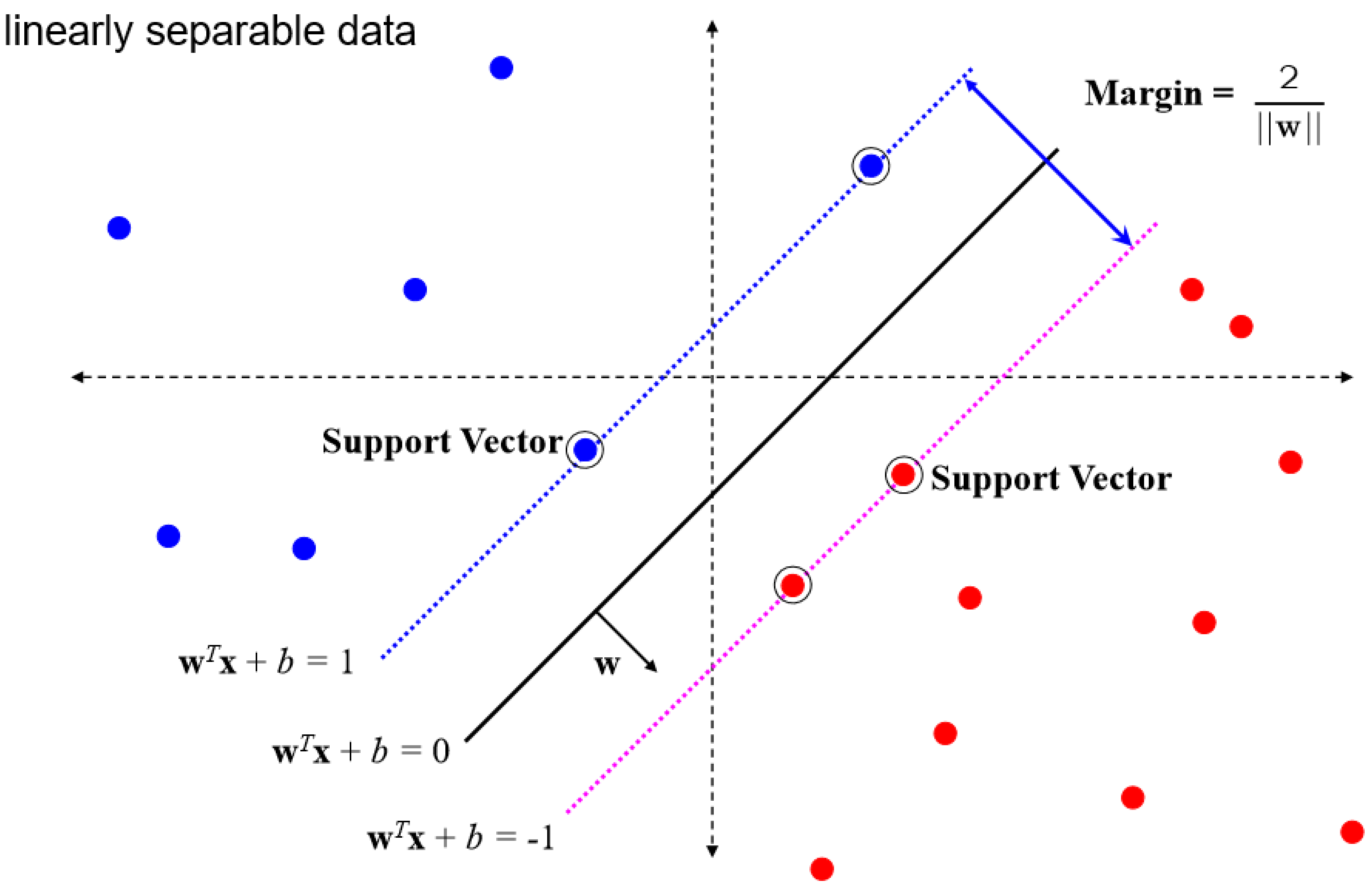
- Since $\mathbf{w}^\top \mathbf{x} + b = 0$ and $c(\mathbf{w}^\top \mathbf{x} + b) = 0$ define the same plane, we have the freedom to choose the normalization of $\mathbf{w}$

- Choose normalization such that $\mathbf{w}^\top \mathbf{x}_+ + b = +1$ and $\mathbf{w}^\top \mathbf{x}_- + b = -1$ for the positive and negative support vectors respectively

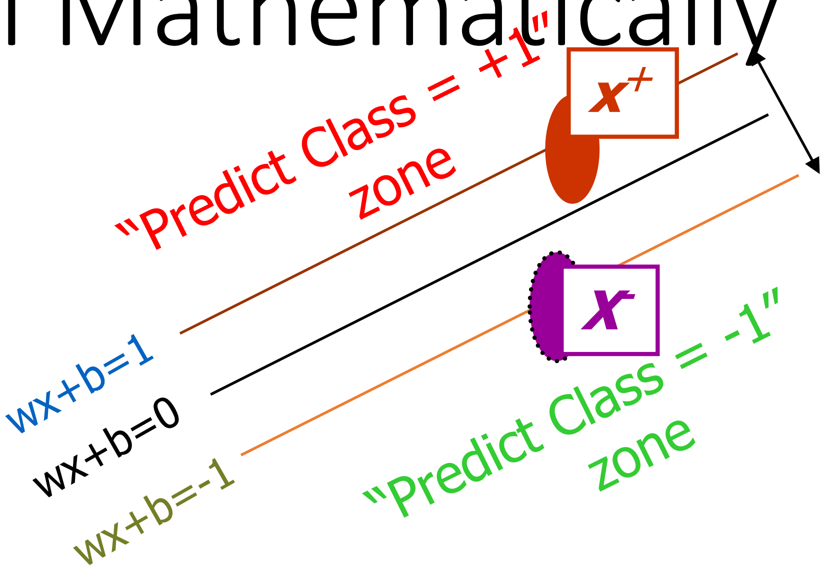
- Then the margin is given by

$$\frac{\mathbf{w}}{||\mathbf{w}||} \cdot \left(\mathbf{x}_+ - \mathbf{x}_-\right) = \frac{\mathbf{w}^\top \left(\mathbf{x}_+ - \mathbf{x}_-\right)}{||\mathbf{w}||} = \frac{2}{||\mathbf{w}||}$$

Support Vector Machine



Linear SVM Mathematically



M=Margin Width

What we know:

- $\mathbf{w} \cdot \mathbf{x}^+ + b = +1$
- $\mathbf{w} \cdot \mathbf{x}^- + b = -1$
- $\mathbf{w} \cdot (\mathbf{x}^+ - \mathbf{x}^-) = 2$

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

Linear SVM Mathematically

- Goal: **1) Correctly classify all training data**

$$wx_i + b \geq 1 \quad \text{if } y_i = +1$$
$$wx_i + b \leq 1 \quad \text{if } y_i = -1$$
$$y_i(wx_i + b) \geq 1 \quad \text{for all i}$$

2) Maximize the Margin $M = \frac{2}{|w|}$

same as minimize $\frac{1}{2} w^t w$

- **We can formulate a Quadratic Optimization Problem and solve for w and b**

- Minimize $\Phi(w) = \frac{1}{2} w^t w$

  subject to $y_i(wx_i + b) \geq 1 \qquad \forall i$

SVM – Optimization

- Learning the SVM can be formulated as an optimization:

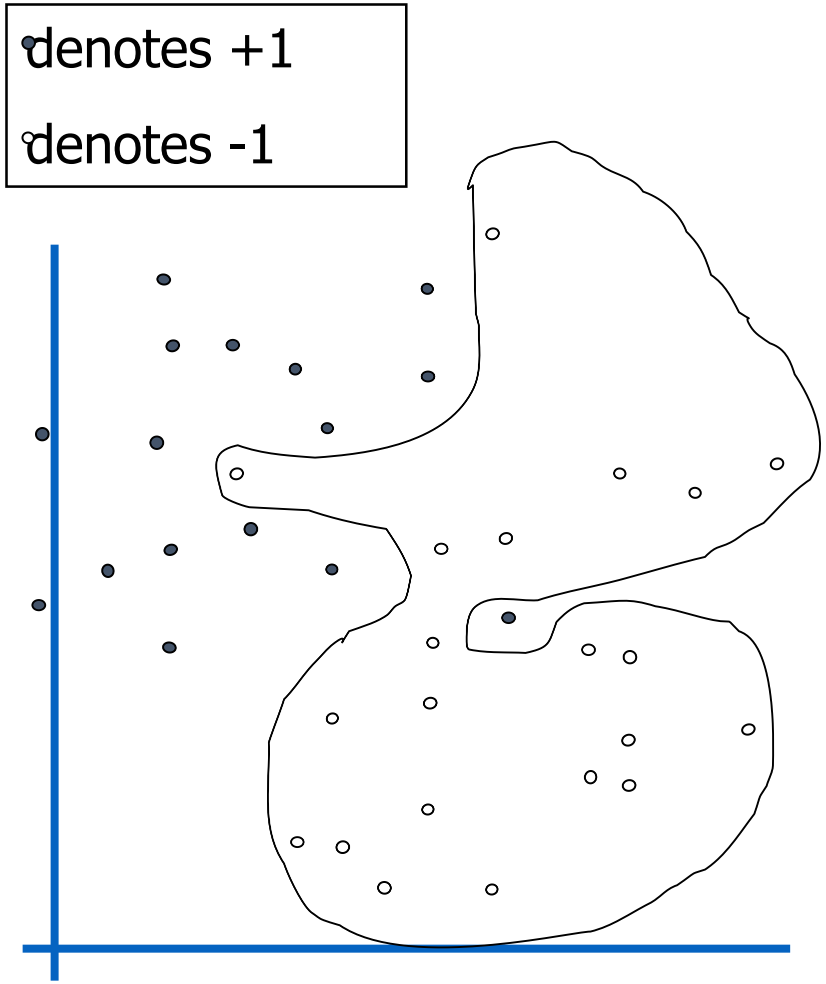
$$\max_{\mathbf{w}} \frac{2}{||\mathbf{w}||} \text{ subject to } \mathbf{w}^\top \mathbf{x}_i + b \begin{array}{ll} \geq 1 & \text{if } y_i = +1 \\ \leq -1 & \text{if } y_i = -1 \end{array} \text{ for } i = 1 \dots N$$

- Or equivalently

$$\min_{\mathbf{w}} ||\mathbf{w}||^2 \text{ subject to } y_i \left(\mathbf{w}^\top \mathbf{x}_i + b \right) \geq 1 \text{ for } i = 1 \dots N$$

- This is a quadratic optimization problem subject to linear constraints and there is a unique minimum

Dataset with noise



- **Hard Margin:** **So far we require all data points be classified correctly**

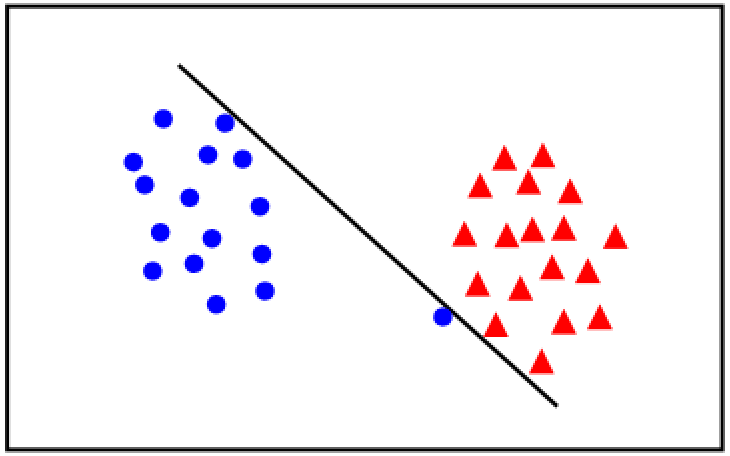
  - No training error

- **What if the training set is noisy?**

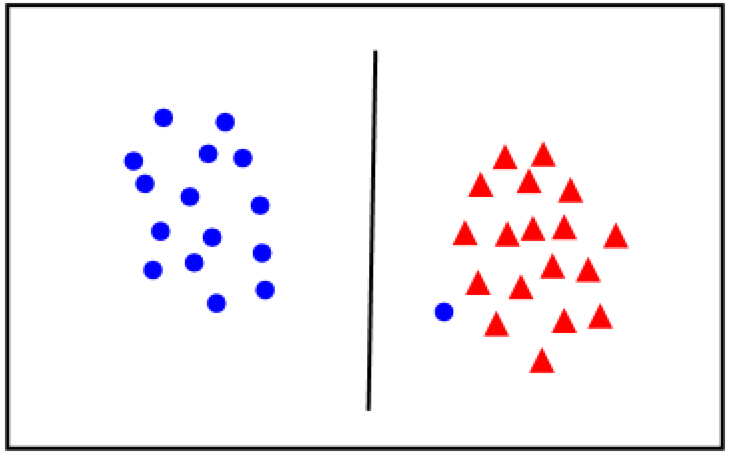
  - Solution 1: **use very powerful kernels**

OVERFITTING!

Linear separability again: What is the best w?



• the points can be linearly separated but there is a very narrow margin



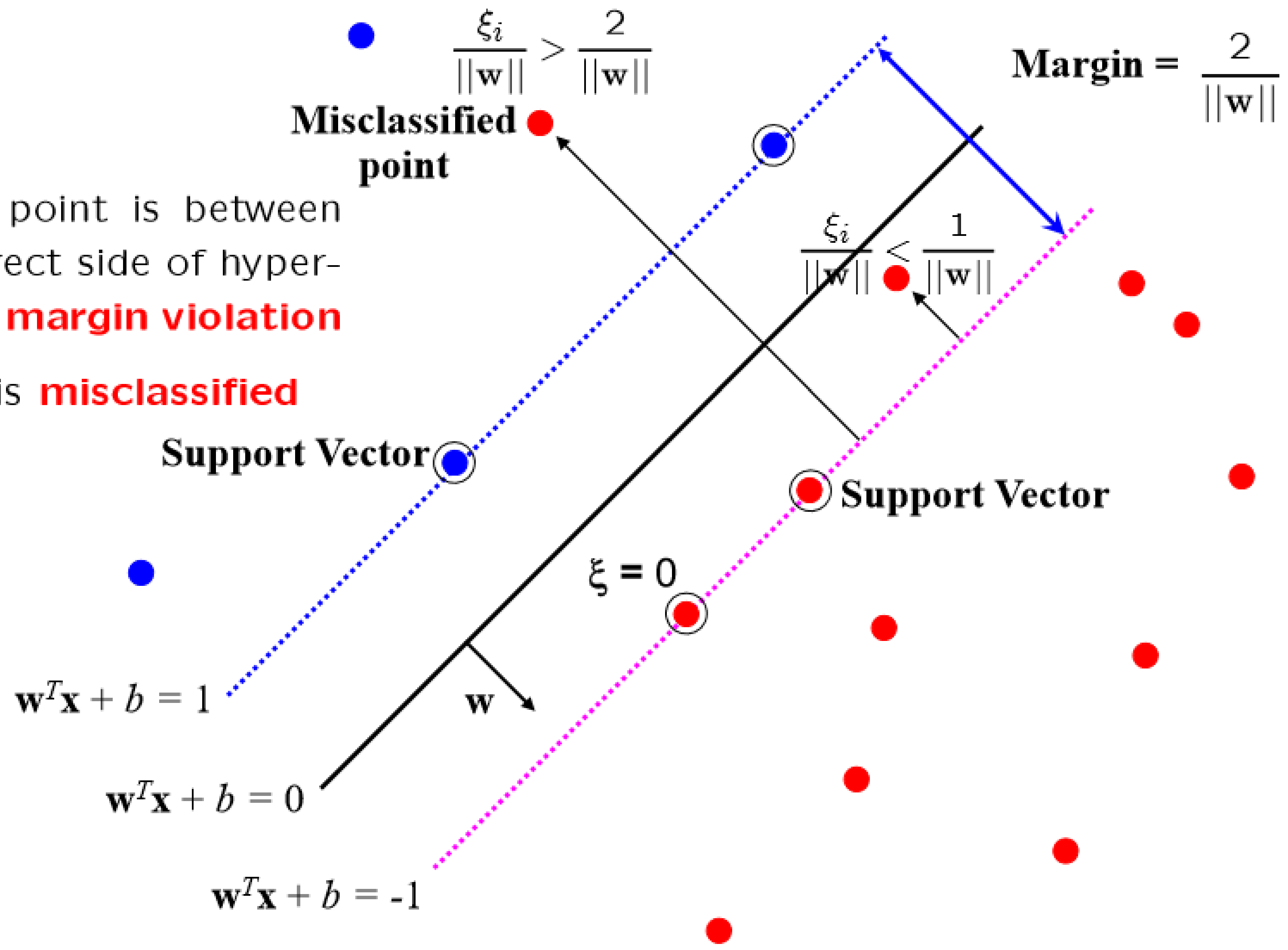
• but possibly the large margin solution is better, even though one constraint is violated

In general there is a trade off between the margin and the number of mistakes on the training data

Introduce "slack" variables

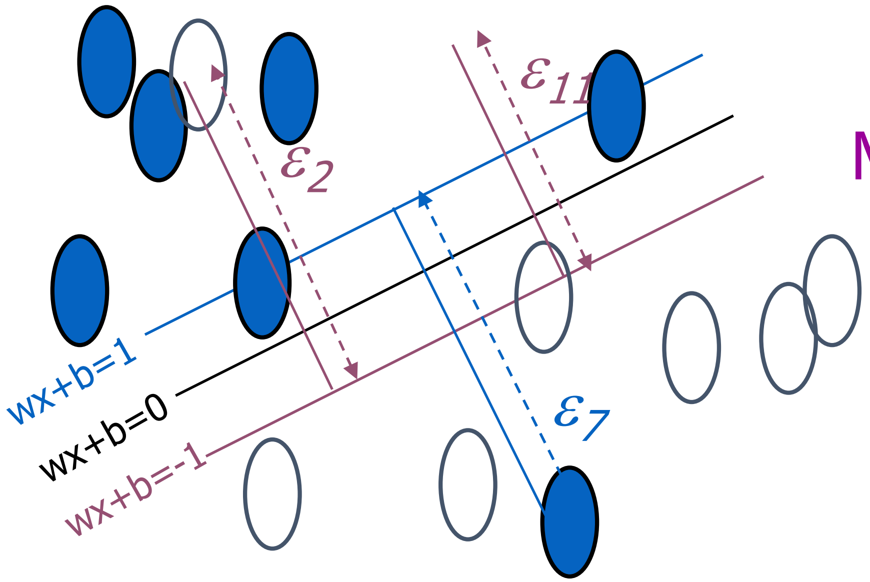
$$\xi_i \geq 0$$

- for $0 < \xi \leq 1$ point is between margin and correct side of hyperplane. This is a **margin violation**
- for $\xi > 1$ point is **misclassified**



Soft Margin Classification

Slack variables ξi **can be added to allow misclassification of difficult or noisy examples.**



What should our quadratic optimization criterion be?

Minimize

$$\frac{1}{2}\mathbf{w}.\mathbf{w} + C\sum_{k=1}^{R}\varepsilon_k$$

“Soft” margin solution

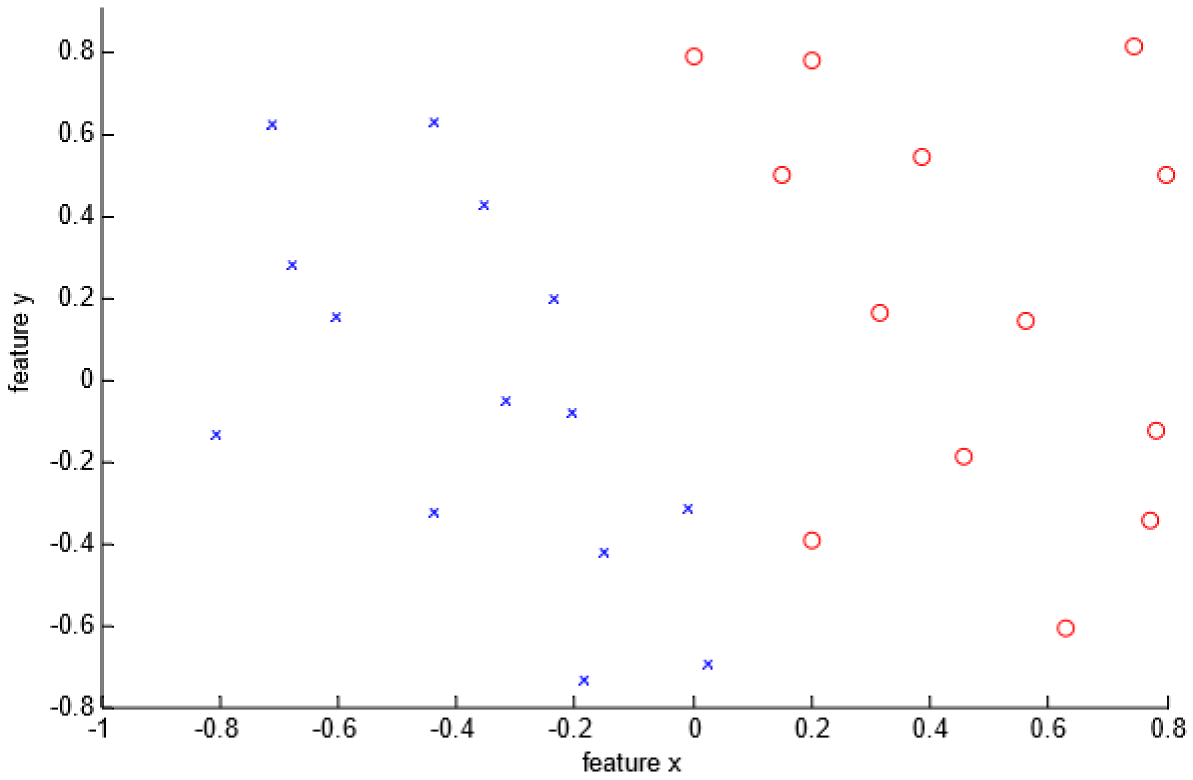
The optimization problem becomes

$$\min_{\mathbf{w}\in\mathbb{R}^d,\xi_i\in\mathbb{R}^+} ||\mathbf{w}||^2 + C\sum_i^N \xi_i$$

subject to

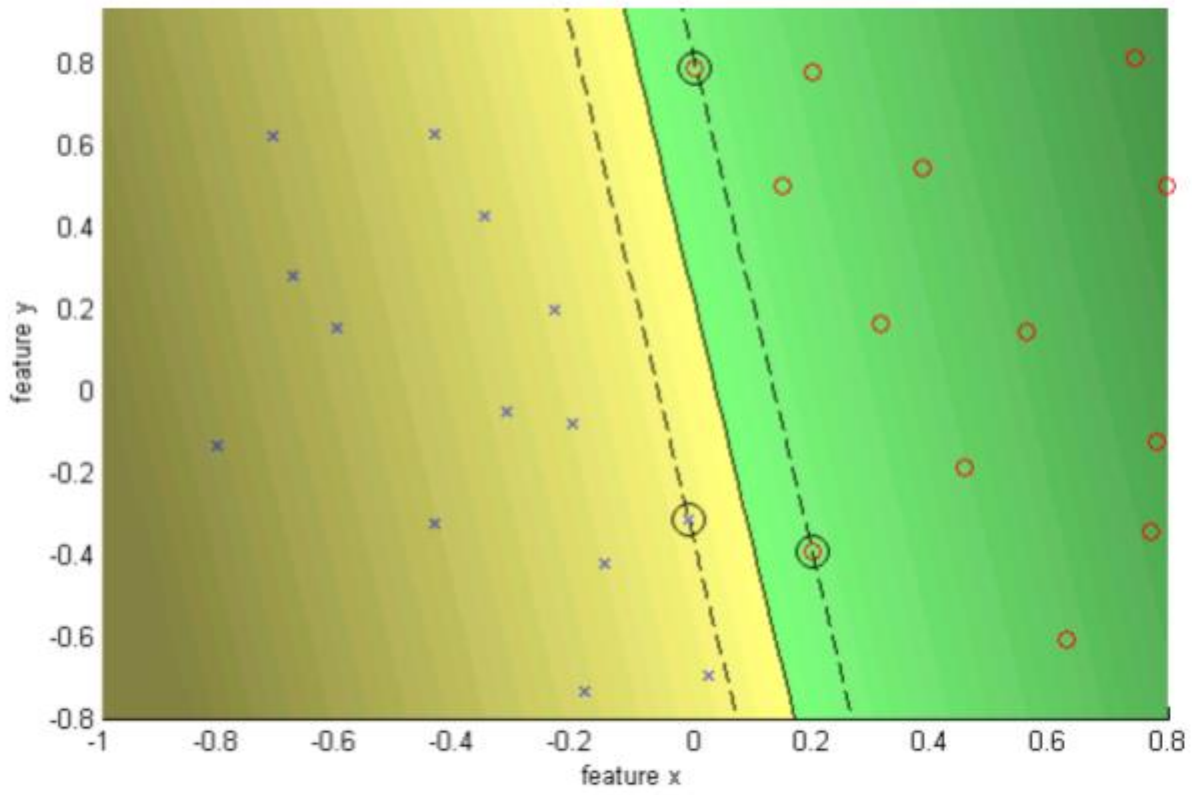
$$y_i\left(\mathbf{w}^\top \mathbf{x}_i + b\right) \geq 1 - \xi_i \text{ for } i = 1 \ldots N$$

- Every constraint can be satisfied if ξ_i is sufficiently large
- C is a regularization parameter:
  - small C allows constraints to be easily ignored $\rightarrow$ large margin
  - large C makes constraints hard to ignore $\rightarrow$ narrow margin
  - $C = \infty$ enforces all constraints: hard margin
- This is still a quadratic optimization problem and there is a unique minimum. Note, there is only one parameter, C.



- data is linearly separable
- but only with a narrow margin

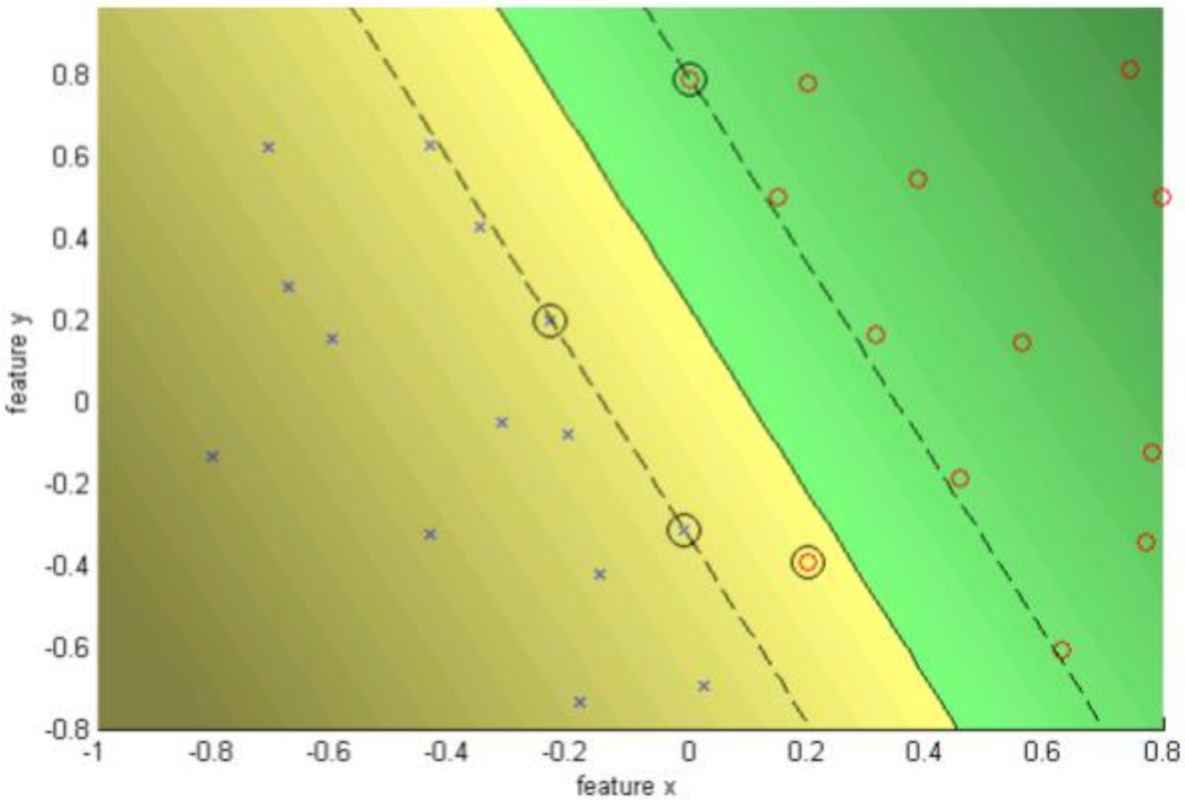
C = Infinity   hard margin



Comment Window

SVM (L1) by Sequential Minimal Optimizer
Kernel: linear (-), C: Inf
Kernel evaluations: 971
Number of Support Vectors: 3
Margin: 0.0966
Training error: 0.00%

C = 10 soft margin



Optimization

Learning an SVM has been formulated as a constrained optimization problem over $\mathbf{w}$ and ξ

$$\min_{\mathbf{w}\in\mathbb{R}^d, \xi_i \in \mathbb{R}^+} ||\mathbf{w}||^2 + C\sum_i^N \xi_i \text{ subject to } y_i\left(\mathbf{w}^\top \mathbf{x}_i + b\right) \geq 1 - \xi_i \text{ for } i = 1 \dots N$$

The constraint $y_i\left(\mathbf{w}^\top \mathbf{x}_i + b\right) \geq 1 - \xi_i$, can be written more concisely as

$$y_i f(\mathbf{x}_i) \geq 1 - \xi_i$$

which, together with $\xi_i \geq 0$, is equivalent to

$$\xi_i = \max\left(0, 1 - y_i f(\mathbf{x}_i)\right)$$

Hence the learning problem is equivalent to the unconstrained optimization problem over $\mathbf{w}$

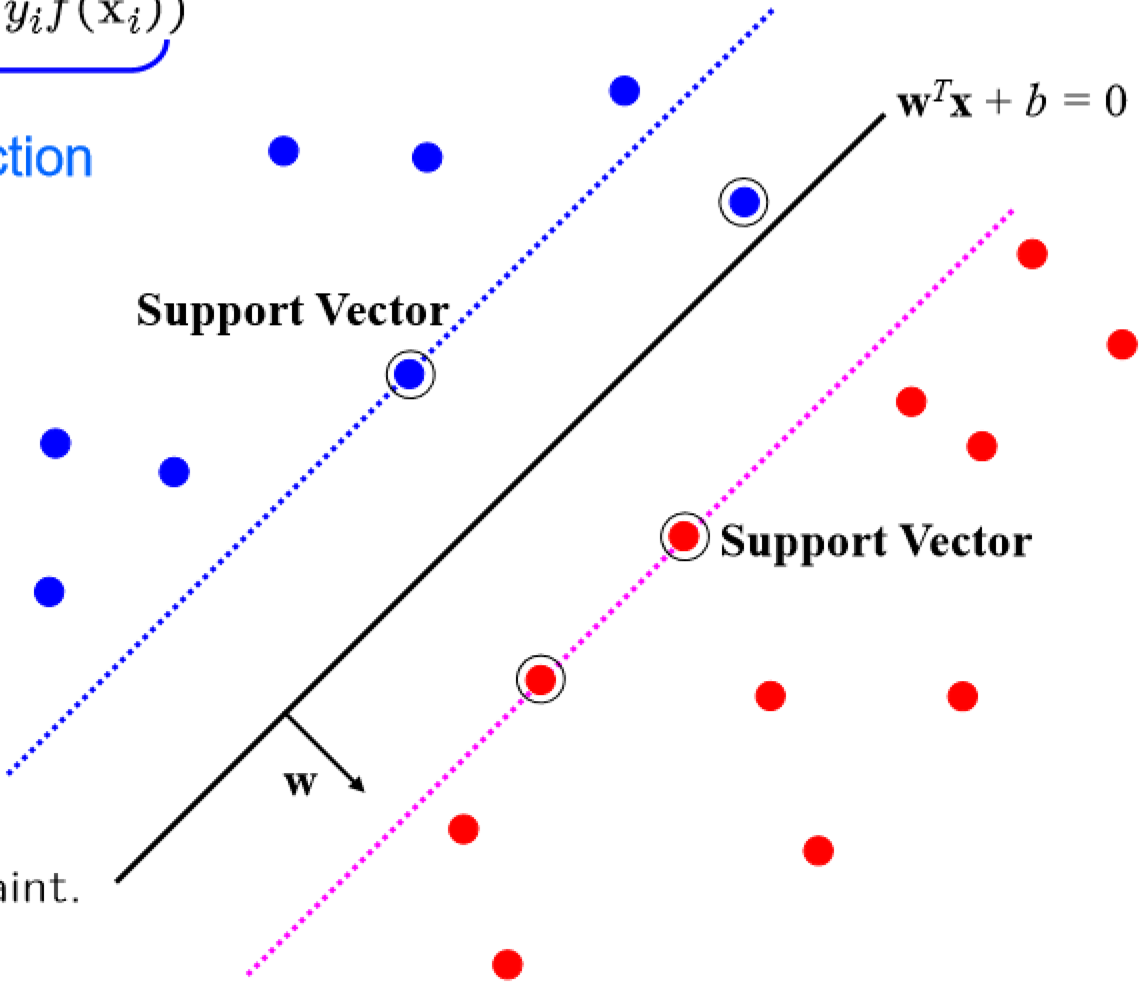
$$\min_{\mathbf{w}\in\mathbb{R}^d} \underbrace{||\mathbf{w}||^2}_{\text{regularization}} + C\sum_i^N \underbrace{\max\left(0, 1 - y_i f(\mathbf{x}_i)\right)}_{\text{loss function}}$$

Loss function

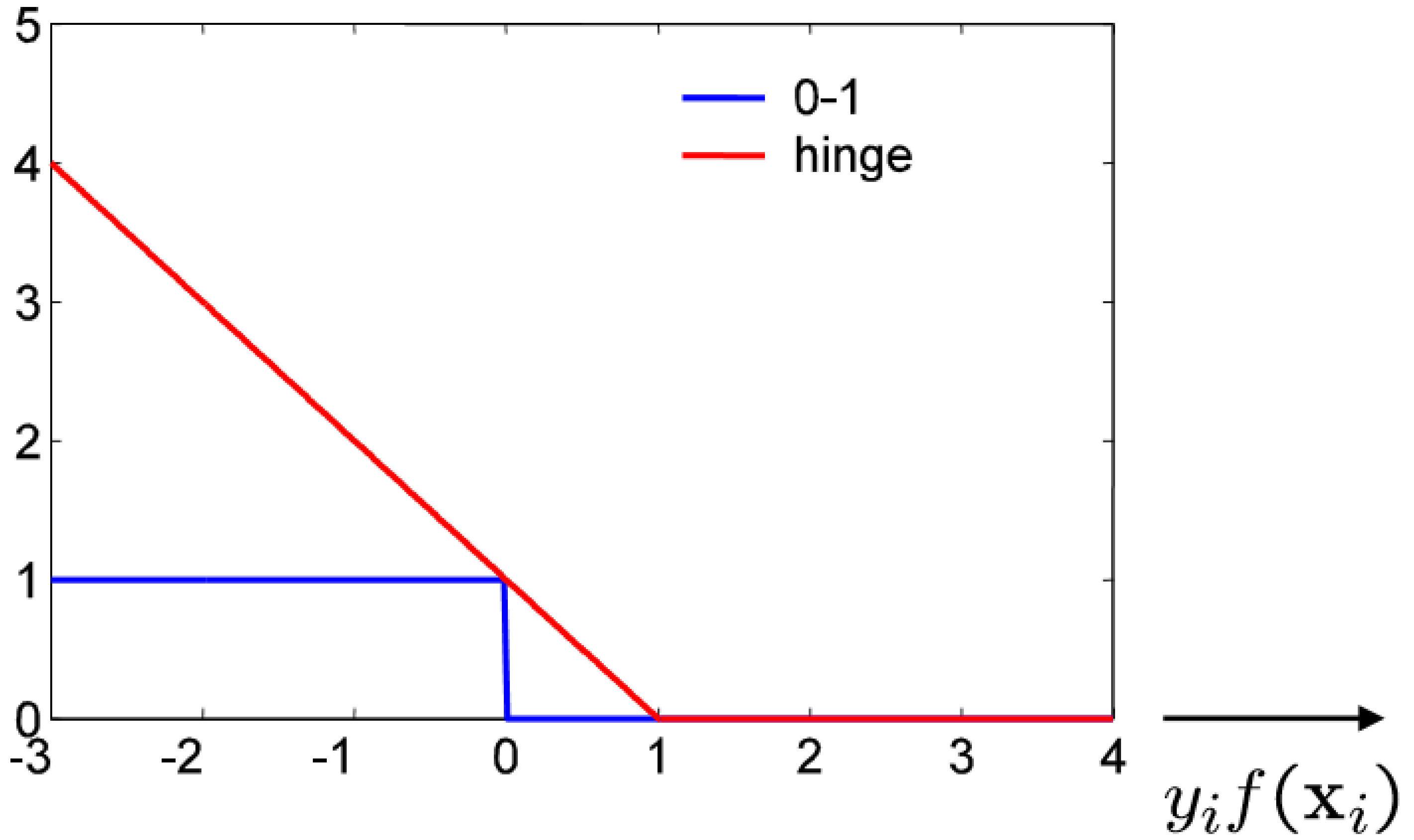
$$\min_{\mathbf{w}\in\mathbb{R}^d} ||\mathbf{w}||^2 + C\sum_i^N \underbrace{\max\left(0, 1 - y_i f(\mathbf{x}_i)\right)}_{\text{loss function}}$$

Points are in three categories:

1. $y_i f(x_i) > 1$
   Point is outside margin.
   No contribution to loss
2. $y_i f(x_i) = 1$
   Point is on margin.
   No contribution to loss.
   As in hard margin case.
3. $y_i f(x_i) < 1$
   Point violates margin constraint.
   Contributes to loss



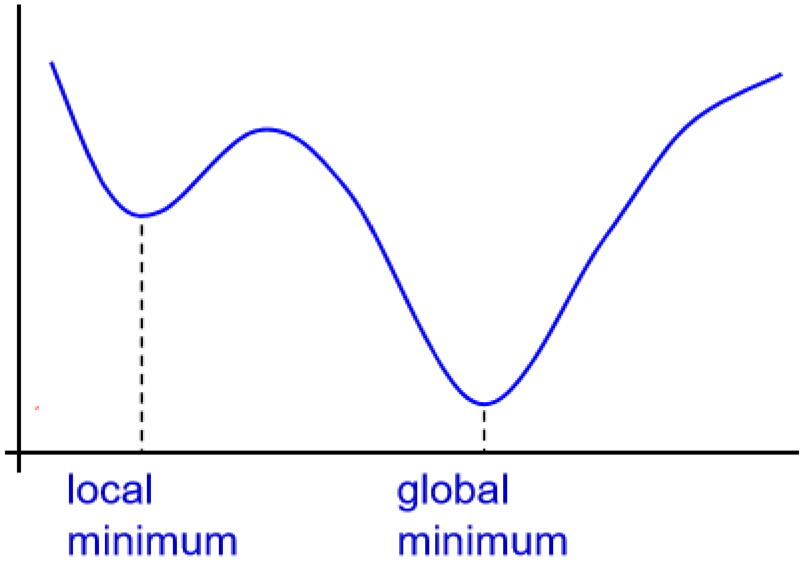
Loss functions



- SVM uses "hinge" loss $\max(0, 1 - y_i f(\mathbf{x}_i))$
- an approximation to the 0-1 loss

Optimization continued

$$\min_{\mathbf{w}\in\mathbb{R}^d} C \sum_i^N \max\left(0, 1 - y_i f(\mathbf{x}_i)\right) + ||\mathbf{w}||^2$$



- Does this cost function have a unique solution?
- Does the solution depend on the starting point of an iterative optimization algorithm (such as gradient descent)?

If the cost function is convex, then a locally optimal point is globally optimal (provided the optimization is over a convex set, which it is in our case)

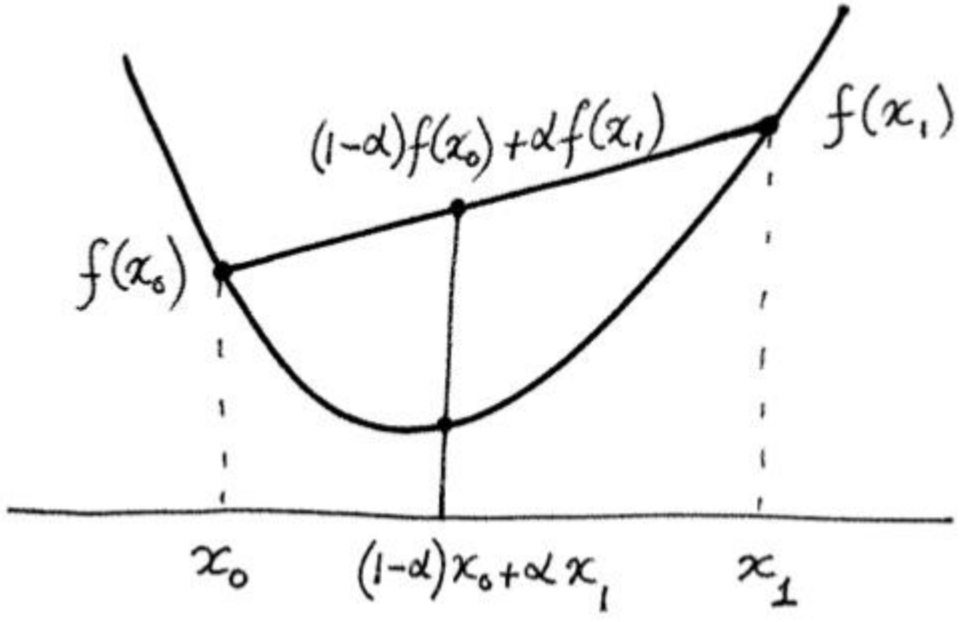
Convex functions

D – a domain in $\mathbb{R}^n$.

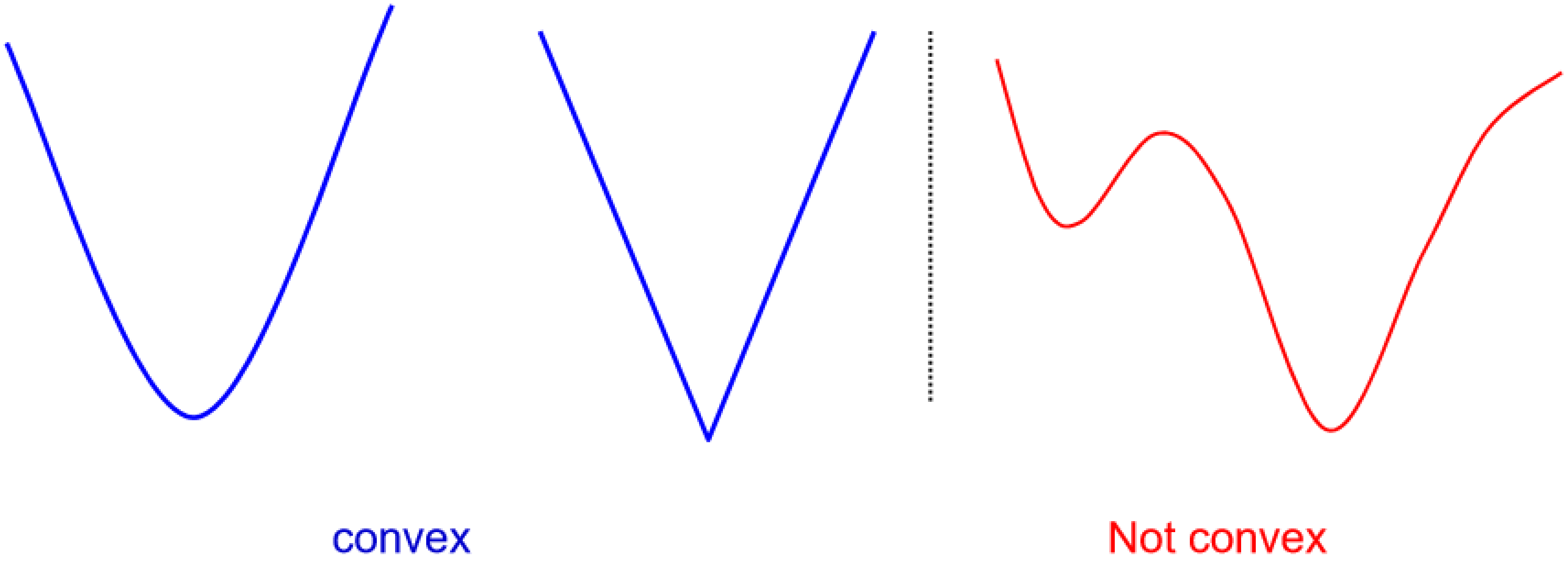
A convex function $f : D \rightarrow \mathbb{R}$ is one that satisfies, for any $\mathbf{x}_0$ and $\mathbf{x}_1$ in D:

$$f((1-\alpha)\mathbf{x}_0 + \alpha\mathbf{x}_1) \leq (1-\alpha)f(\mathbf{x}_0) + \alpha f(\mathbf{x}_1) \quad .$$

Line joining $(\mathbf{x}_0, f(\mathbf{x}_0))$ and $(\mathbf{x}_1, f(\mathbf{x}_1))$ lies above the function graph.



Convex function examples



A non-negative sum of convex functions is convex

SVM

$$\min_{\mathbf{w}\in\mathbb{R}^d} C \sum_{i}^{N} \max\left(0, 1 - y_i f(\mathbf{x}_i)\right) + ||\mathbf{w}||^2$$

convex

Gradient (or steepest) descent algorithm for SVM

To minimize a cost function $\mathcal{C}(\mathbf{w})$ use the iterative update

$$\mathbf{w}_{t+1} \leftarrow \mathbf{w}_t - \eta_t \nabla_{\mathbf{w}} \mathcal{C}(\mathbf{w}_t)$$

where η is the learning rate.

First, rewrite the optimization problem as an average

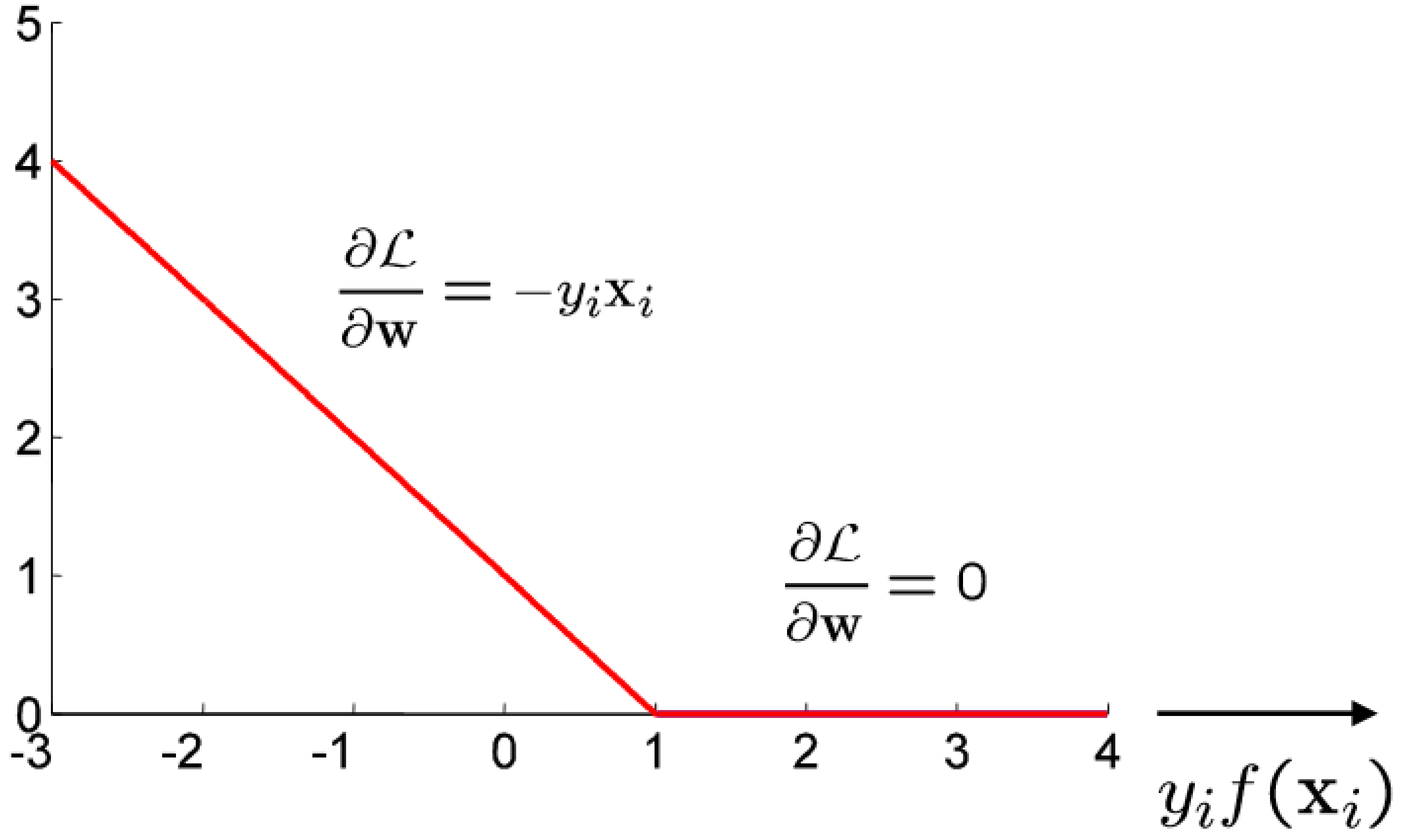
$$\begin{aligned}
\min_{\mathbf{w}} \mathcal{C}(\mathbf{w}) &= \frac{\lambda}{2}||\mathbf{w}||^2 + \frac{1}{N}\sum_i^N \max\left(0, 1 - y_i f(\mathbf{x}_i)\right) \\
&= \frac{1}{N}\sum_i^N \left(\frac{\lambda}{2}||\mathbf{w}||^2 + \max\left(0, 1 - y_i f(\mathbf{x}_i)\right)\right)
\end{aligned}$$

(with $\lambda = 2/(NC)$ up to an overall scale of the problem) and $f(\mathbf{x}) = \mathbf{w}^\top \mathbf{x} + b$

Because the hinge loss is not differentiable, a sub-gradient is computed

Sub-gradient for hinge loss

$$\mathcal{L}(\mathbf{x}_i, y_i; \mathbf{w}) = \max\left(0, 1 - y_i f(\mathbf{x}_i)\right) \qquad f(\mathbf{x}_i) = \mathbf{w}^\top \mathbf{x}_i + b$$



Sub-gradient descent algorithm for SVM

$$\mathcal{C}(\mathbf{w}) = \frac{1}{N}\sum_i^N \left(\frac{\lambda}{2}||\mathbf{w}||^2 + \mathcal{L}(\mathbf{x}_i, y_i; \mathbf{w})\right)$$

The iterative update is

$$\begin{aligned}
\mathbf{w}_{t+1} &\leftarrow \mathbf{w}_t - \eta \nabla_{\mathbf{w}_t} \mathcal{C}(\mathbf{w}_t) \\
&\leftarrow \mathbf{w}_t - \eta \frac{1}{N}\sum_i^N (\lambda \mathbf{w}_t + \nabla_{\mathbf{w}} \mathcal{L}(\mathbf{x}_i, y_i; \mathbf{w}_t))
\end{aligned}$$

where η is the learning rate.

Then each iteration t involves cycling through the training data with the updates:

$$\begin{aligned}
\mathbf{w}_{t+1} &\leftarrow \mathbf{w}_t - \eta(\lambda \mathbf{w}_t - y_i \mathbf{x}_i) && \text{if } y_i f(\mathbf{x}_i) < 1 \\
&\leftarrow \mathbf{w}_t - \eta \lambda \mathbf{w}_t && \text{otherwise}
\end{aligned}$$

In the Pegasos algorithm the learning rate is set at $\eta_t = \frac{1}{\lambda t}$

SVM – review

- We have seen that for an SVM learning a linear classifier

$$f(x) = \mathbf{w}^\top \mathbf{x} + b$$

  is formulated as solving an optimization problem over $\mathbf{w}$:

$$\min_{\mathbf{w} \in \mathbb{R}^d} ||\mathbf{w}||^2 + C \sum_i^N \max\left(0, 1 - y_i f(\mathbf{x}_i)\right)$$

- This quadratic optimization problem is known as the primal problem.

- Instead, the SVM can be formulated to learn a linear classifier

$$f(\mathbf{x}) = \sum_i^N \alpha_i y_i (\mathbf{x}_i{}^\top \mathbf{x}) + b$$

  by solving an optimization problem over α_i.

- This is know as the dual problem, and we will look at the advantages of this formulation.

Sketch derivation of dual form

The Representer Theorem states that the solution $\mathbf{w}$ can always be written as a linear combination of the training data:

$$\mathbf{w} = \sum_{j=1}^{N} \alpha_j y_j \mathbf{x}_j$$

Proof: see example sheet .

Now, substitute for $\mathbf{w}$ in $f(x) = \mathbf{w}^\top \mathbf{x} + b$

$$f(x) = \left(\sum_{j=1}^{N} \alpha_j y_j \mathbf{x}_j \right)^\top \mathbf{x} + b = \sum_{j=1}^{N} \alpha_j y_j \left(\mathbf{x}_j{}^\top \mathbf{x} \right) + b$$

and for $\mathbf{w}$ in the cost function $\min_{\mathbf{w}} ||\mathbf{w}||^2$ subject to $y_i \left(\mathbf{w}^\top \mathbf{x}_i + b \right) \geq 1, \forall i$

$$||\mathbf{w}||^2 = \left\{ \sum_j \alpha_j y_j \mathbf{x}_j \right\}^\top \left\{ \sum_k \alpha_k y_k \mathbf{x}_k \right\} = \sum_{jk} \alpha_j \alpha_k y_j y_k (\mathbf{x}_j{}^\top \mathbf{x}_k)$$

Hence, an equivalent optimization problem is over α_j

$$\min_{\alpha_j} \sum_{jk} \alpha_j \alpha_k y_j y_k (\mathbf{x}_j{}^\top \mathbf{x}_k) \quad \text{subject to } y_i \left(\sum_{j=1}^{N} \alpha_j y_j (\mathbf{x}_j{}^\top \mathbf{x}_i) + b \right) \geq 1, \forall i$$

and a few more steps are required to complete the derivation.

Primal and dual formulations

N is number of training points, and d is dimension of feature vector $\mathbf{x}$.

Primal problem: for $\mathbf{w} \in \mathbb{R}^d$

$$\min_{\mathbf{w}\in\mathbb{R}^d} ||\mathbf{w}||^2 + C\sum_i^N \max(0, 1 - y_i f(\mathbf{x}_i))$$

Dual problem: for $\boldsymbol{\alpha} \in \mathbb{R}^N$ (stated without proof):

$$\max_{\alpha_i \geq 0} \sum_i \alpha_i - \frac{1}{2}\sum_{jk} \alpha_j \alpha_k y_j y_k (\mathbf{x}_j{}^\top \mathbf{x}_k) \text{ subject to } 0 \leq \alpha_i \leq C \text{ for } \forall i, \text{ and} \sum_i \alpha_i y_i = 0$$

- Need to learn d parameters for primal, and N for dual
- If $N << d$ then more efficient to solve for α than $\mathbf{w}$
- Dual form only involves $(\mathbf{x}_j{}^\top \mathbf{x}_k)$. We will return to why this is an advantage when we look at kernels.

Primal and dual formulations

Primal version of classifier:

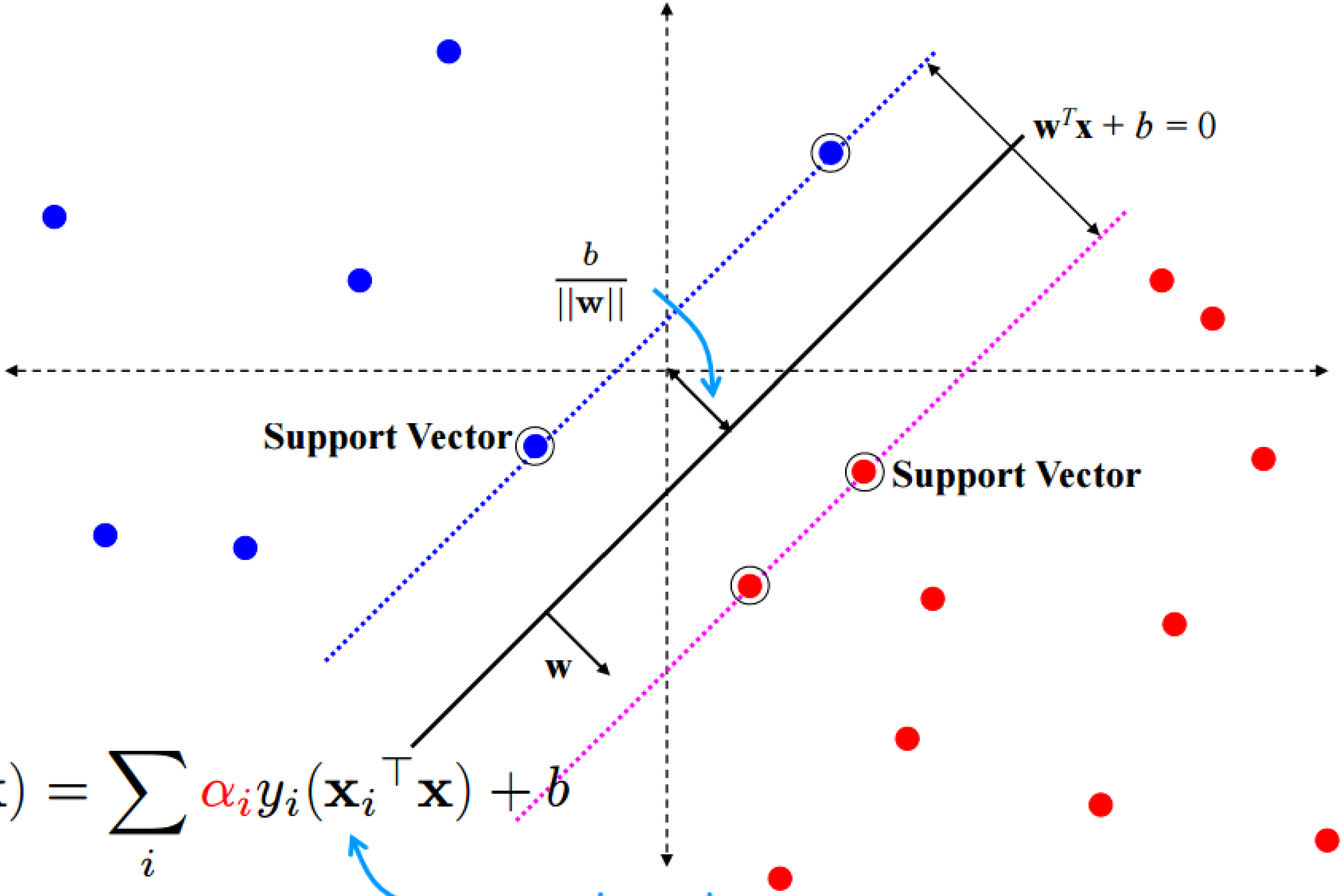
$$f(\mathbf{x}) = \mathbf{w}^\top \mathbf{x} + b$$

Dual version of classifier:

$$f(\mathbf{x}) = \sum_i^N \alpha_i y_i (\mathbf{x}_i{}^\top \mathbf{x}) + b$$

At first sight the dual form appears to have the disadvantage of a K-NN classifier – it requires the training data points $\mathbf{x}_i$. However, many of the α_i's are zero. The ones that are non-zero define the support vectors $\mathbf{x}_i$.

Support Vector Machine

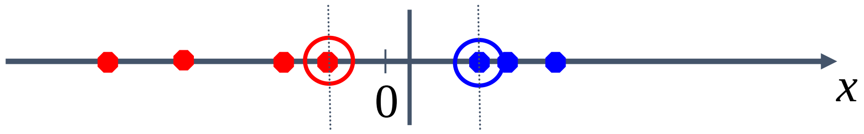


$$f(\mathbf{x}) = \sum_i \alpha_i y_i (\mathbf{x}_i{}^\top \mathbf{x}) + b$$

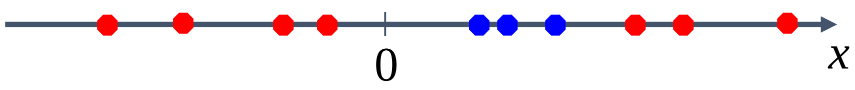
support vectors

Non-linear SVMs

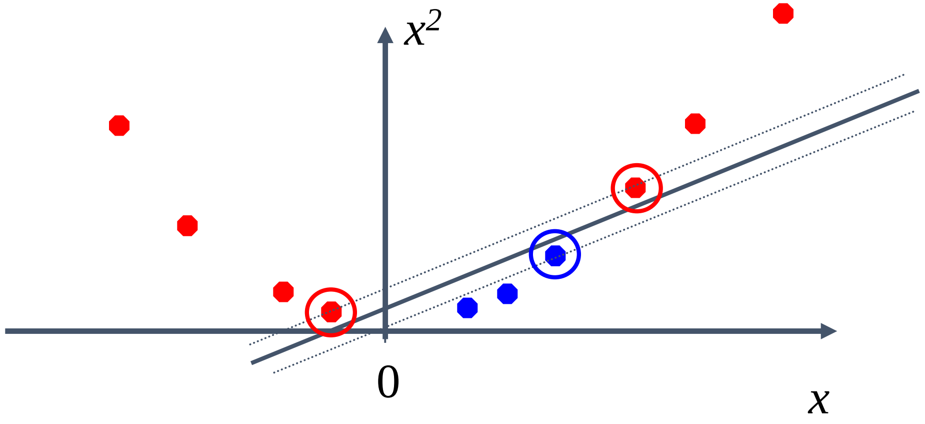
- Datasets that are linearly separable with some noise work out great:



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

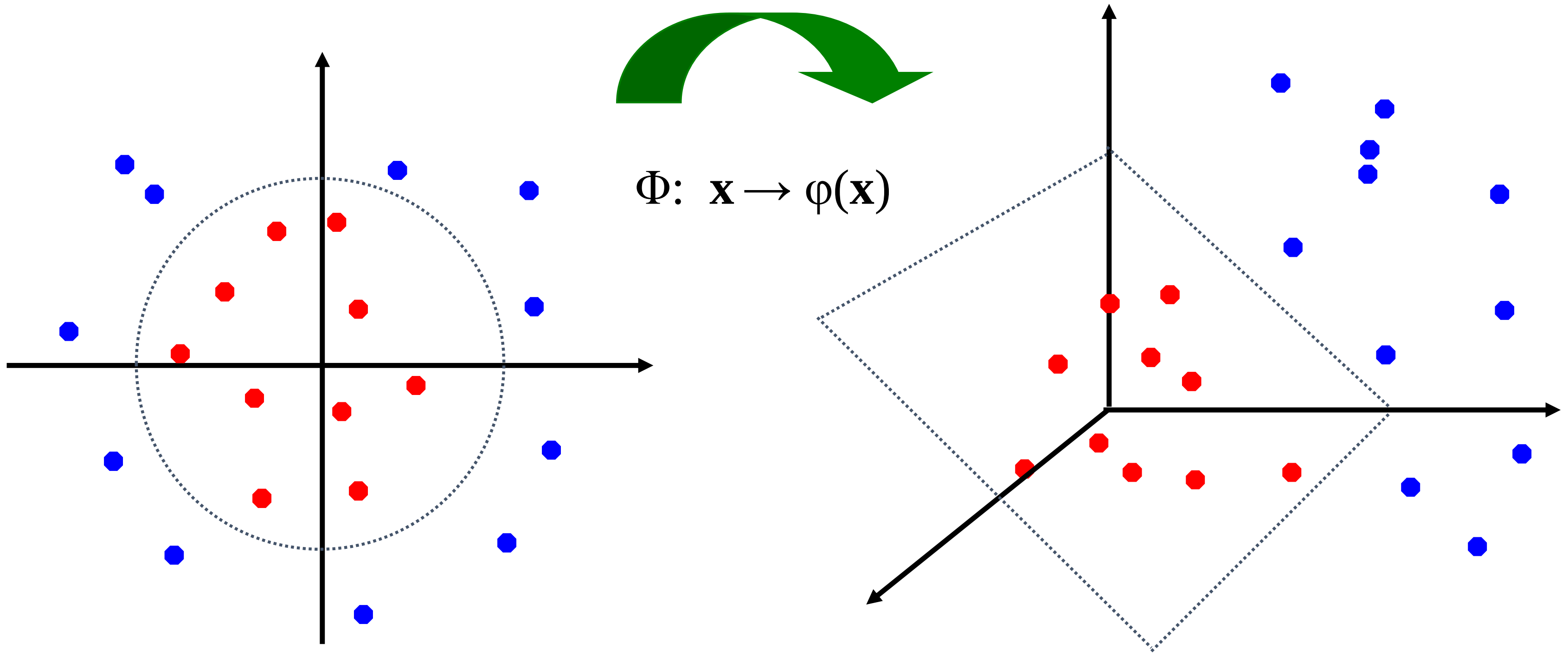


- How about… mapping data to a higher-dimensional space:

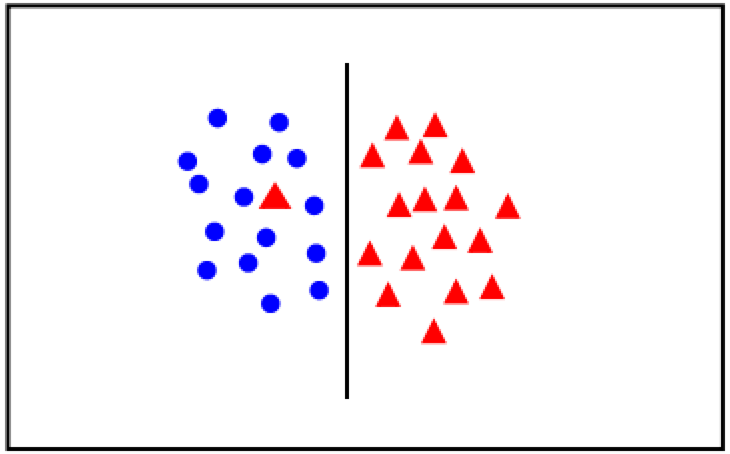


Non-linear SVMs: Feature spaces

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



Handling data that is not linearly separable

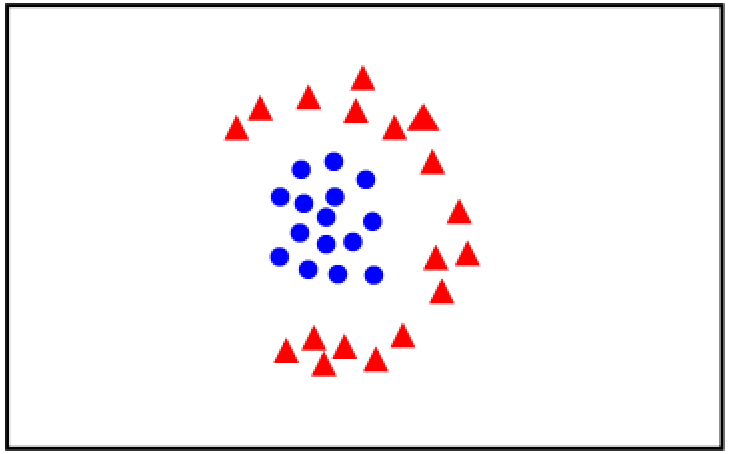


- introduce slack variables

$$\min_{\mathbf{w}\in\mathbb{R}^d, \xi_i\in\mathbb{R}^+} ||\mathbf{w}||^2 + C\sum_i^N \xi_i$$

subject to

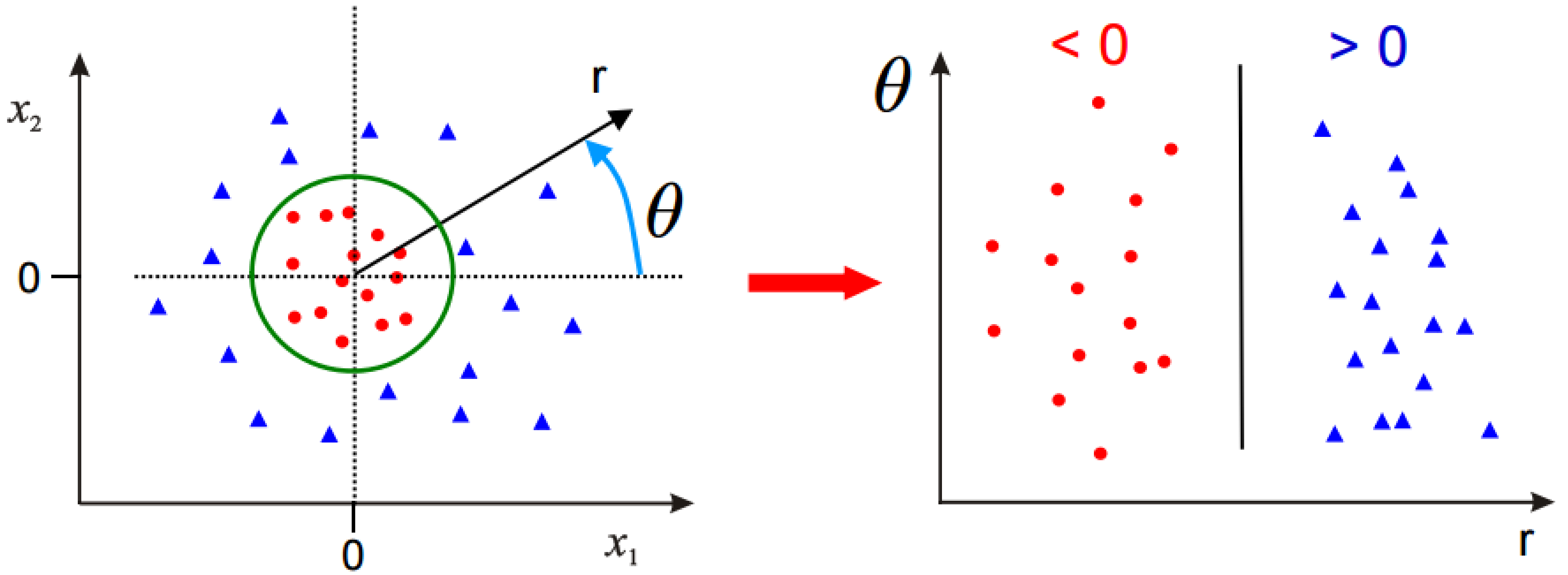
$$y_i\left(\mathbf{w}^\top\mathbf{x}_i + b\right) \geq 1 - \xi_i \text{ for } i = 1\ldots N$$



- linear classifier not appropriate

??

Solution 1: use polar coordinates

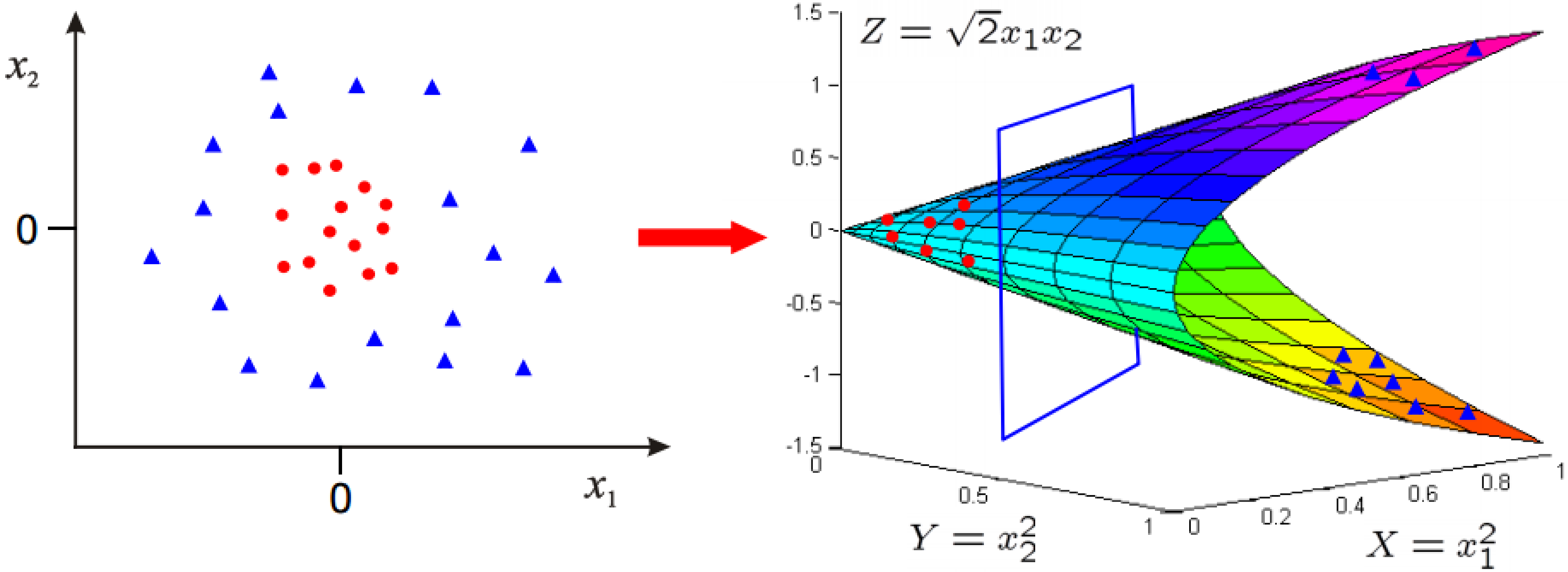


- Data is linearly separable in polar coordinates
- Acts non-linearly in original space

$$\Phi : \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} \rightarrow \begin{pmatrix} r \\ \theta \end{pmatrix} \quad \mathbb{R}^2 \rightarrow \mathbb{R}^2$$

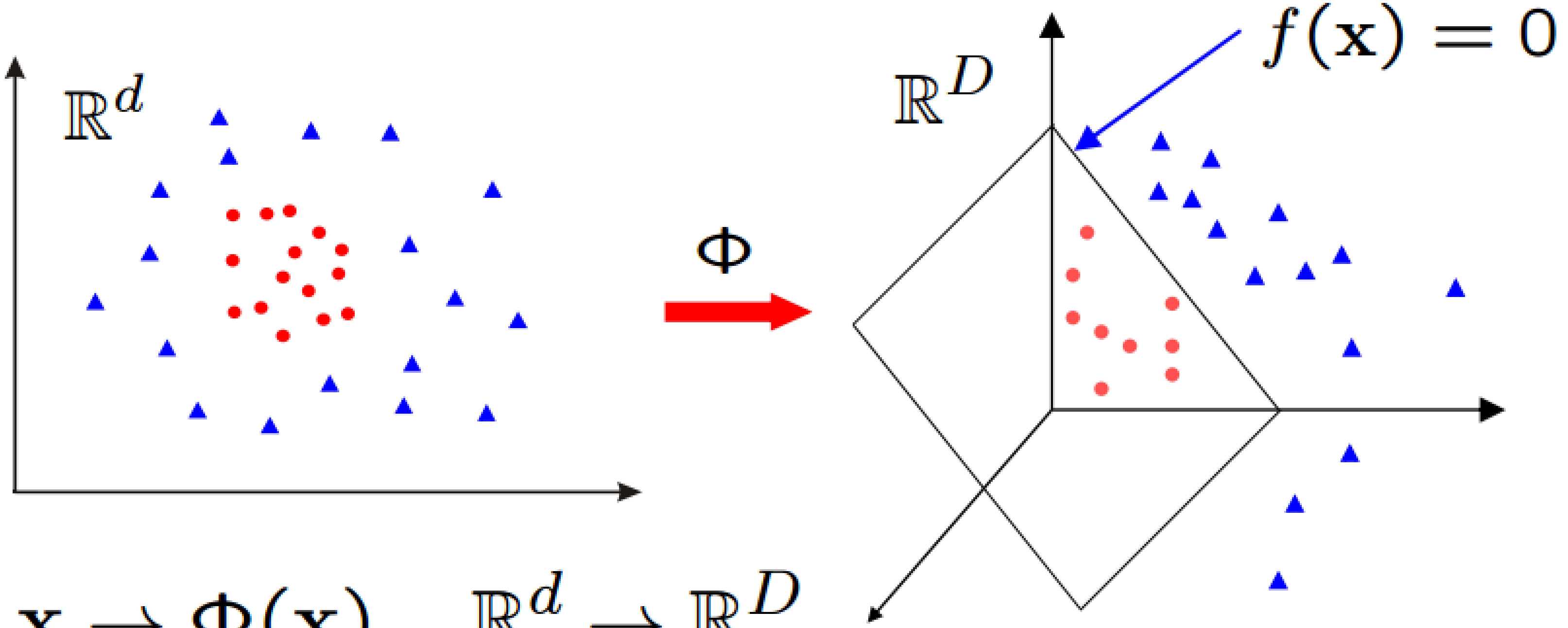
Solution 2: map data to higher dimension

$$\Phi : \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} \to \begin{pmatrix} x_1^2 \\ x_2^2 \\ \sqrt{2}x_1x_2 \end{pmatrix} \qquad \mathbb{R}^2 \to \mathbb{R}^3$$



- Data is linearly separable in 3D
- This means that the problem can still be solved by a linear classifier

SVM classifiers in a transformed feature space



$\Phi : \mathbf{x} \to \Phi(\mathbf{x}) \quad \mathbb{R}^d \to \mathbb{R}^D$

Learn classifier linear in $\mathbf{w}$ for $\mathbb{R}^D$:

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

$\Phi(\mathbf{x})$ is a feature map

Primal Classifier in transformed feature space

Classifier, with $\mathbf{w} \in \mathbb{R}^D$:

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

Learning, for $\mathbf{w} \in \mathbb{R}^D$

$$\min_{\mathbf{w} \in \mathbb{R}^D} ||\mathbf{w}||^2 + C \sum_i^N \max(0, 1 - y_i f(\mathbf{x}_i))$$

- Simply map $\mathbf{x}$ to $\Phi(\mathbf{x})$ where data is separable
- Solve for $\mathbf{w}$ in high dimensional space $\mathbb{R}^D$
- If $D >> d$ then there are many more parameters to learn for $\mathbf{w}$. Can this be avoided?

Dual Classifier in transformed feature space

Classifier:

$$
\begin{aligned}
f(\mathbf{x}) &= \sum_i^N \alpha_i y_i \, \mathbf{x}_i{}^\top \mathbf{x} + b \\
\rightarrow f(\mathbf{x}) &= \sum_i^N \alpha_i y_i \, \boldsymbol{\Phi}(\mathbf{x}_i)^\top \boldsymbol{\Phi}(\mathbf{x}) + b
\end{aligned}
$$

Learning:

$$
\max_{\alpha_i \geq 0} \sum_i \alpha_i - \frac{1}{2} \sum_{jk} \alpha_j \alpha_k y_j y_k \mathbf{x}_j{}^\top \mathbf{x}_k
$$

$$
\rightarrow \quad \max_{\alpha_i \geq 0} \sum_i \alpha_i - \frac{1}{2} \sum_{jk} \alpha_j \alpha_k y_j y_k \boldsymbol{\Phi}(\mathbf{x}_j)^\top \boldsymbol{\Phi}(\mathbf{x}_k)
$$

subject to

$$
0 \leq \alpha_i \leq C \text{ for } \forall i, \text{ and } \sum_i \alpha_i y_i = 0
$$

Dual Classifier in transformed feature space

- Note, that $\Phi(\mathbf{x})$ only occurs in pairs $\Phi(\mathbf{x}_j)^\top \Phi(\mathbf{x}_i)$
- Once the scalar products are computed, only the N dimensional vector $\boldsymbol{\alpha}$ needs to be learnt; it is not necessary to learn in the D dimensional space, as it is for the primal
- Write $k(\mathbf{x}_j, \mathbf{x}_i) = \Phi(\mathbf{x}_j)^\top \Phi(\mathbf{x}_i)$. This is known as a Kernel

Classifier:

$$f(\mathbf{x}) = \sum_i^N \alpha_i y_i \, k(\mathbf{x}_i, \mathbf{x}) + b$$

Learning:

$$\max_{\alpha_i \geq 0} \sum_i \alpha_i - \frac{1}{2} \sum_{jk} \alpha_j \alpha_k y_j y_k \, k(\mathbf{x}_j, \mathbf{x}_k)$$

subject to

$$0 \leq \alpha_i \leq C \text{ for } \forall i, \text{ and } \sum_i \alpha_i y_i = 0$$

Special transformations

$$\Phi : \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} \rightarrow \begin{pmatrix} x_1^2 \\ x_2^2 \\ \sqrt{2}x_1x_2 \end{pmatrix} \qquad \mathbb{R}^2 \rightarrow \mathbb{R}^3$$

$$\begin{aligned} \Phi(\mathbf{x})^\top \Phi(\mathbf{z}) &= \left(x_1^2, x_2^2, \sqrt{2}x_1x_2\right) \begin{pmatrix} z_1^2 \\ z_2^2 \\ \sqrt{2}z_1z_2 \end{pmatrix} \\ &= x_1^2z_1^2 + x_2^2z_2^2 + 2x_1x_2z_1z_2 \\ &= (x_1z_1 + x_2z_2)^2 \\ &= (\mathbf{x}^\top\mathbf{z})^2 \end{aligned}$$

Kernel Trick

- Classifier can be learnt and applied without explicitly computing $\Phi(\mathbf{x})$
- All that is required is the kernel $k(\mathbf{x}, \mathbf{z}) = (\mathbf{x}^\top\mathbf{z})^2$
- Complexity of learning depends on N (typically it is $O(N^3)$) not on D

Example kernels

- Linear kernels $k(\mathbf{x}, \mathbf{x}') = \mathbf{x}^\top \mathbf{x}'$
- Polynomial kernels $k(\mathbf{x}, \mathbf{x}') = \left(1 + \mathbf{x}^\top \mathbf{x}'\right)^d$ for any $d > 0$
  - Contains all polynomials terms up to degree d
- Gaussian kernels $k(\mathbf{x}, \mathbf{x}') = \exp\left(-||\mathbf{x} - \mathbf{x}'||^2 / 2\sigma^2\right)$ for $\sigma > 0$
  - Infinite dimensional feature space

SVM classifier with Gaussian kernel

N = size of training data

$$f(\mathbf{x}) = \sum_{i}^{N} \alpha_i y_i k(\mathbf{x}_i, \mathbf{x}) + b$$

weight (may be zero)

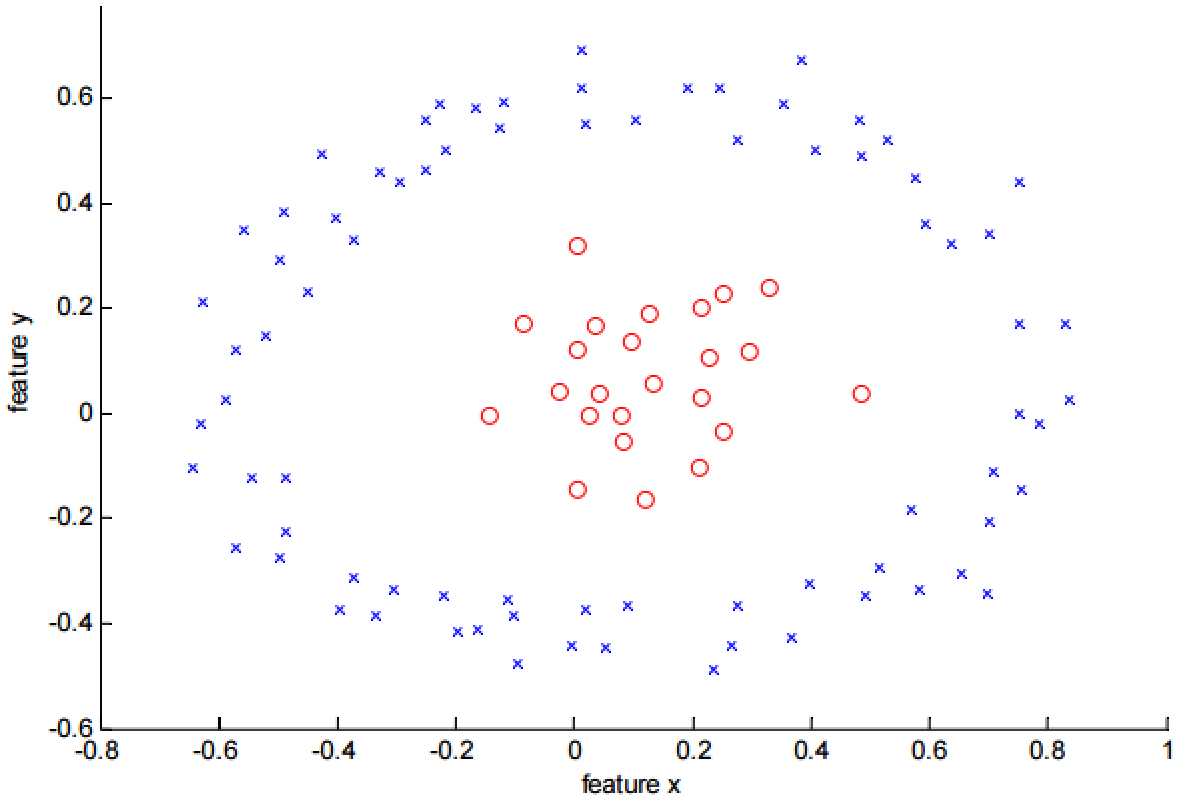
support vector

Gaussian kernel $k(\mathbf{x}, \mathbf{x}') = \exp\left(-||\mathbf{x} - \mathbf{x}'||^2 / 2\sigma^2\right)$

Radial Basis Function (RBF) SVM

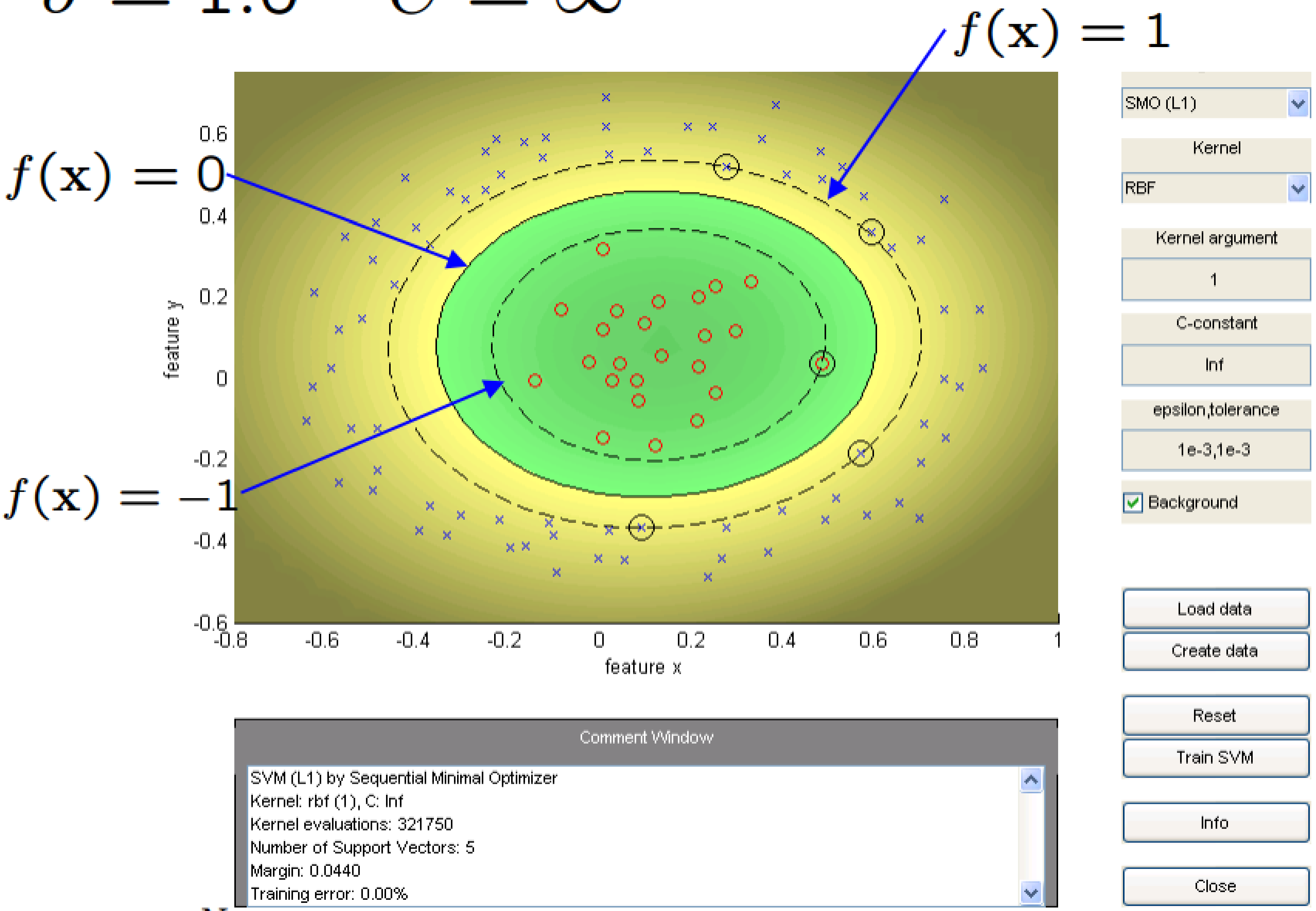
$$f(\mathbf{x}) = \sum_{i}^{N} \alpha_i y_i \exp\left(-||\mathbf{x} - \mathbf{x}_i||^2 / 2\sigma^2\right) + b$$

RBF Kernel SVM Example



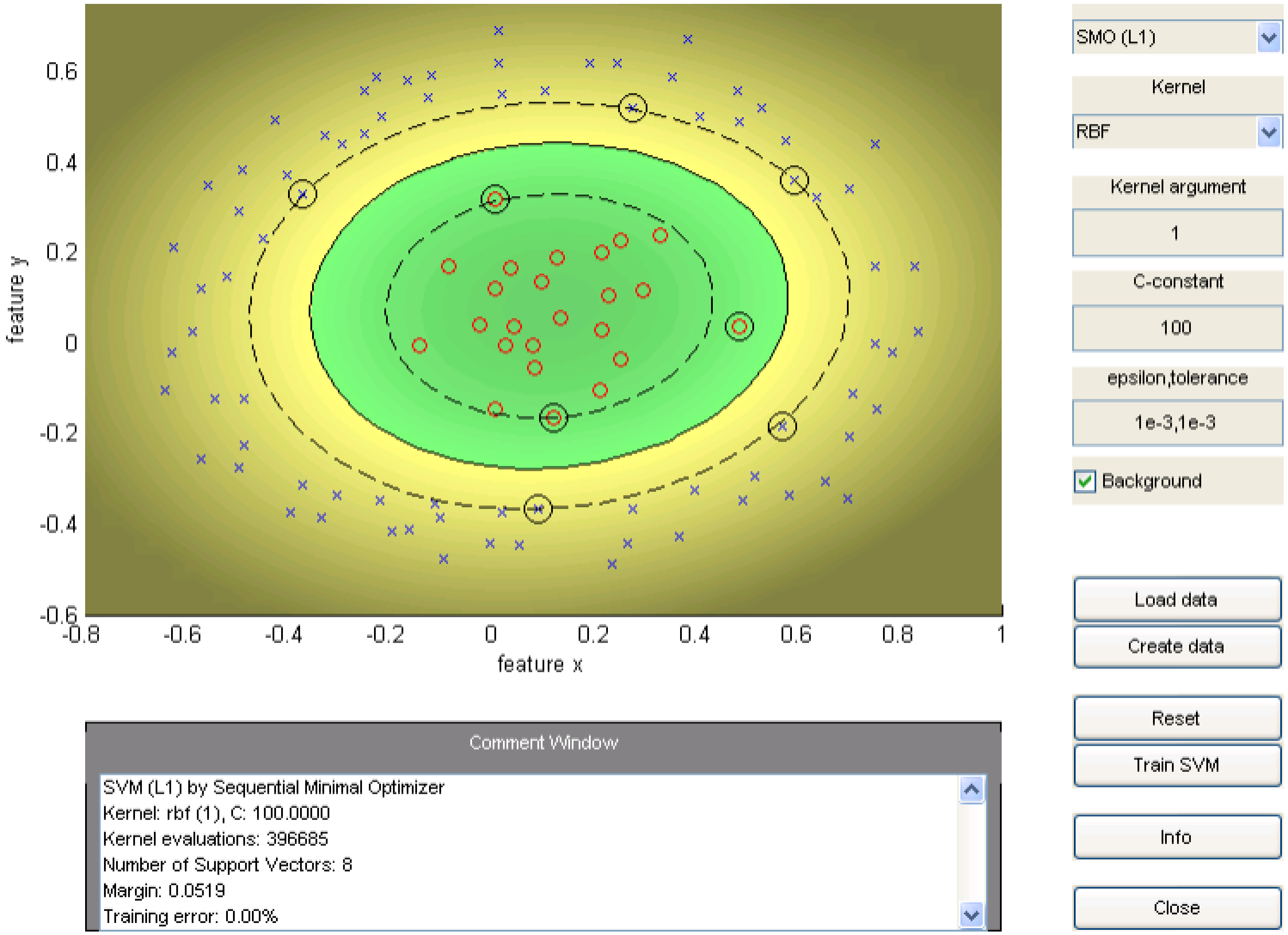
• data is not linearly separable in original feature space

$$\sigma = 1.0 \quad C = \infty$$



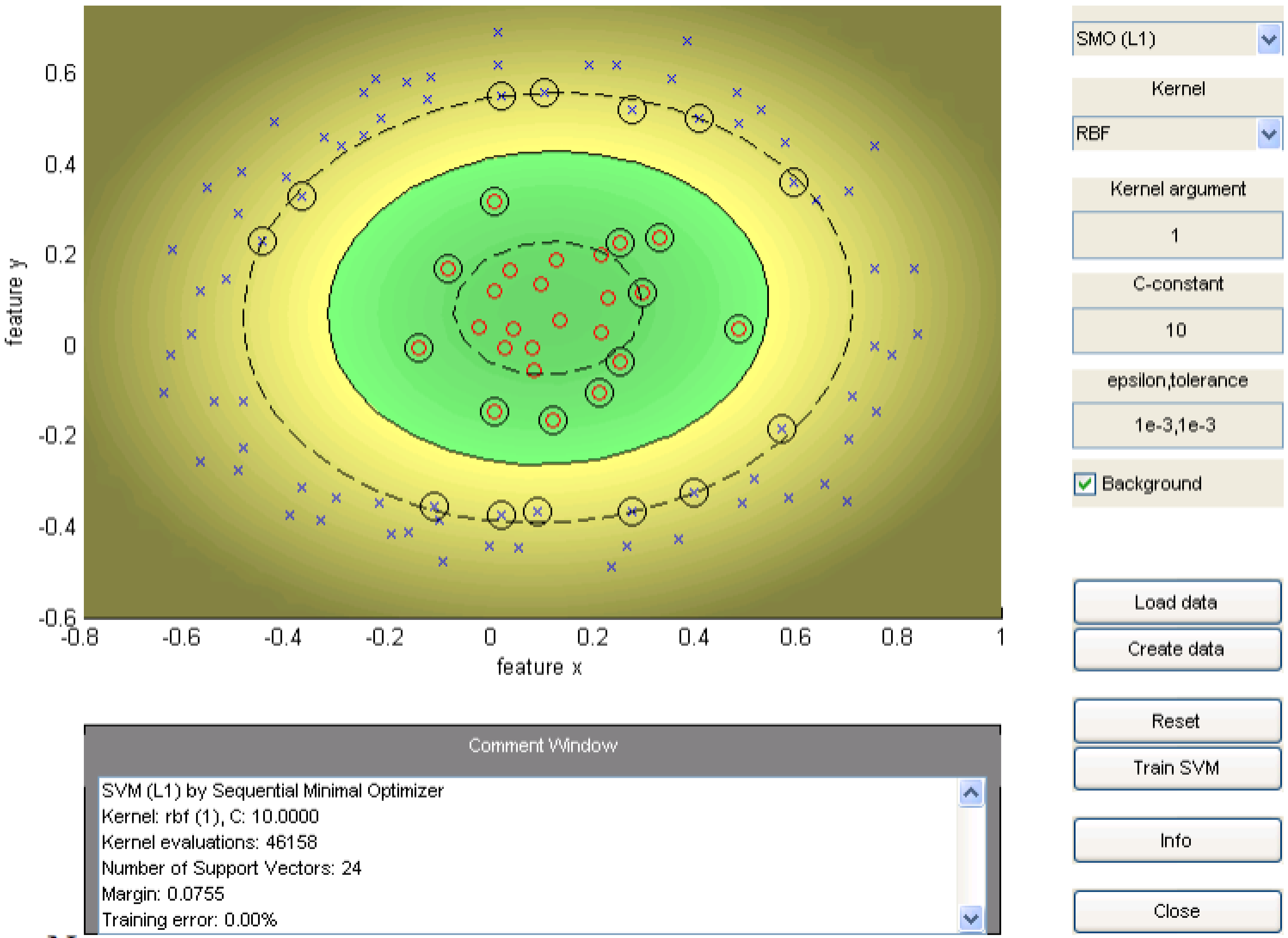
$$f(\mathbf{x}) = \sum_i^N \alpha_i y_i \exp\left(-||\mathbf{x} - \mathbf{x}_i||^2 / 2\sigma^2\right) + b$$

$$\sigma = 1.0 \quad C = 100$$



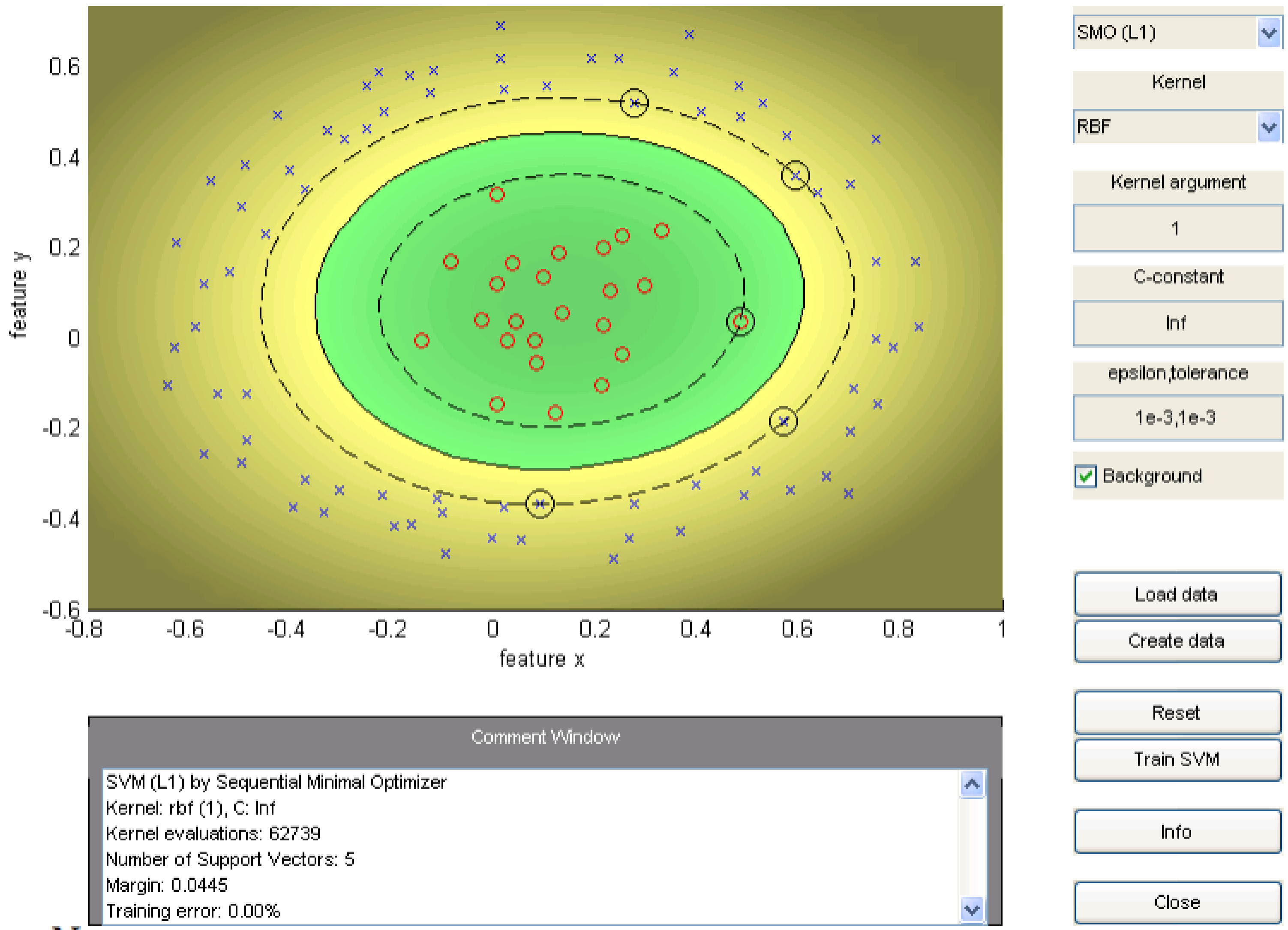
Decrease C, gives wider (soft) margin

$$\sigma = 1.0 \quad C = 10$$



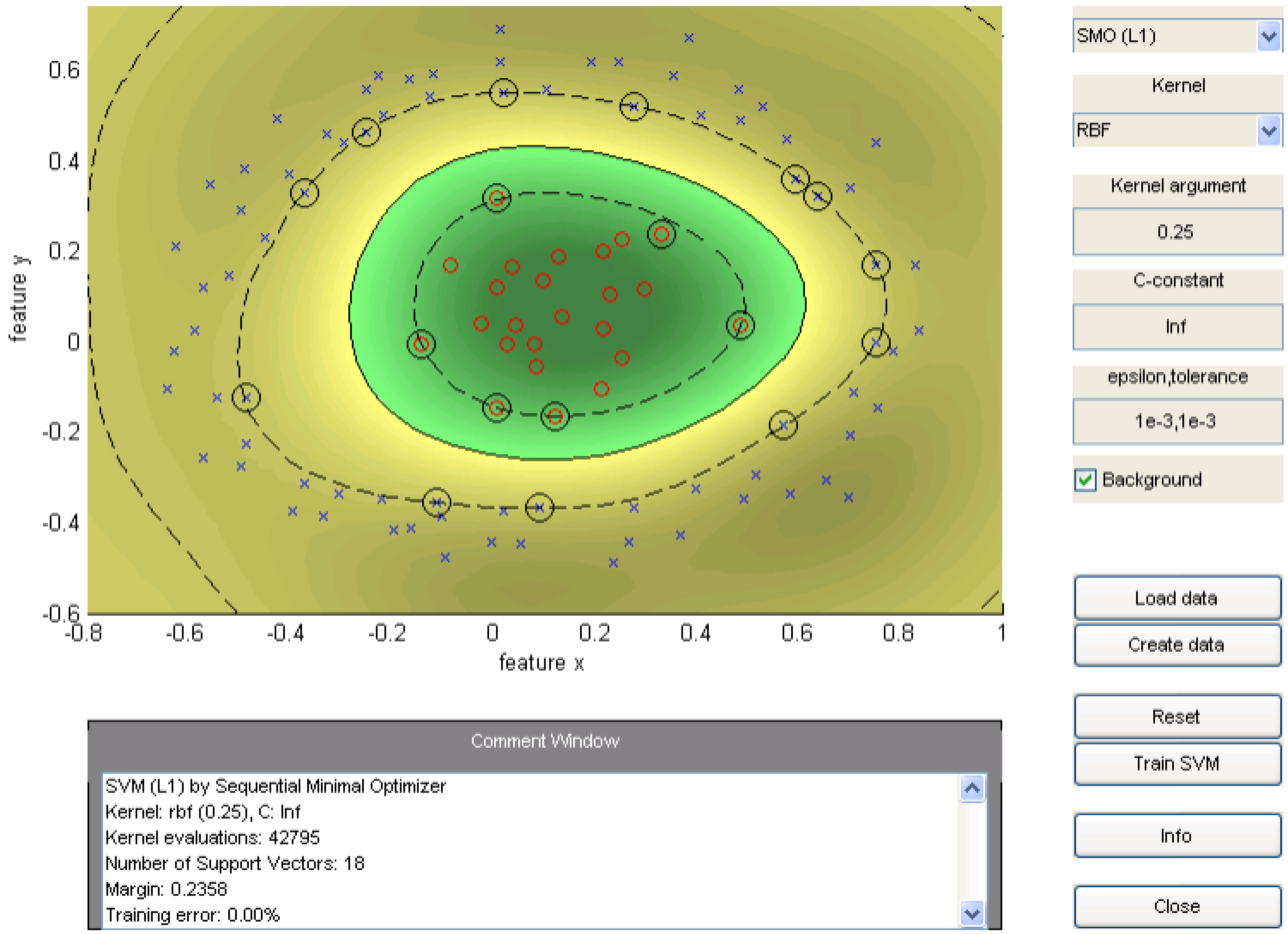
$$f(\mathbf{x}) = \sum_{i}^{N} \alpha_i y_i \exp\left(-||\mathbf{x} - \mathbf{x}_i||^2 / 2\sigma^2\right) + b$$

$$\sigma = 1.0 \quad C = \infty$$



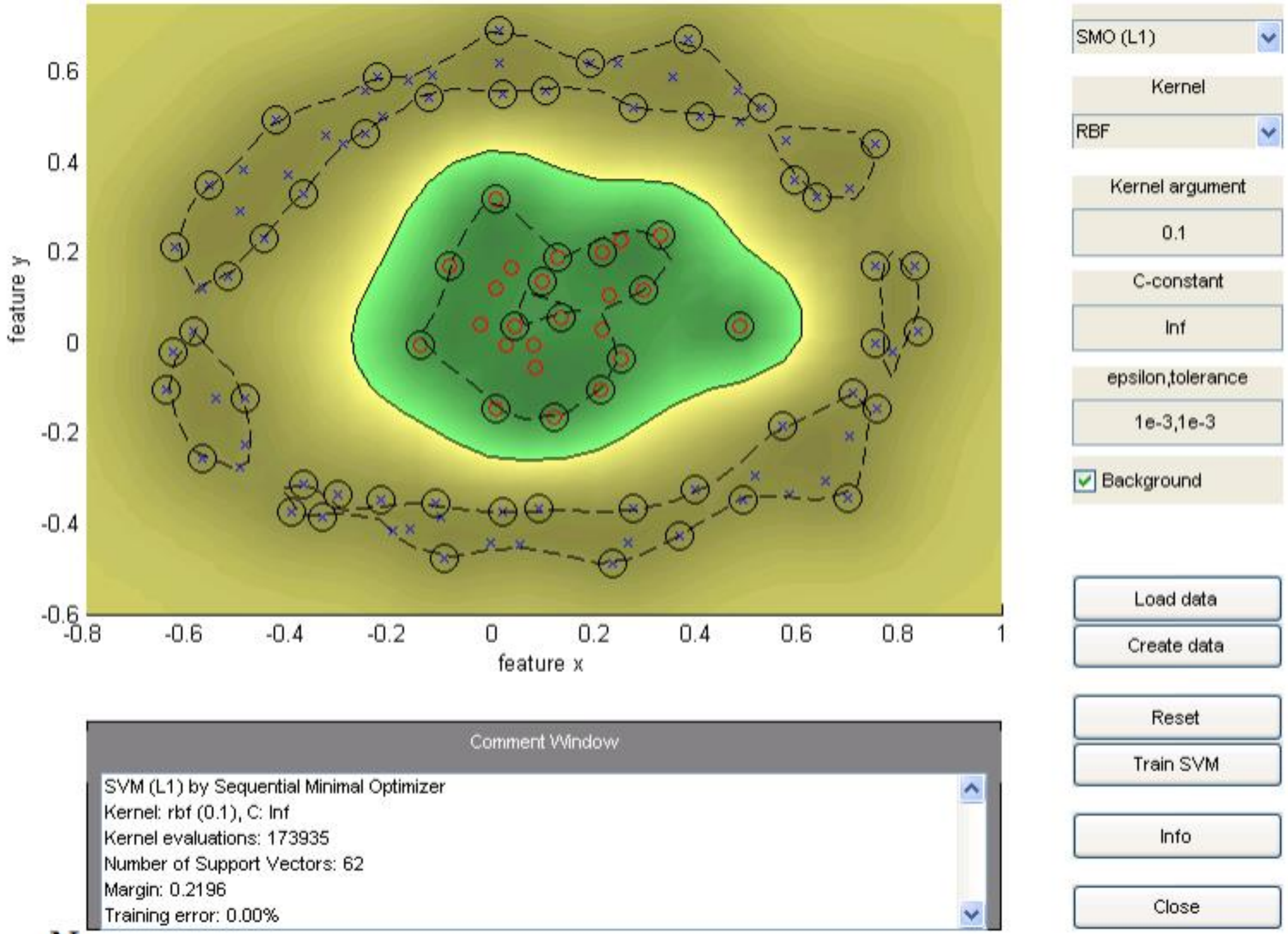
$$f(\mathbf{x}) = \sum_i^N \alpha_i y_i \exp\left(-||\mathbf{x} - \mathbf{x}_i||^2 / 2\sigma^2\right) + b$$

$$\sigma = 0.25 \quad C = \infty$$



Decrease sigma, moves towards nearest neighbour classifier

$$\sigma = 0.1 \quad C = \infty$$



$$f(\mathbf{x}) = \sum_{i}^{N} \alpha_i y_i \exp\left(-||\mathbf{x} - \mathbf{x}_i||^2 / 2\sigma^2\right) + b$$

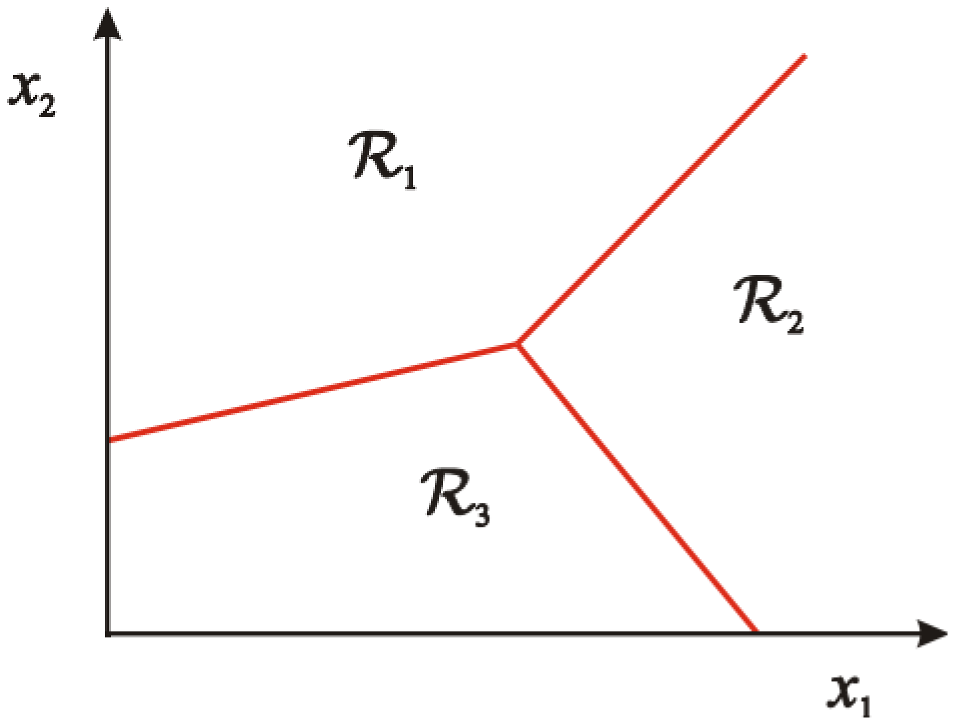
Nonlinear SVM - Overview

- SVM locates a separating hyperplane in the feature space and classify points in that space
- It does not need to represent the space explicitly, simply by defining a kernel function
- The kernel function plays the role of the dot product in the feature space.

Multi-Class Classification – what we would like

Assign input vector $\mathbf{x}$ to one of K classes C_k

Goal: a decision rule that divides input space into K *decision regions* separated by *decision boundaries*



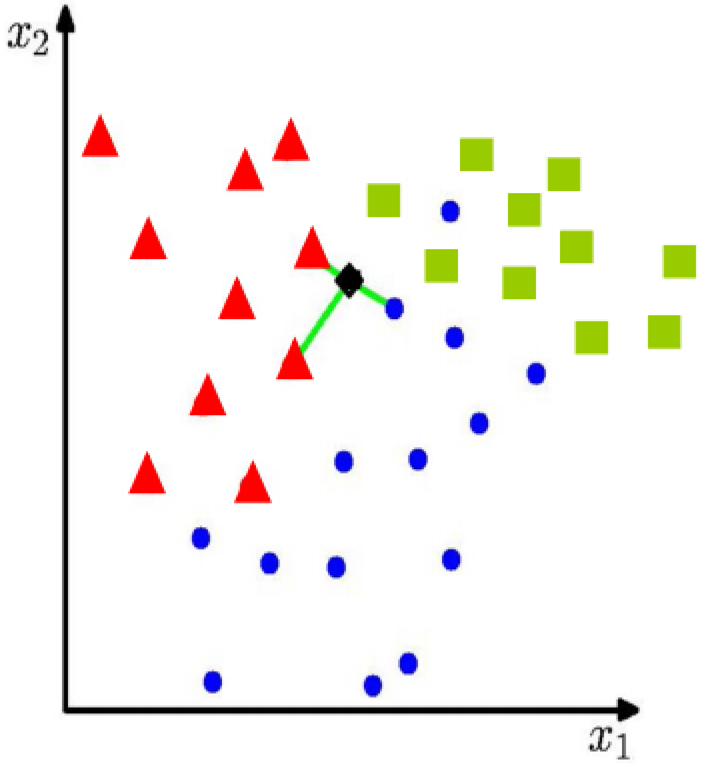
Reminder: K Nearest Neighbour (K-NN) Classifier

Algorithm

- For each test point, x, to be classified, find the K nearest samples in the training data
- Classify the point, x, according to the majority vote of their class labels

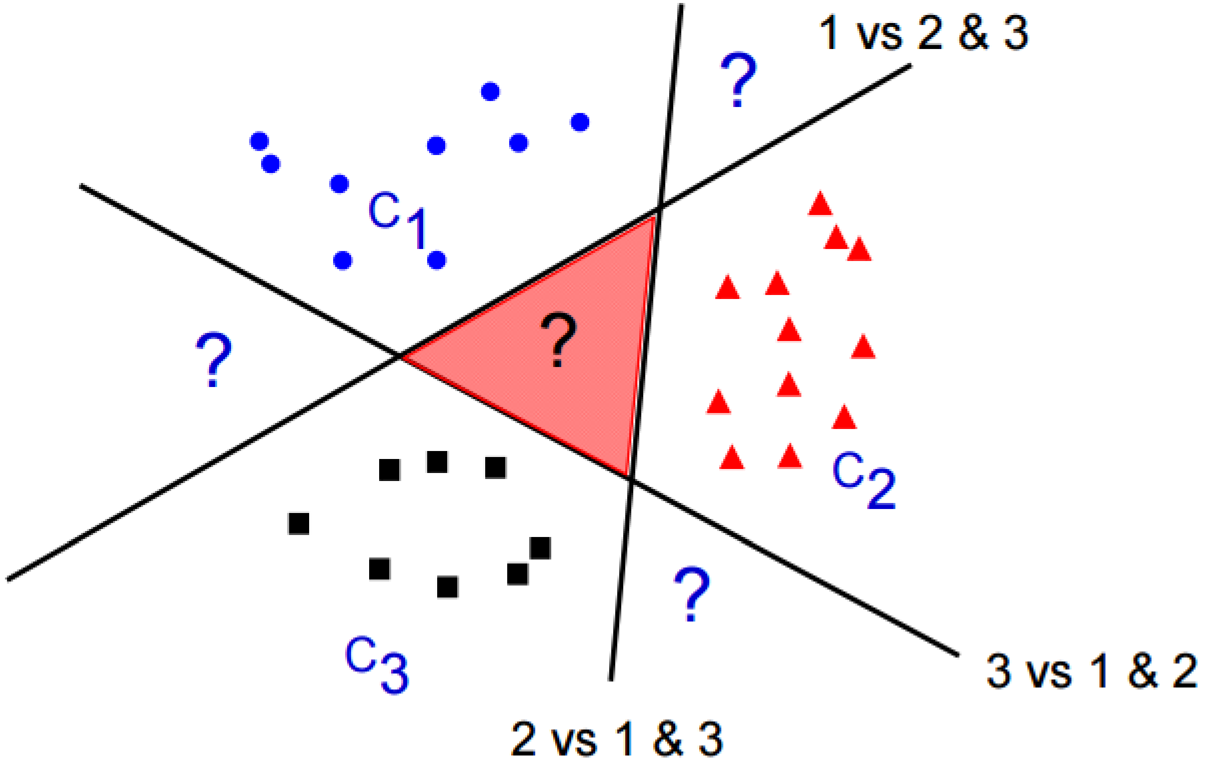
e.g. K = 3

• naturally applicable to multi-class case



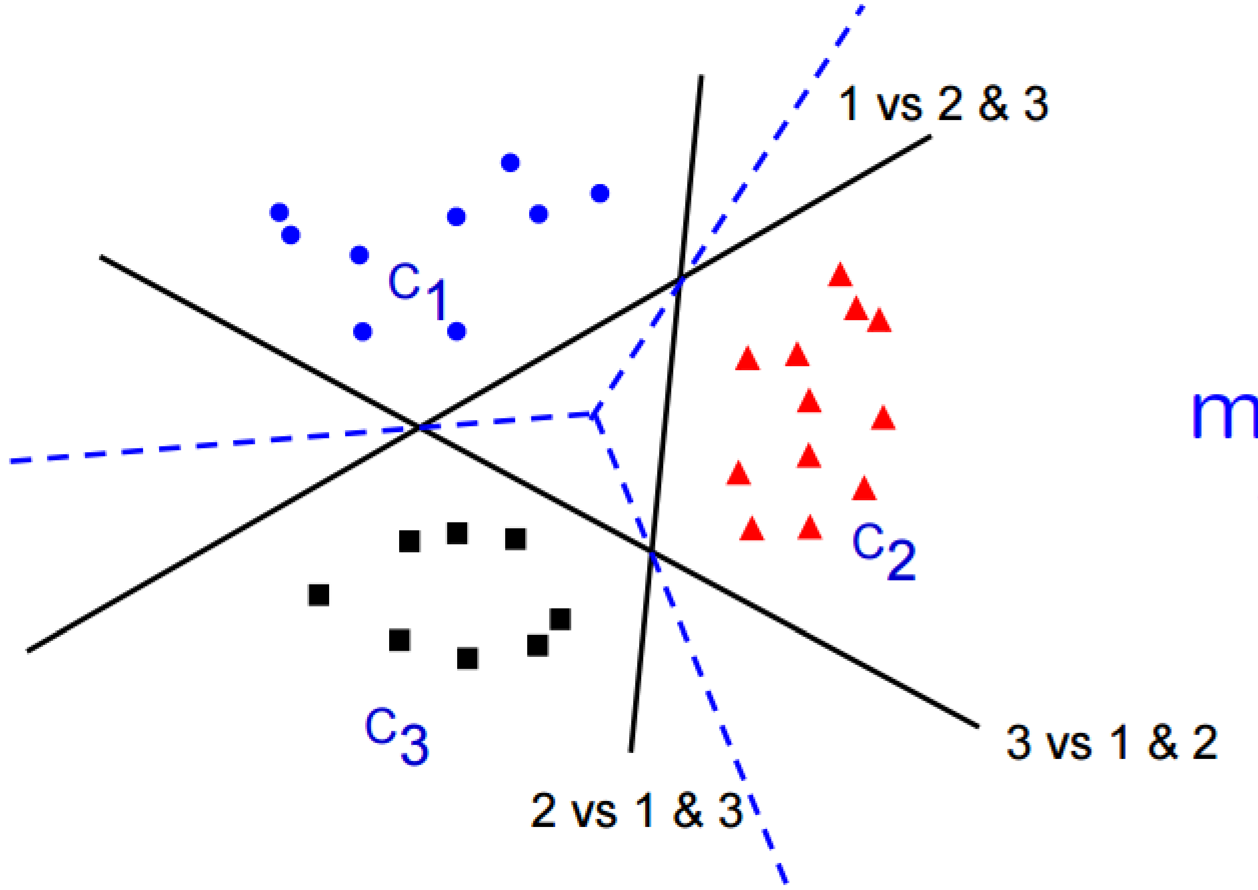
Build from binary classifiers

- Learn: K two-class 1-vs-the-rest classifiers $f_k(\mathbf{x})$



Build from binary classifiers continued

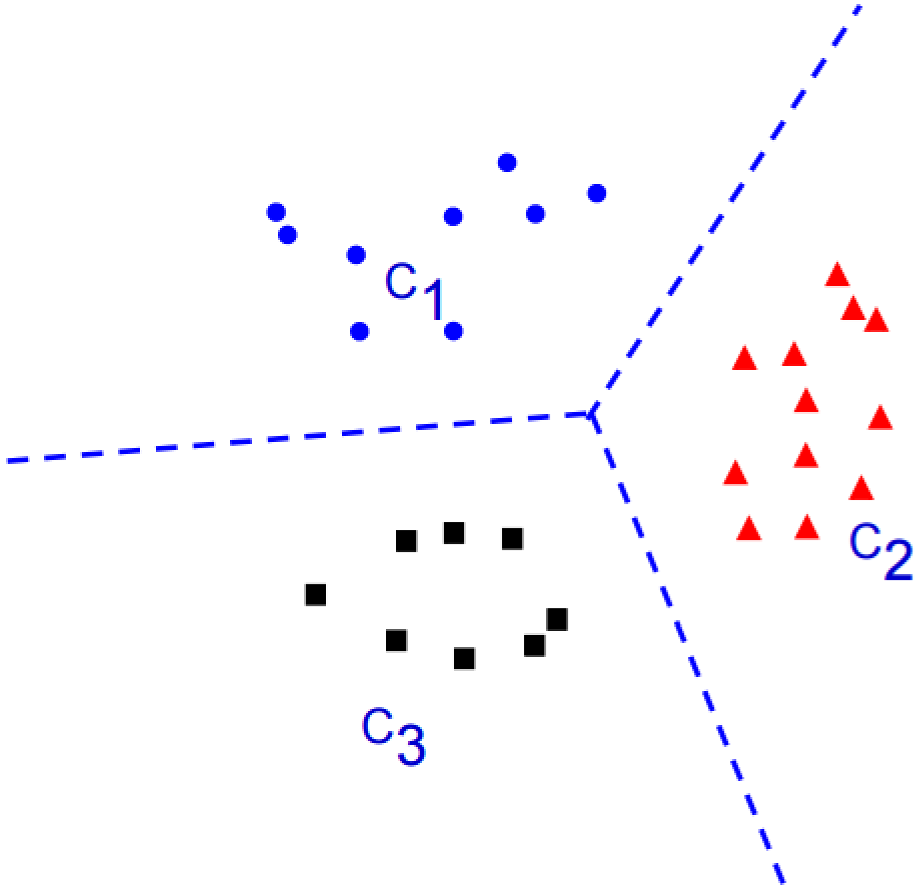
- Learn: K two-class 1 vs the rest classifiers $f_k(\mathbf{x})$
- Classification: choose class with most positive score



$$\max_k f_k(\mathbf{x})$$

Build from binary classifiers continued

- Learn: K two-class 1 vs the rest classifiers $f_k(\mathbf{x})$
- Classification: choose class with most positive score



$$\max_k f_k(\mathbf{x})$$

Why not learn a multi-class SVM directly?

For example for three classes

- Learn $\mathbf{w} = (\mathbf{w}_1, \mathbf{w}_2, \mathbf{w}_3)^\top$ using the cost function

$$\min_{\mathbf{w}} ||\mathbf{w}||^2 \ \text{ subject to}$$

$$\begin{array}{lll}
\mathbf{w}_1{}^\top \mathbf{x}_i \geq \mathbf{w}_2{}^\top \mathbf{x}_i \ \ \& \ \ \mathbf{w}_1{}^\top \mathbf{x}_i \geq \mathbf{w}_3{}^\top \mathbf{x}_i & & \text{for } i \in \text{class } 1 \\
\mathbf{w}_2{}^\top \mathbf{x}_i \geq \mathbf{w}_3{}^\top \mathbf{x}_i \ \ \& \ \ \mathbf{w}_2{}^\top \mathbf{x}_i \geq \mathbf{w}_1{}^\top \mathbf{x}_i & & \text{for } i \in \text{class } 2 \\
\mathbf{w}_3{}^\top \mathbf{x}_i \geq \mathbf{w}_1{}^\top \mathbf{x}_i \ \ \& \ \ \mathbf{w}_3{}^\top \mathbf{x}_i \geq \mathbf{w}_2{}^\top \mathbf{x}_i & & \text{for } i \in \text{class } 3
\end{array}$$

- This is a quadratic optimization problem subject to linear constraints and there is a unique minimum

- Note, a margin can also be included in the constraints

In practice there is a little or no improvement over the binary case

SVM Applications

- **SVM has been used successfully in many real-world problems**

  - text (and hypertext) categorization

  - image classification

  - bioinformatics (Protein classification,
  Cancer classification)

  - hand-written character recognition

Application 1: Cancer Classification

- High Dimensional

  - p>1000; n<100

- Imbalanced

  - less positive samples

$$K[x,x] = k(x,x) + \lambda \frac{n^+}{N}$$

- Many irrelevant features
- Noisy

SVM is sensitive to noisy (mis-labeled) data ☹

Genes				
Patients	**g-1**	**g-2**	**……**	**g-p**
P-1				
p-2				
…….				
p-n				

FEATURE SELECTION

In the linear case,
w_i^2 gives the ranking of dim i

Application 2: Text Categorization

- Task: The classification of natural text (or hypertext) documents into a fixed number of predefined categories based on their content.

  - email filtering, web searching, sorting documents by topic, etc..

- A document can be assigned to more than one category, so this can be viewed as a series of binary classification problems, one for each category

Representation of Text

IR′s vector space model (aka bag-of-words representation)

- A doc is represented by a vector indexed by a pre-fixed set or dictionary of terms
- Values of an entry can be binary or weights

$$\phi_i(x) = \frac{\mathrm{tf}_i \log(\mathrm{idf}_i)}{\kappa},$$

- Normalization, stop words, word stems
- Doc x => $\boldsymbol{\varphi}(x)$

Text Categorization using SVM

- **The distance between two documents is $\phi(x)\cdot\phi(z)$**

- **$K(x,z) = \langle\phi(x)\cdot\phi(z)$ is a valid kernel, SVM can be used with $K(x,z)$ for discrimination.**

- **Why SVM?**

  -High dimensional input space

  -Few irrelevant features (dense concept)

  -Sparse document vectors (sparse instances)

  -Text categorization problems are linearly separable

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.

- **Optimization criterion** – Hard margin v.s. Soft margin

  - a lengthy series of experiments in which various parameters are tested

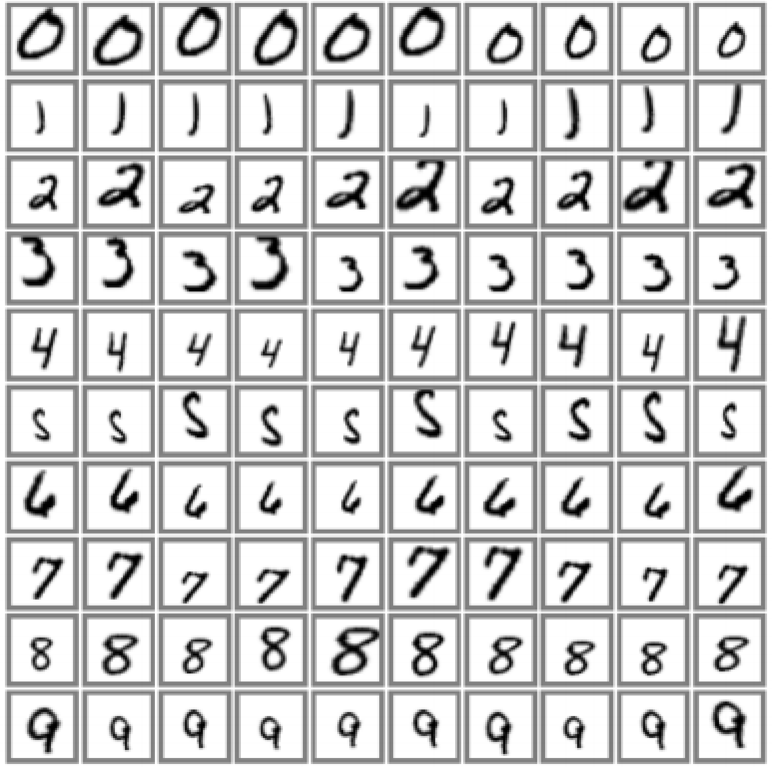
Application: hand written digit recognition

- Feature vectors: each image is 28 x 28 pixels. Rearrange as a 784-vector **x**

- Training: learn k=10 two-class 1-vs-the-rest SVM classifiers $f_k(\mathbf{x})$

- Classification: choose class with most positive score

$$f(\mathbf{x}) = \max_k f_k(\mathbf{x})$$



Application: Pedestrian detection in Computer Vision

Objective: detect (localize) standing humans in an image

- cf face detection with a sliding window classifier

- reduces object detection to binary classification
- does an image window contain a person or not?

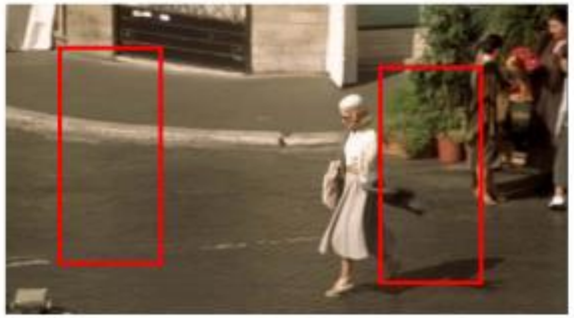
Method: the HOG detector

Training data and features

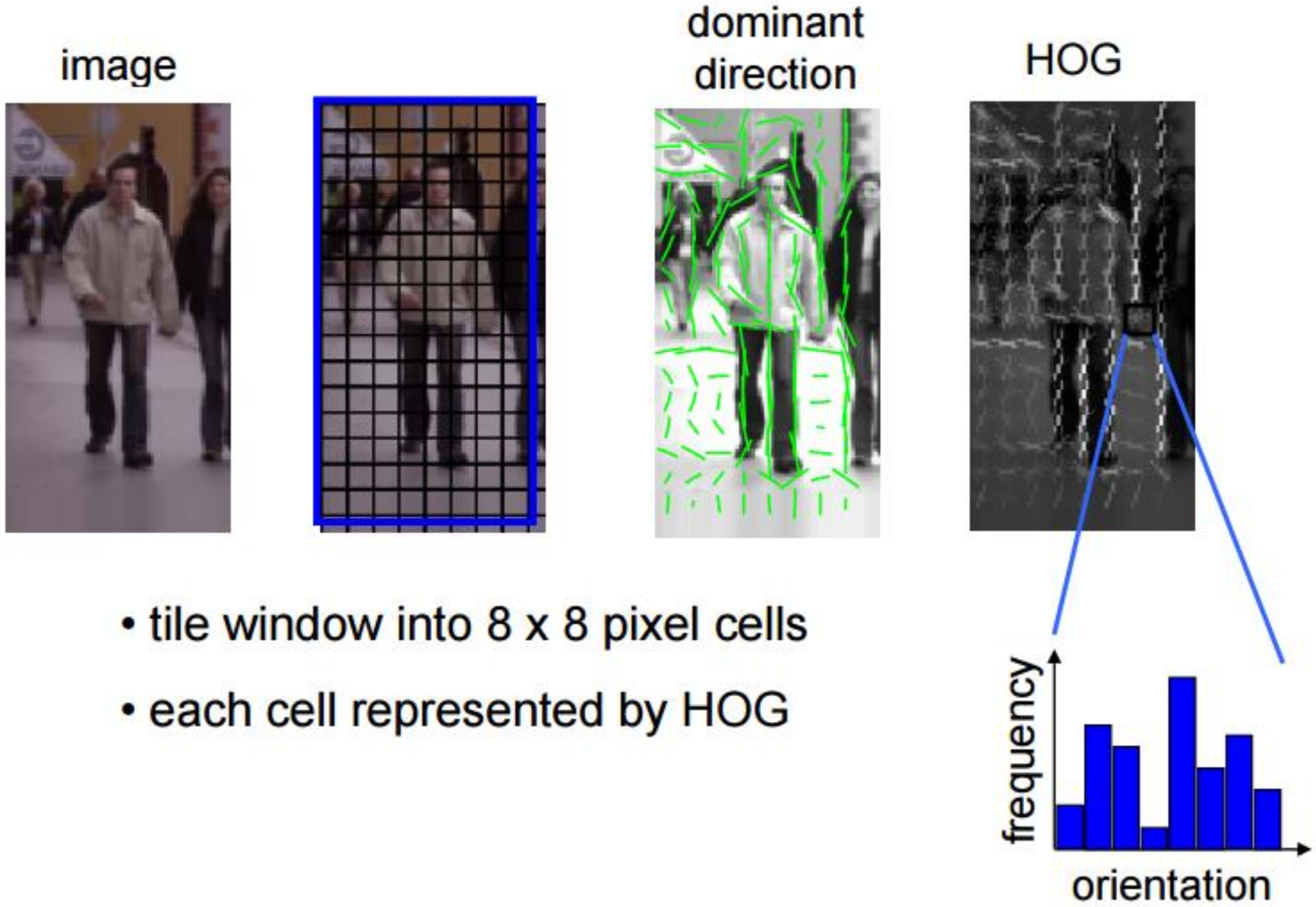
- Positive data – 1208 positive window examples




- Negative data – 1218 negative window examples (initially)

Feature: histogram of oriented gradients (HOG)



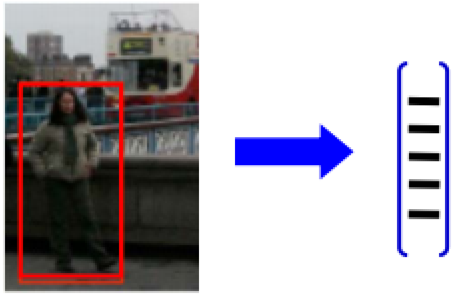
- tile window into 8 x 8 pixel cells
- each cell represented by HOG

Feature vector dimension = 16 x 8 (for tiling) x 8 (orientations) = 1024

Algorithm

Training (Learning)

- Represent each example window by a HOG feature vector



$$\mathbf{x}_i \in \mathbb{R}^d, \text{ with } d = 1024$$

- Train a SVM classifier

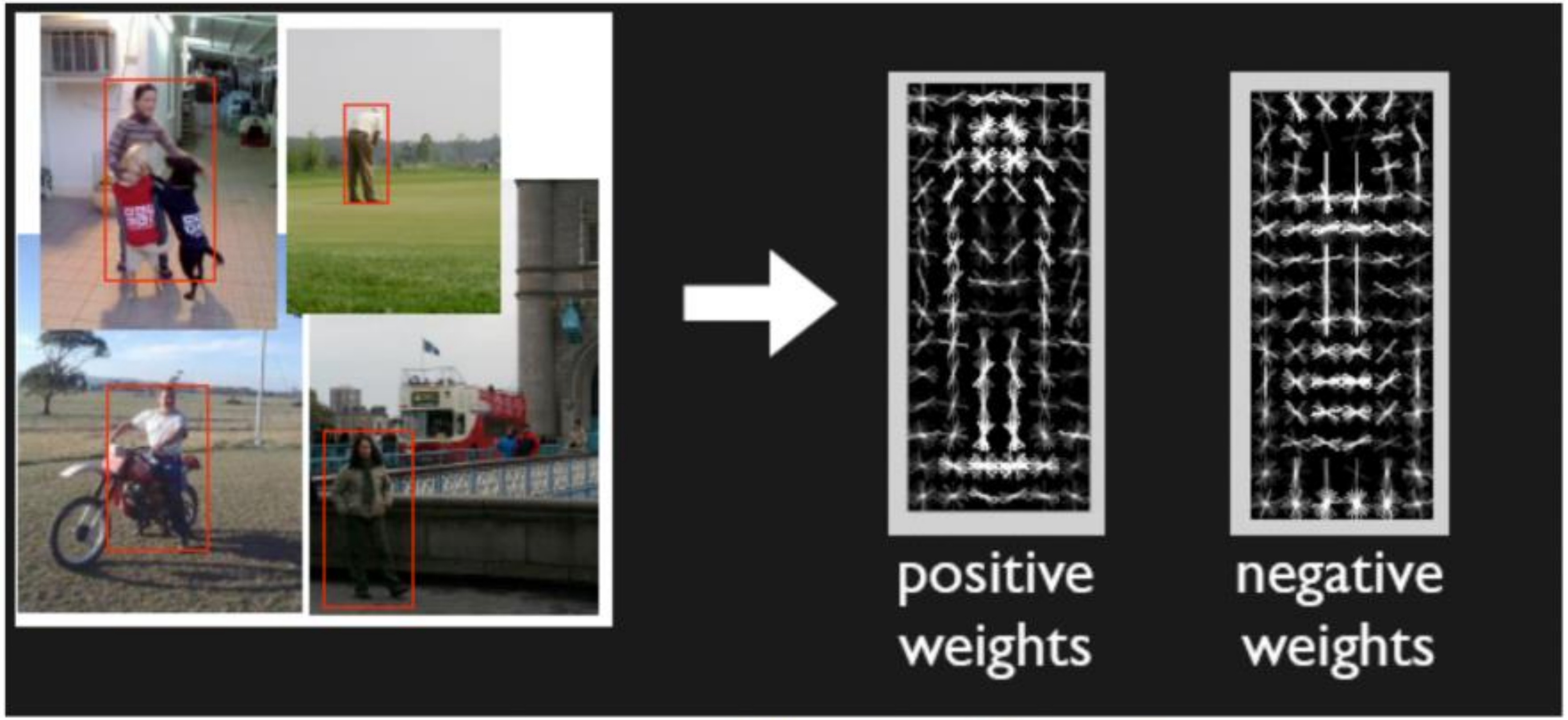
Testing (Detection)

- Sliding window classifier

$$f(x) = \mathbf{w}^\top \mathbf{x} + b$$

Learned model

$$f(\mathbf{x}) = \mathbf{w}^\top \mathbf{x} + b$$



Slide from Deva Ramanan

What do negative weights mean?

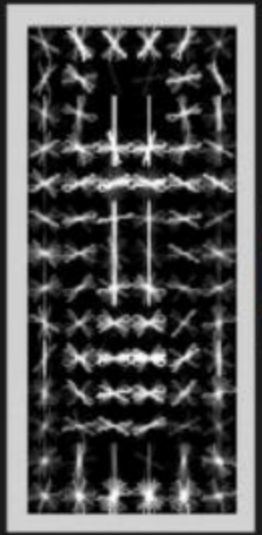
$$wx > 0$$

$$(w_+ - w_-)x > 0$$

$$w_+ > w_- x$$

pedestrian model

>



pedestrian background model

Complete system should compete pedestrian/pillar/doorway models

Discriminative models come equipped with own bg

(avoid firing on doorways by penalizing vertical edges)

Slide from Deva Ramanan

Additional Resources

- **An excellent tutorial on VC-dimension and Support Vector Machines:**

  C.J.C. Burges. A tutorial on support vector machines for pattern recognition. Data Mining and Knowledge Discovery, 2(2):955-974, 1998.

- **The VC/SRM/SVM Bible:**

  Statistical Learning Theory by Vladimir Vapnik, Wiley-Interscience; 1998

http://www.kernel-machines.org/

Reference

- **Support Vector Machine Classification of Microarray Gene Expression Data**, Michael P. S. Brown William Noble Grundy, David Lin, Nello Cristianini, Charles Sugnet, Manuel Ares, Jr., David Haussler
- www.cs.utexas.edu/users/mooney/cs391L/svm.**ppt**
- **Text categorization with Support Vector Machines: learning with many relevant features**

  T. Joachims, ECML - 98