

Support Vector Machines

Mario Martin

CS-UPC

December 8, 2019

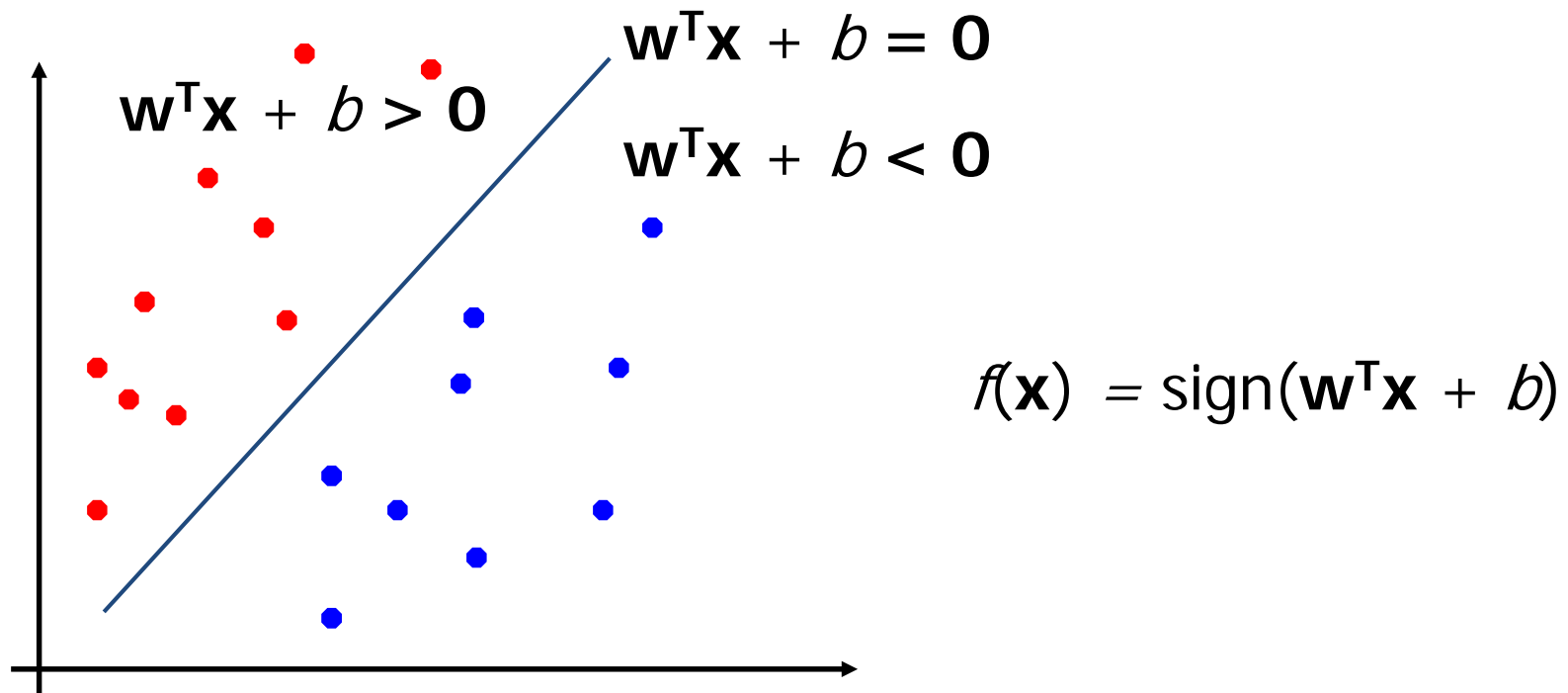
Outline

- Large-margin linear classifier
 - Linear separable
 - Nonlinear separable
- Creating nonlinear classifiers: kernel trick
- Discussion on SVM
- Conclusion

SVM: Large-margin linear classifier

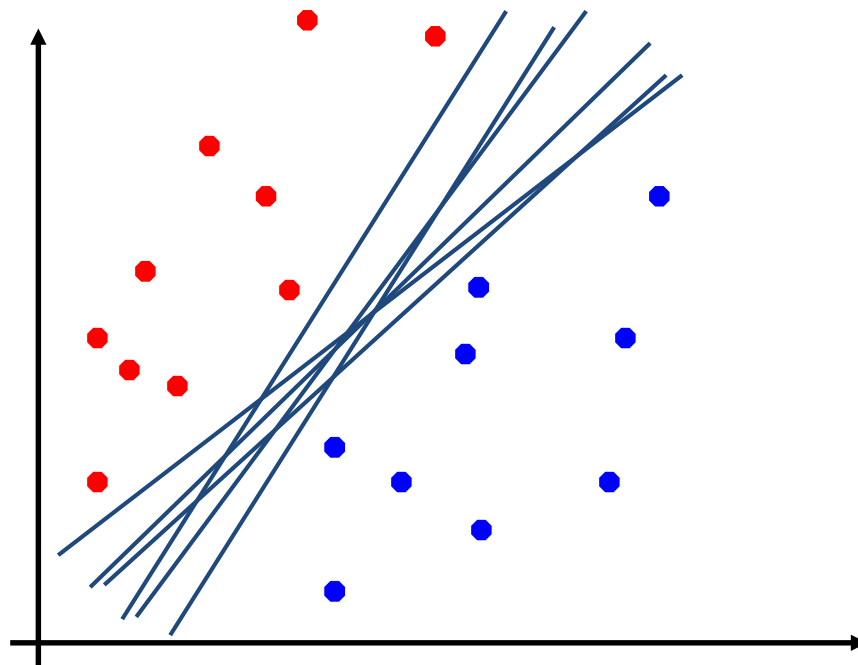
Perceptron Revisited: Linear Separators

Binary classification can be viewed as the task of separating classes in feature space:



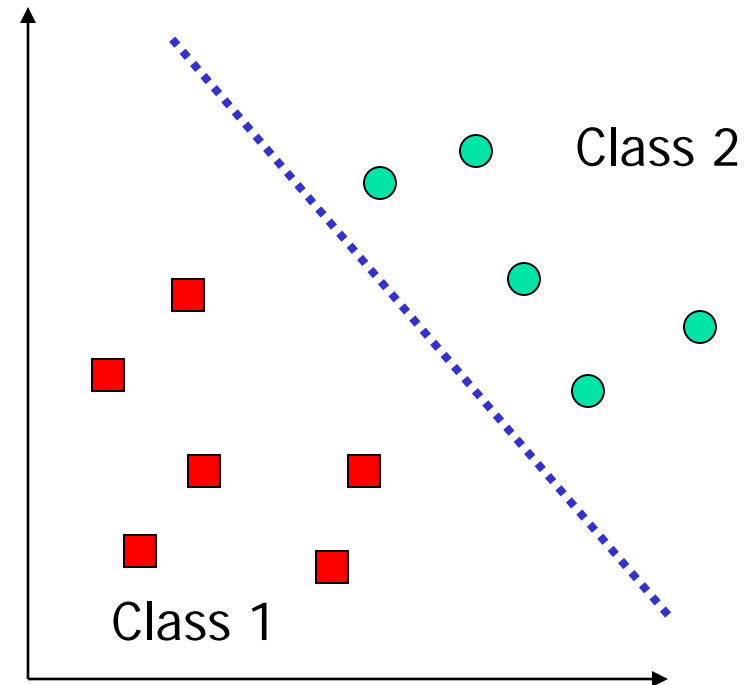
Perceptron Revisited: Linear Separators

There are infinite linear separators. Are all them equally good?

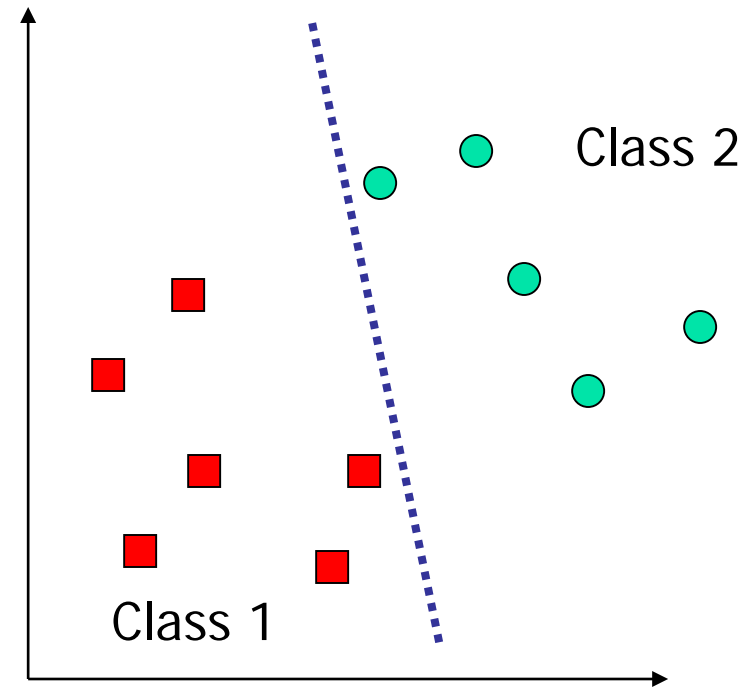
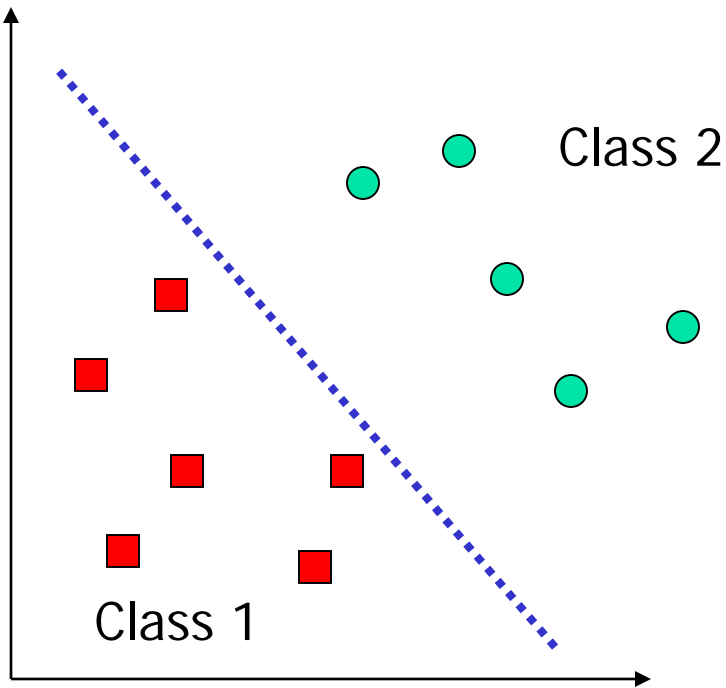


What is a good Decision Boundary?

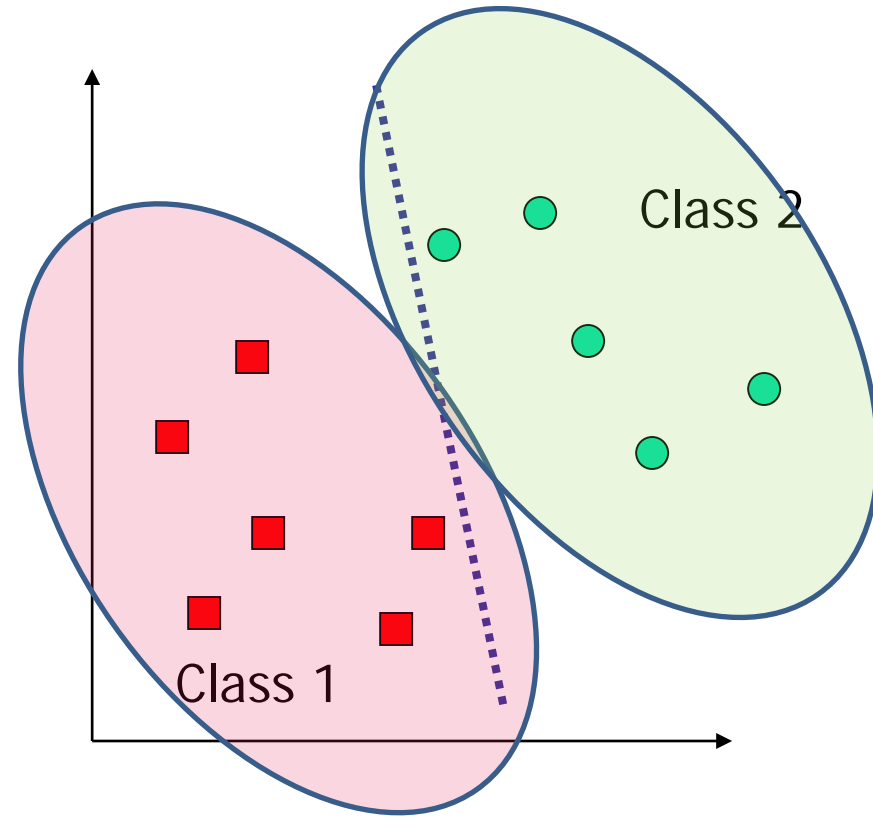
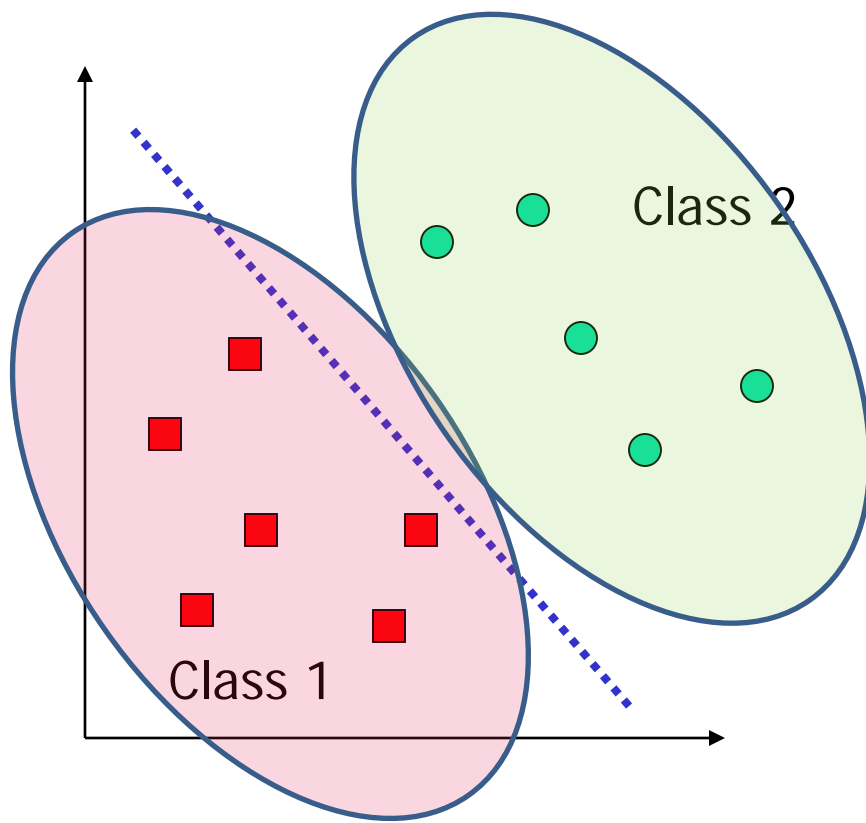
- Consider a two-class, linearly separable classification problem
- Many decision boundaries!
 - The Perceptron algorithm can be used to find such a boundary
 - Different algorithms have been proposed
 - Are all decision boundaries equally good?



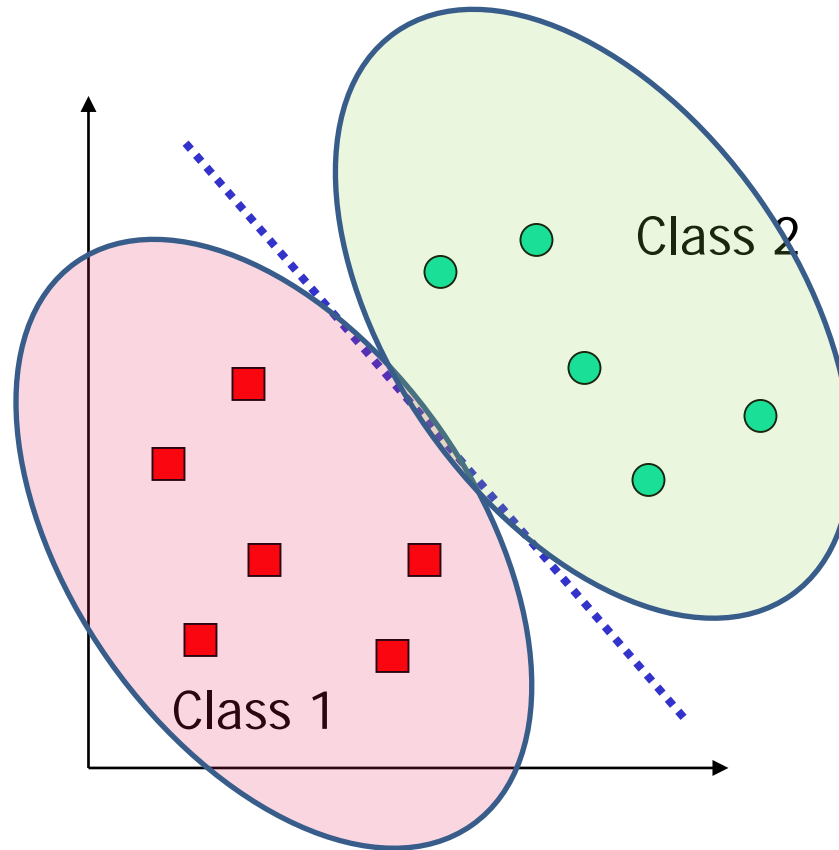
Examples of Bad Decision Boundaries



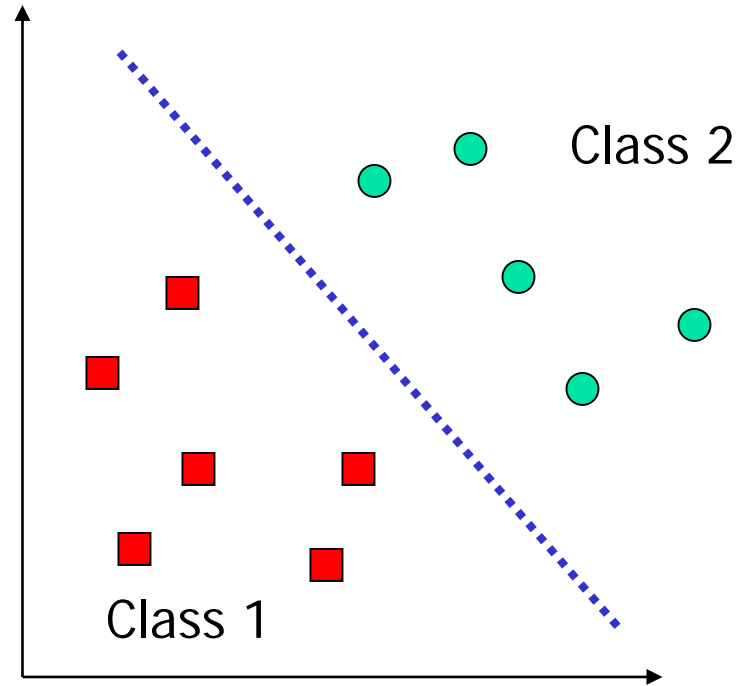
Examples of Bad Decision Boundaries



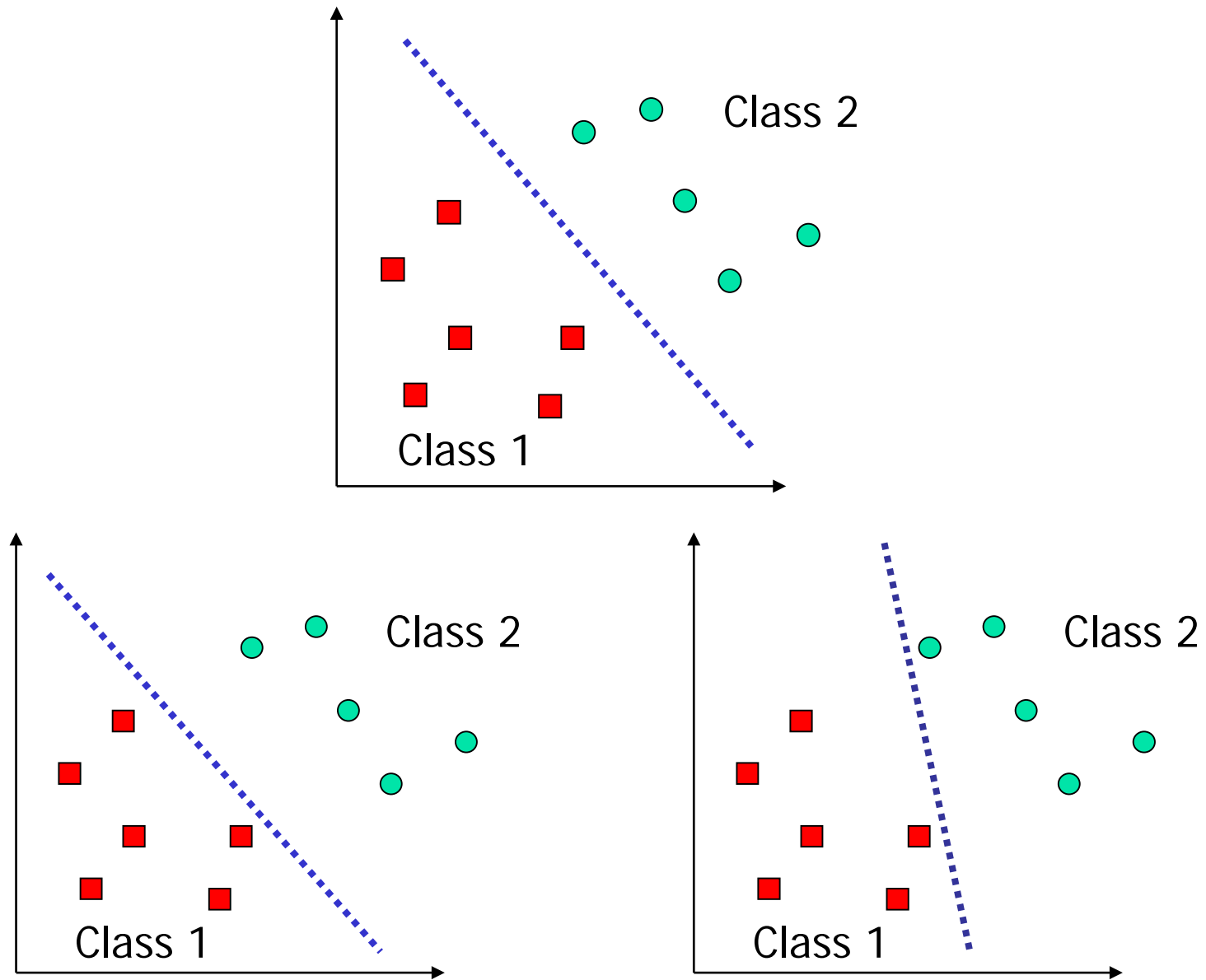
Better Decision Boundary



Better Decision Boundary

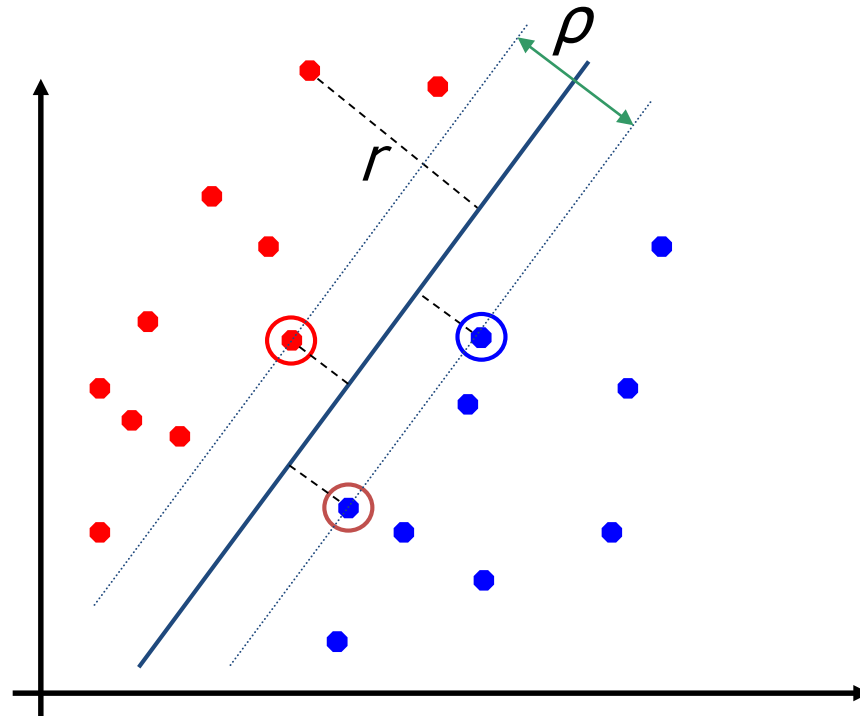


Decision Boundaries



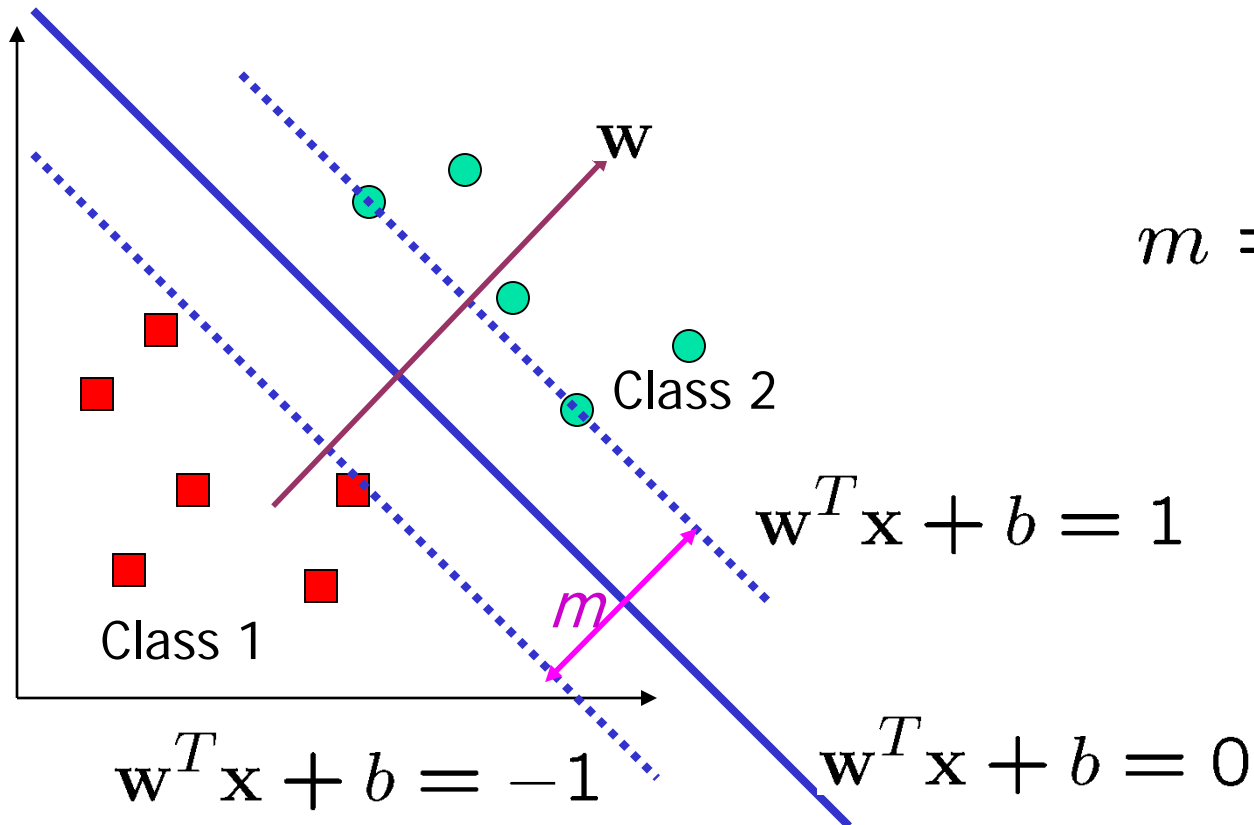
Classification Margin

- Distance from example \mathbf{x}_i to the separator is $r = \frac{\mathbf{w}^T \mathbf{x}_i + b}{\|\mathbf{w}\|}$
- Examples closest to the hyperplane are *support vectors*.
- *Margin* ρ of the separator is the distance between support vectors.



Large-margin Decision Boundary

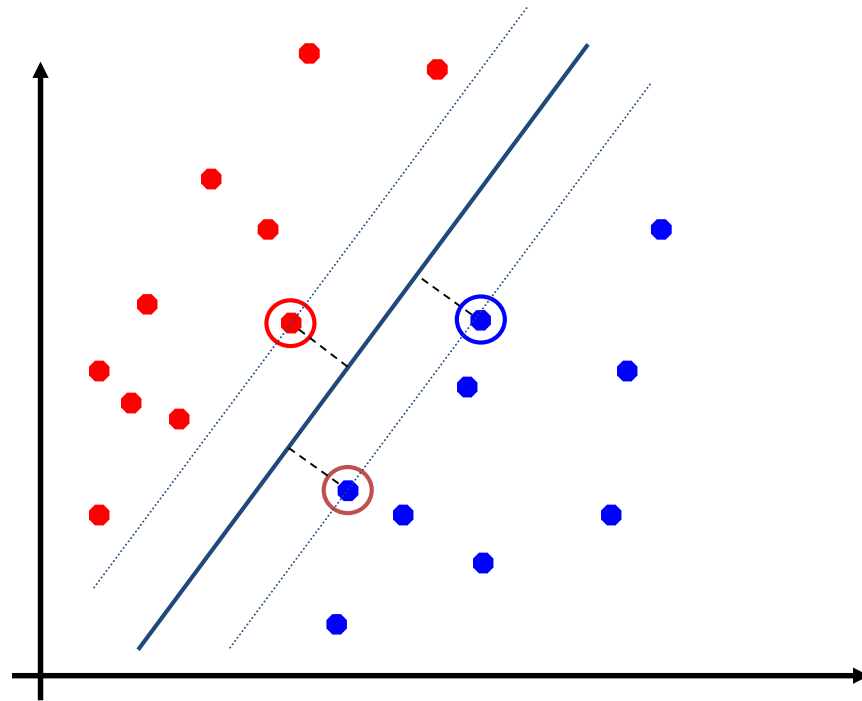
- The decision boundary should be as far away from the data of both classes as possible
 - We should maximize the margin, m
 - Distance between the origin and the line $\mathbf{w}^T \mathbf{x} = k$ is $k / ||\mathbf{w}||$



$$m = \frac{2}{||\mathbf{w}||}$$

Maximum Margin Classification

- Maximizing the margin is good according to intuition and PAC theory.
- Implies that only support vectors matter; other training examples are ignorable.



Finding the Decision Boundary

- Let $\{x_1, \dots, x_n\}$ be our data set and let $y_i \in \{1, -1\}$ be the class label of x_i

- The decision boundary should classify all points correctly

$$y_i(\mathbf{w}^T \mathbf{x}_i + b) \geq 1, \quad \forall i$$

- The decision boundary can be found by solving the following constrained optimization problem

$$\begin{aligned} &\text{Minimize } \frac{1}{2} \|\mathbf{w}\|^2 \\ &\text{subject to } y_i(\mathbf{w}^T \mathbf{x}_i + b) \geq 1 \quad \forall i \end{aligned}$$

- This is a constrained optimization problem. Solving it requires some new tools
 - Feel free to ignore the following several slides; what is important is the constrained optimization problem above

[Recap of Constrained Optimization]

- Suppose we want to: minimize $f(\mathbf{x})$ subject to $g(\mathbf{x}) = 0$
- A necessary condition for \mathbf{x}_0 to be a solution:

$$\begin{cases} \frac{\partial}{\partial \mathbf{x}}(f(\mathbf{x}) + \alpha g(\mathbf{x})) \Big|_{\mathbf{x}=\mathbf{x}_0} = 0 \\ g(\mathbf{x}) = 0 \end{cases}$$

- α : the Lagrange multiplier
- For multiple constraints $g_i(\mathbf{x}) = 0$, $i=1, \dots, m$, we need a Lagrange multiplier α_i for each of the constraints

$$\begin{cases} \frac{\partial}{\partial \mathbf{x}}(f(\mathbf{x}) + \sum_{i=1}^n \alpha_i g_i(\mathbf{x})) \Big|_{\mathbf{x}=\mathbf{x}_0} = 0 \\ g_i(\mathbf{x}) = 0 \quad \text{for } i = 1, \dots, m \end{cases}$$

[Recap of Constrained Optimization]

- The case for inequality constraint $g_i(\mathbf{x}) \leq 0$ is similar, except that the Lagrange multiplier α_i should be positive
- If \mathbf{x}_0 is a solution to the constrained optimization problem

$$\min_{\mathbf{x}} f(\mathbf{x}) \quad \text{subject to} \quad g_i(\mathbf{x}) \leq 0 \quad \text{for } i = 1, \dots, m$$

- There must exist $\alpha_i \geq 0$ for $i=1, \dots, m$ such that \mathbf{x}_0 satisfy

$$\begin{cases} \left. \frac{\partial}{\partial \mathbf{x}} \left(f(\mathbf{x}) + \sum_i \alpha_i g_i(\mathbf{x}) \right) \right|_{\mathbf{x}=\mathbf{x}_0} = \mathbf{0} \\ g_i(\mathbf{x}) \leq 0 \quad \text{for } i = 1, \dots, m \end{cases}$$

- The function $f(\mathbf{x}) + \sum_i \alpha_i g_i(\mathbf{x})$ is also known as the Lagrangian; we want to set its gradient to $\mathbf{0}$

[Back to the Original Problem]

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

$$\text{subject to } 1 - y_i(\mathbf{w}^T \mathbf{x}_i + b) \leq 0 \quad \text{for } i = 1, \dots, n$$

- The Lagrangian is

$$\mathcal{L} = \frac{1}{2} \mathbf{w}^T \mathbf{w} + \sum_{i=1}^n \alpha_i (1 - y_i(\mathbf{w}^T \mathbf{x}_i + b))$$

- Note that $\|\mathbf{w}\|^2 = \mathbf{w}^T \mathbf{w}$

- Setting the gradient of \mathcal{L} w.r.t. \mathbf{w} and b to zero, we have

$$\mathbf{w} + \sum_{i=1}^n \alpha_i (-y_i) \mathbf{x}_i = 0 \quad \Rightarrow \quad \mathbf{w} = \sum_{i=1}^n \alpha_i y_i \mathbf{x}_i$$

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

[The Dual problem]

- If we substitute $\mathbf{w} = \sum_{i=1}^n \alpha_i y_i \mathbf{x}_i$, we have \mathcal{L}

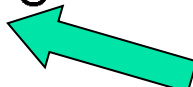

$$\begin{aligned}\mathcal{L} &= \frac{1}{2} \sum_{i=1}^n \alpha_i y_i \mathbf{x}_i^T \sum_{j=1}^n \alpha_j y_j \mathbf{x}_j + \sum_{i=1}^n \alpha_i \left(1 - y_i \left(\sum_{j=1}^n \alpha_j y_j \mathbf{x}_j^T \mathbf{x}_i + b \right) \right) \\ &= \frac{1}{2} \sum_{i=1}^n \sum_{j=1}^n \alpha_i \alpha_j y_i y_j \mathbf{x}_i^T \mathbf{x}_j + \sum_{i=1}^n \alpha_i - \sum_{i=1}^n \alpha_i y_i \sum_{j=1}^n \alpha_j y_j \mathbf{x}_j^T \mathbf{x}_i - b \sum_{i=1}^n \alpha_i y_i \\ &= -\frac{1}{2} \sum_{i=1}^n \sum_{j=1}^n \alpha_i \alpha_j y_i y_j \mathbf{x}_i^T \mathbf{x}_j + \sum_{i=1}^n \alpha_i\end{aligned}$$

- Note that $\sum_{i=1}^n \alpha_i y_i = 0$

- This is a function of α_i only

The Dual problem

- The new objective function is in terms of α_i only
- It is known as the dual problem: if we know \mathbf{w} , we know all α_i ; if we know all α_i , we know \mathbf{w}
- The original problem is known as the primal problem
- The objective function of the dual problem needs to be maximized!
- The dual problem is therefore:

$$\begin{aligned} \max. \quad W(\boldsymbol{\alpha}) &= \sum_{i=1}^n \alpha_i - \frac{1}{2} \sum_{i=1, j=1}^n \alpha_i \alpha_j y_i y_j \mathbf{x}_i^T \mathbf{x}_j \\ \text{subject to } \alpha_i &\geq 0, \quad \sum_{i=1}^n \alpha_i y_i = 0 \end{aligned}$$


Properties of α_i when we introduce the Lagrange multipliers

The result when we differentiate the original Lagrangian w.r.t. b

The Dual problem

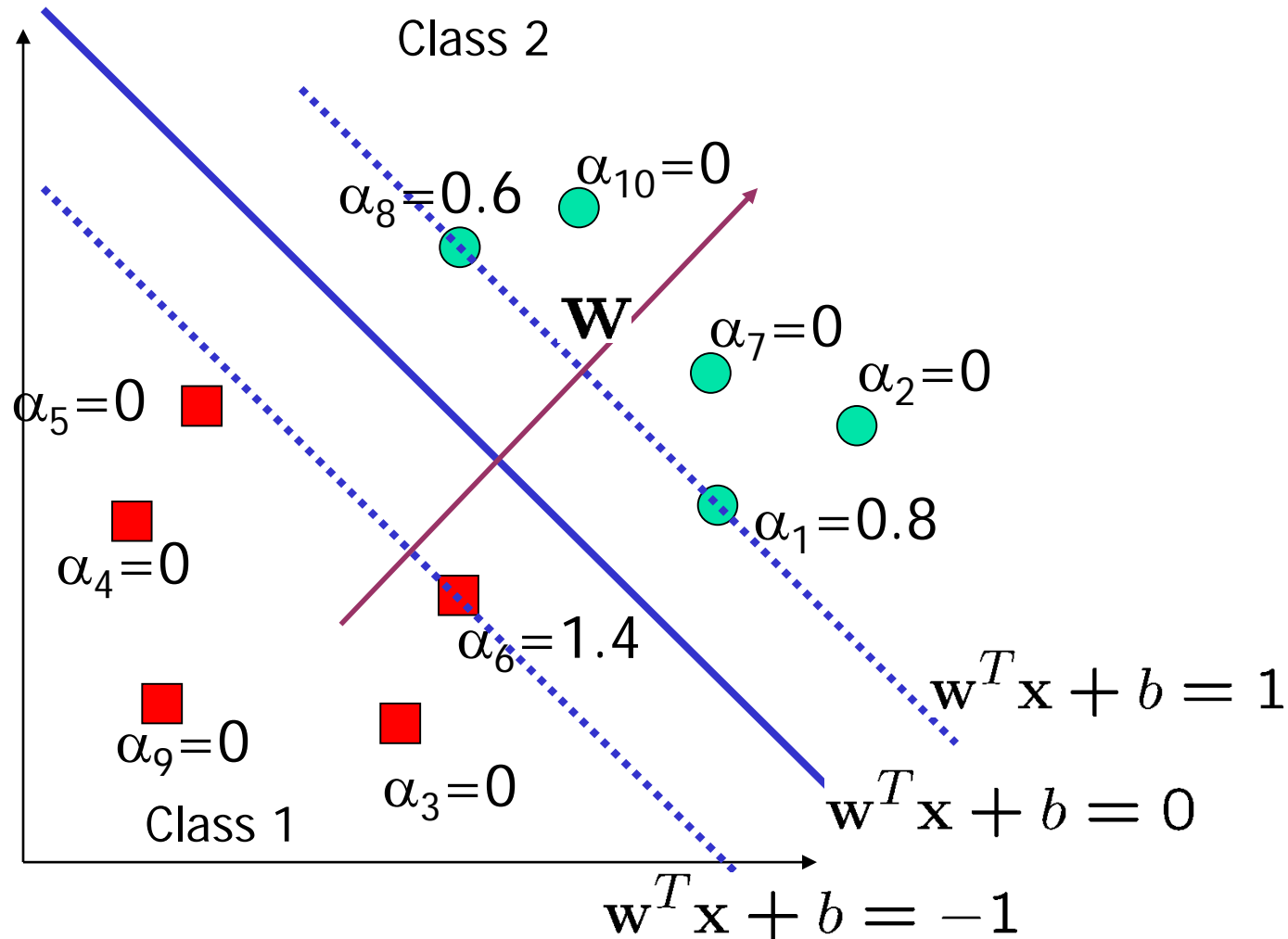
$$\max. \quad W(\boldsymbol{\alpha}) = \sum_{i=1}^n \alpha_i - \frac{1}{2} \sum_{i=1, j=1}^n \alpha_i \alpha_j y_i y_j \mathbf{x}_i^T \mathbf{x}_j$$

$$\text{subject to } \alpha_i \geq 0, \sum_{i=1}^n \alpha_i y_i = 0$$

- This is a quadratic programming (QP) problem
 - A global maximum of α_i can always be found

- \mathbf{w} can be recovered by
$$\mathbf{w} = \sum_{i=1}^n \alpha_i y_i \mathbf{x}_i$$

A Geometrical interpretation



Characteristics of the Solution

- Many of the α_i are zero
 - \mathbf{w} is a linear combination of a small number of data points
 - This “sparse” representation can be viewed as data compression as in the construction of knn classifier
- \mathbf{x}_i with non-zero α_i are called support vectors (SV)
 - The decision boundary is determined only by the SV
 - Let t_j ($j=1, \dots, s$) be the indices of the s support vectors. We can write
$$\mathbf{w} = \sum_{j=1}^s \alpha_{t_j} y_{t_j} \mathbf{x}_{t_j}$$
- For testing with a new data \mathbf{z}
 - Compute $\mathbf{w}^T \mathbf{z} + b = \sum_{j=1}^s \alpha_{t_j} y_{t_j} (\mathbf{x}_{t_j}^T \mathbf{z}) + b$ classify \mathbf{z} as class 1 if the sum is positive, and class 2 otherwise
 - Note: \mathbf{w} need not be formed explicitly

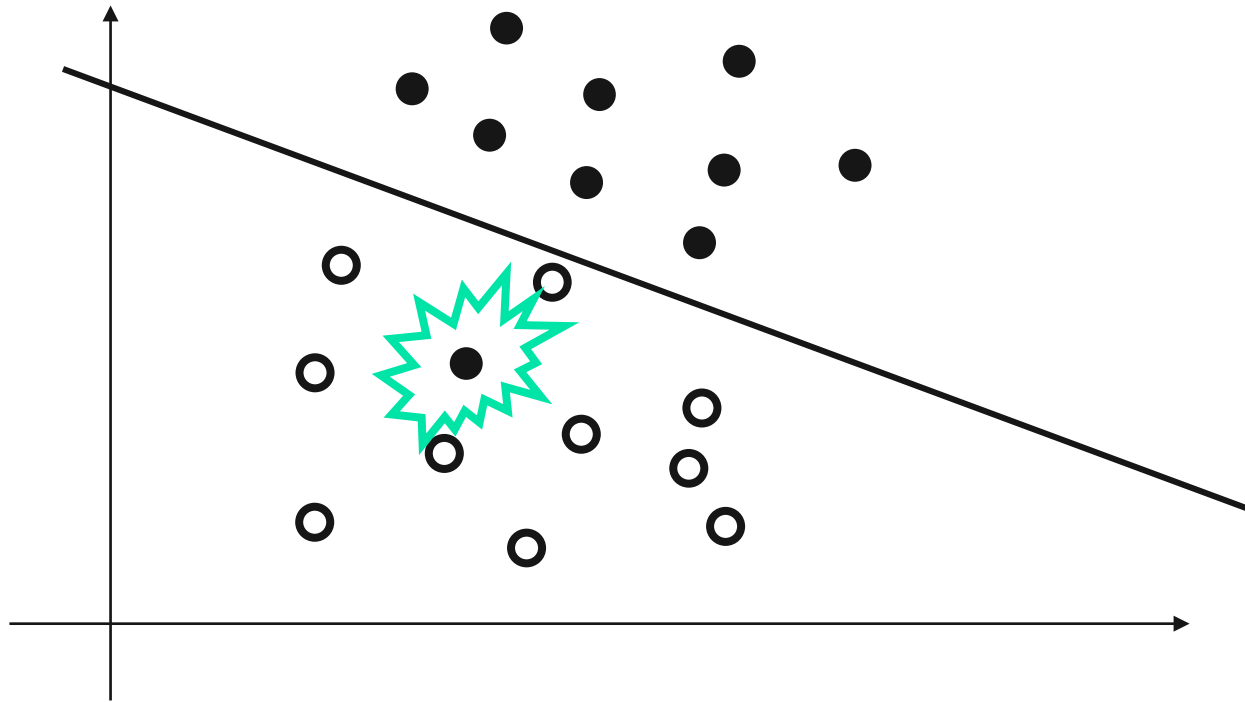
The Quadratic Programming Problem

- Many approaches have been proposed
 - Loqo, cplex, etc. (see <http://www.numerical.rl.ac.uk/qp/qp.html>)
- Most are “interior-point” methods
 - Start with an initial solution that can violate the constraints
 - Improve this solution by optimizing the objective function and/or reducing the amount of constraint violation
- For SVM, sequential minimal optimization (SMO) seems to be the most popular
 - A QP with two variables is trivial to solve
 - Each iteration of SMO picks a pair of (α_i, α_j) and solve the QP with these two variables; repeat until convergence
- In practice, we can just regard the QP solver as a “black-box” without bothering how it works

Non-linear separable datasets: Soft-Margin SVM

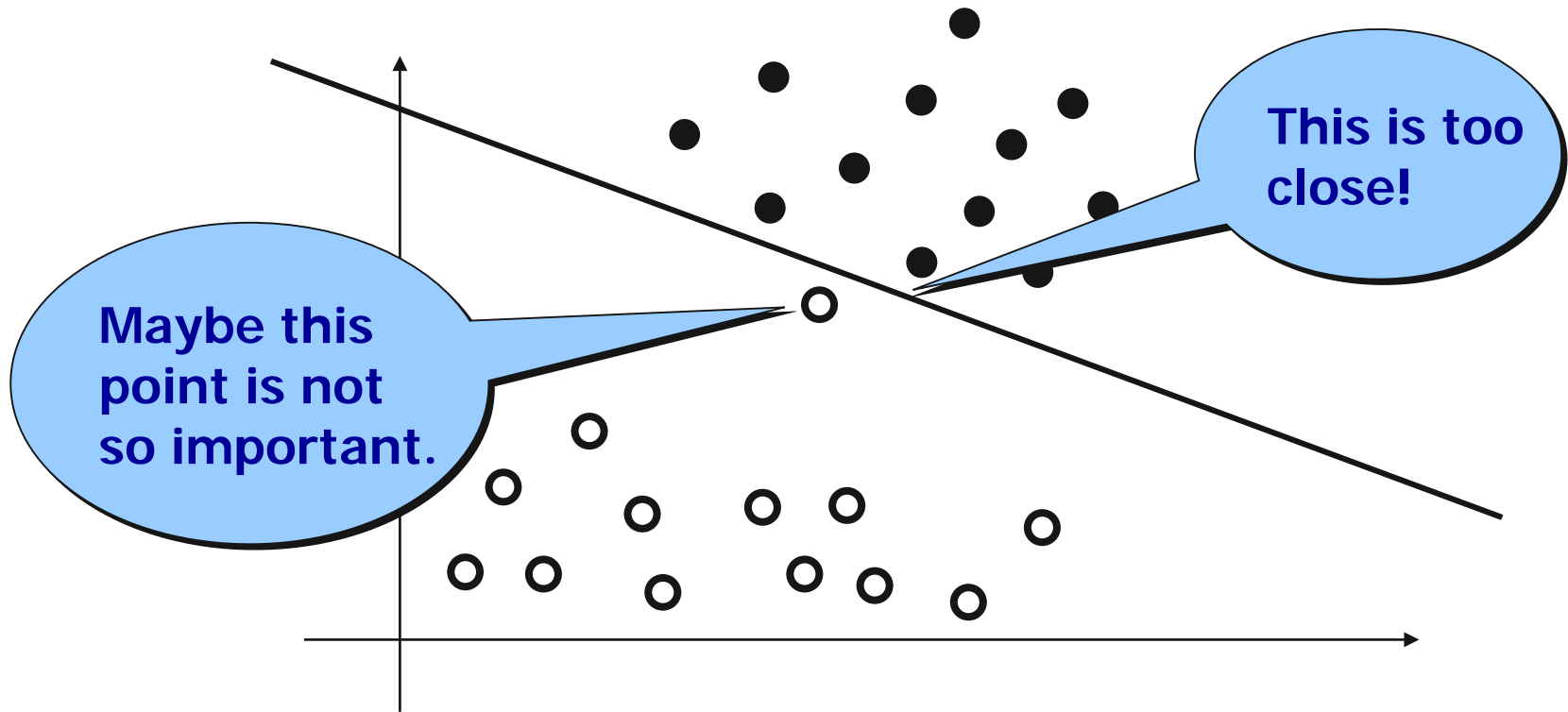
Non-Separable Sets

- Sometimes, data sets are not linearly separable.



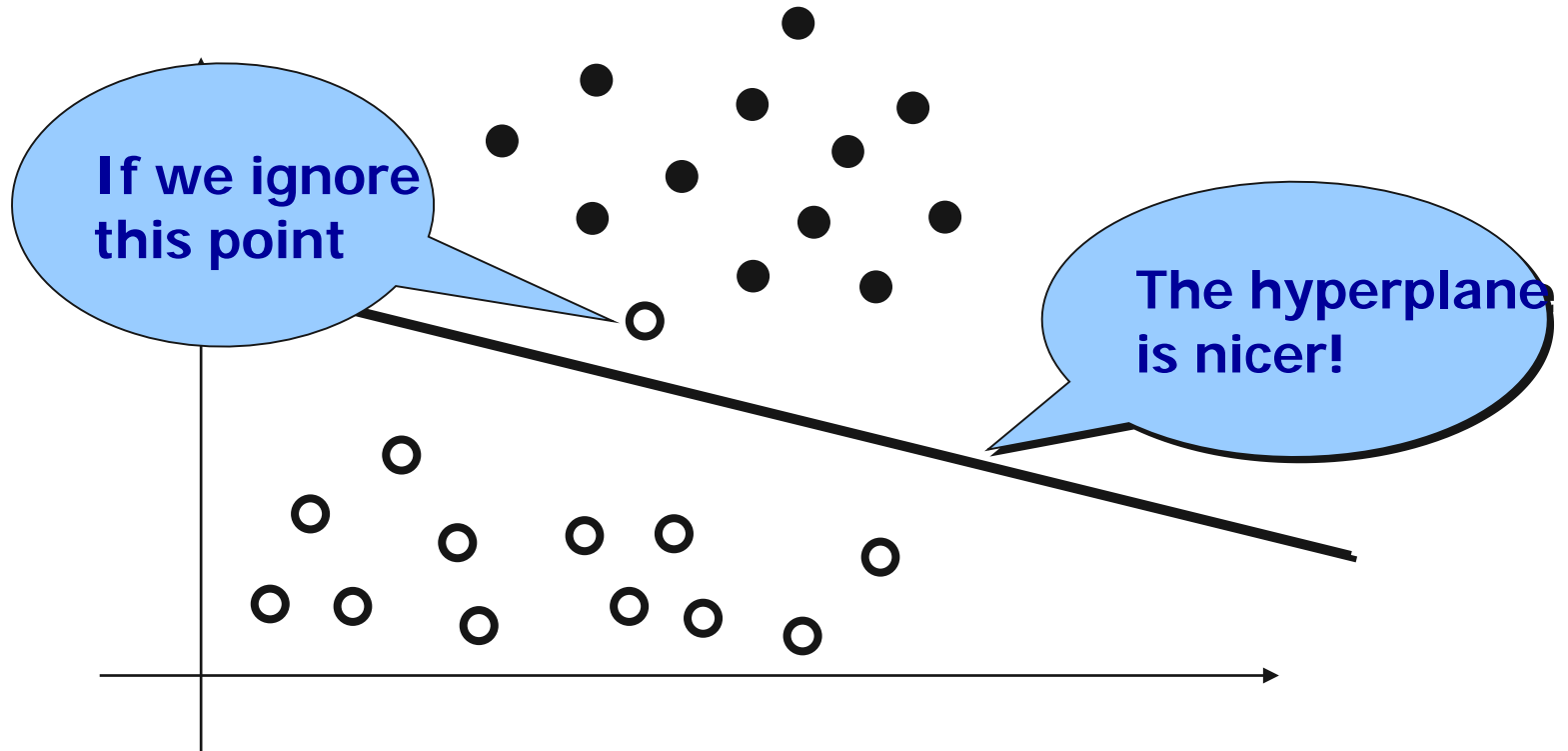
Non-Separable Sets

- Sometimes, we **do not want** to separate perfectly.



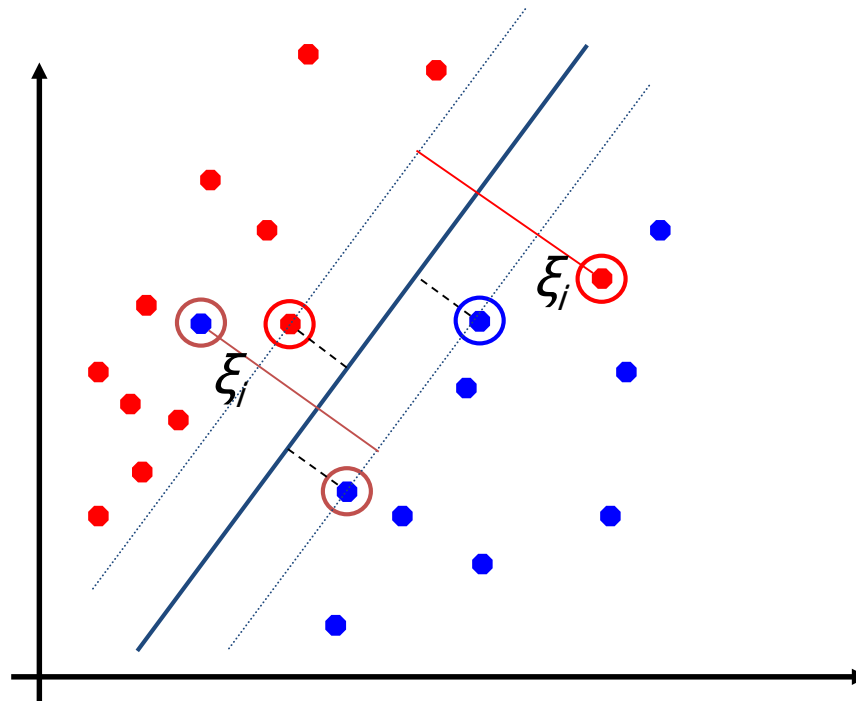
Non-Separable Sets

- Sometimes, we **do not want** to separate perfectly.



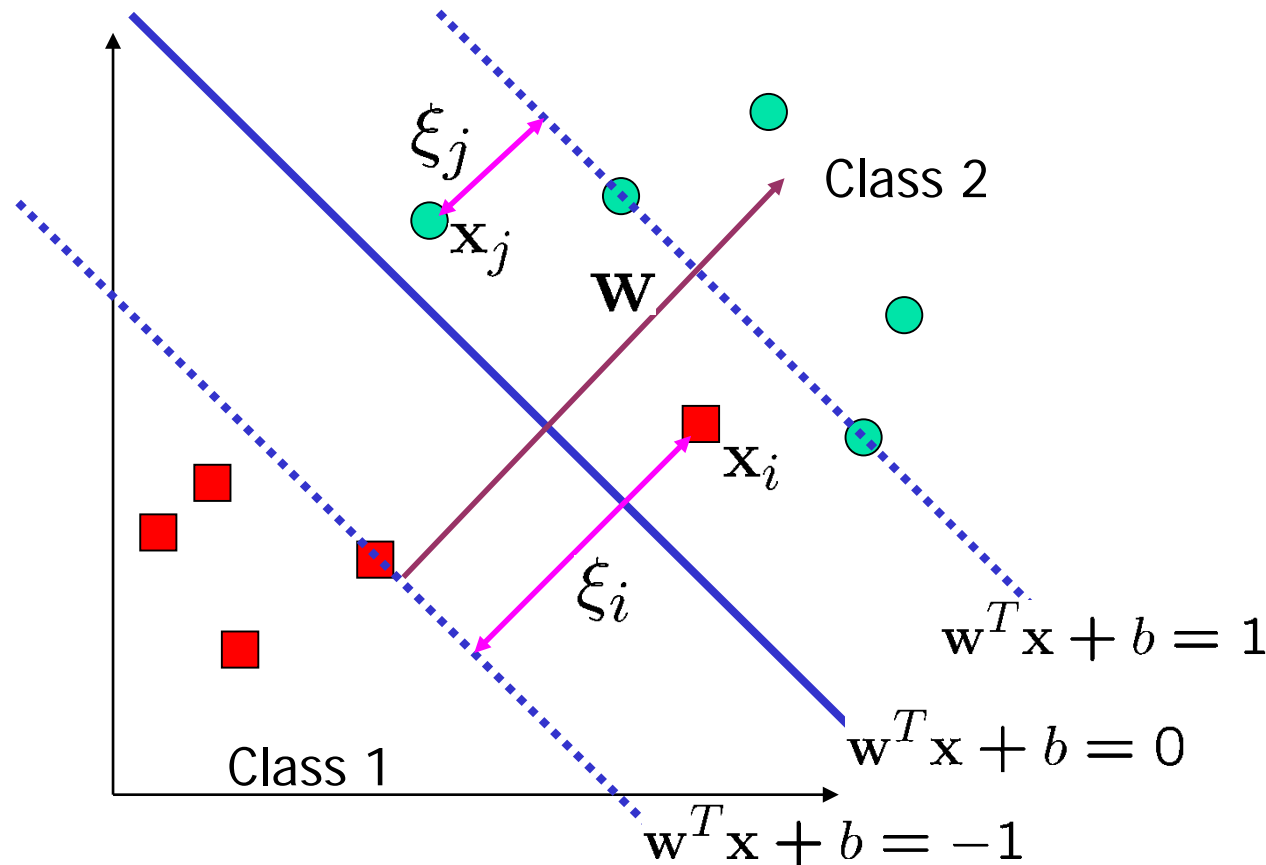
Soft Margin Classification

- *Slack variables* ξ_i can be added to allow misclassification of difficult or noisy examples, resulting margin called *soft*.



Soft Margin Classification

- We allow “error” ξ_i in classification; it is based on the output of the discriminant function $\mathbf{w}^T \mathbf{x} + b$
- ξ_i different from 0 for misclassified samples



Soft Margin Classification

- If we minimize $\sum_i \xi_i$, ξ_i can be computed by

$$\begin{cases} \mathbf{w}^T \mathbf{x}_i + b \geq 1 - \xi_i & y_i = 1 \\ \mathbf{w}^T \mathbf{x}_i + b \leq -1 + \xi_i & y_i = -1 \\ \xi_i \geq 0 & \forall i \end{cases}$$

- ξ_i are “slack variables” in optimization
- Note that $\xi_i=0$ if there is no error for \mathbf{x}_i
- ξ_i is an upper bound of the number of errors
- We want to minimize $\frac{1}{2} \|\mathbf{w}\|^2 + C \sum_{i=1}^n \xi_i$
 - C : tradeoff parameter between error and margin

- The optimization problem becomes

$$\begin{aligned} &\text{Minimize } \frac{1}{2} \|\mathbf{w}\|^2 + C \sum_{i=1}^n \xi_i \\ &\text{subject to } y_i(\mathbf{w}^T \mathbf{x}_i + b) \geq 1 - \xi_i, \quad \xi_i \geq 0 \end{aligned}$$

The Optimization Problem

- The dual of this new constrained optimization problem is

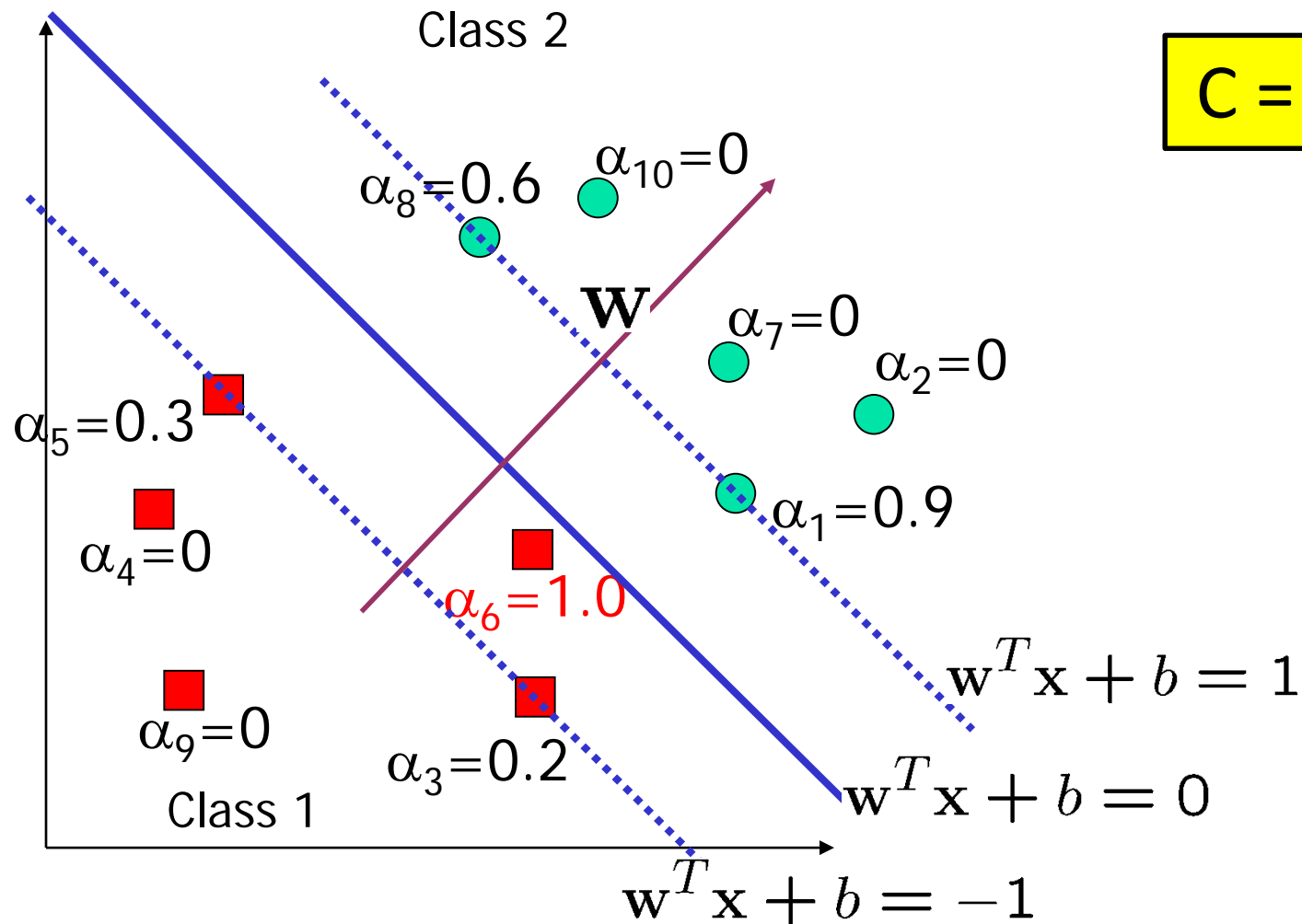
$$\max. W(\alpha) = \sum_{i=1}^n \alpha_i - \frac{1}{2} \sum_{i=1, j=1}^n \alpha_i \alpha_j y_i y_j \mathbf{x}_i^T \mathbf{x}_j$$

$$\text{subject to } C \geq \alpha_i \geq 0, \sum_{i=1}^n \alpha_i y_i = 0$$

- \mathbf{w} is recovered as $\mathbf{w} = \sum_{j=1}^s \alpha_{t_j} y_{t_j} \mathbf{x}_{t_j}$

- This is very similar to the optimization problem in the linear separable case, except that there is an upper bound C on α_i now
- Once again, a QP solver can be used to find α_i

A Geometrical interpretation



$$C = 1.0$$

Importance of support set

- Supports are points in the frontier between classes (supports + errors)

- Solution can be reconstructed from only supports

$$\mathbf{w} = \sum_{j=1}^n \alpha_{t_j} y_{t_j} \mathbf{x}_{t_j} = \sum_{j=1}^s \alpha_{t_j} y_{t_j} \mathbf{x}_{t_j}$$

- Number of supports is usually smaller than the input dimension
- Number of supports is upper bound of *Leave-one-out error*

$$E_{LOO} \leq \|S\|$$

... because using non-support points for testing will not change the boundary and it will be correctly classified

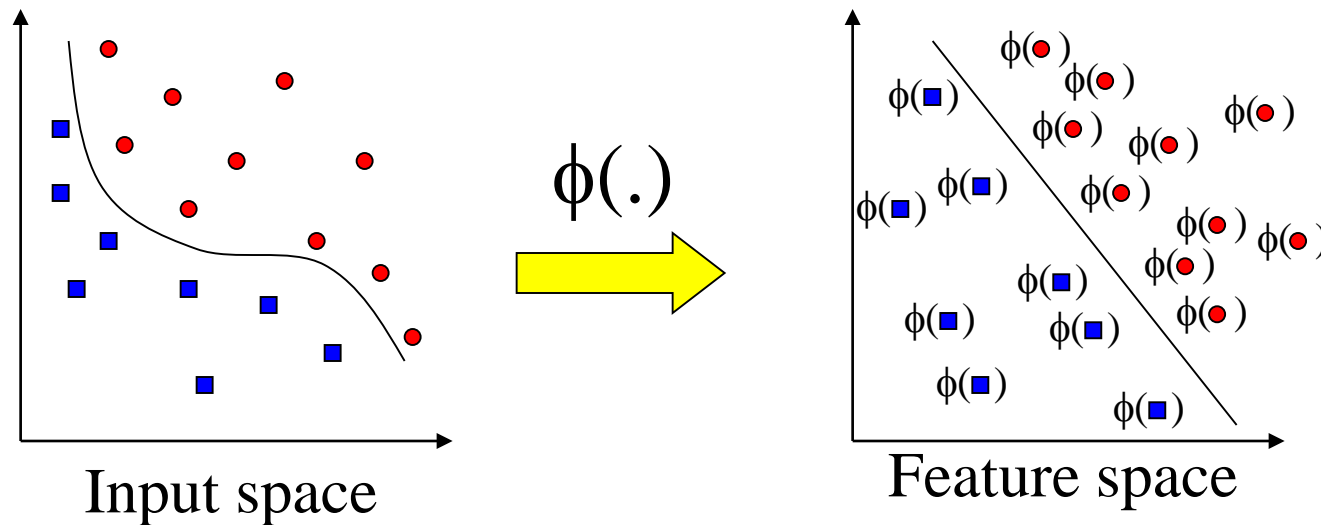
Non-linear separable datasets: Kernel methods

Extension to Non-linear Decision Boundary

- So far, we have only considered large-margin classifier with a linear decision boundary
- How to generalize it to become nonlinear?
- Key idea: transform \mathbf{x}_i to a higher dimensional space to “make life easier”
 - Input space: the space the point \mathbf{x}_i are located
 - Feature space: the space of $\phi(\mathbf{x}_i)$ after transformation
- Why transform?
 - Linear operation in the feature space is equivalent to non-linear operation in input space
 - Classification can become easier with a proper transformation. In the XOR problem, for example, adding a new feature of x_1x_2 make the problem linearly separable

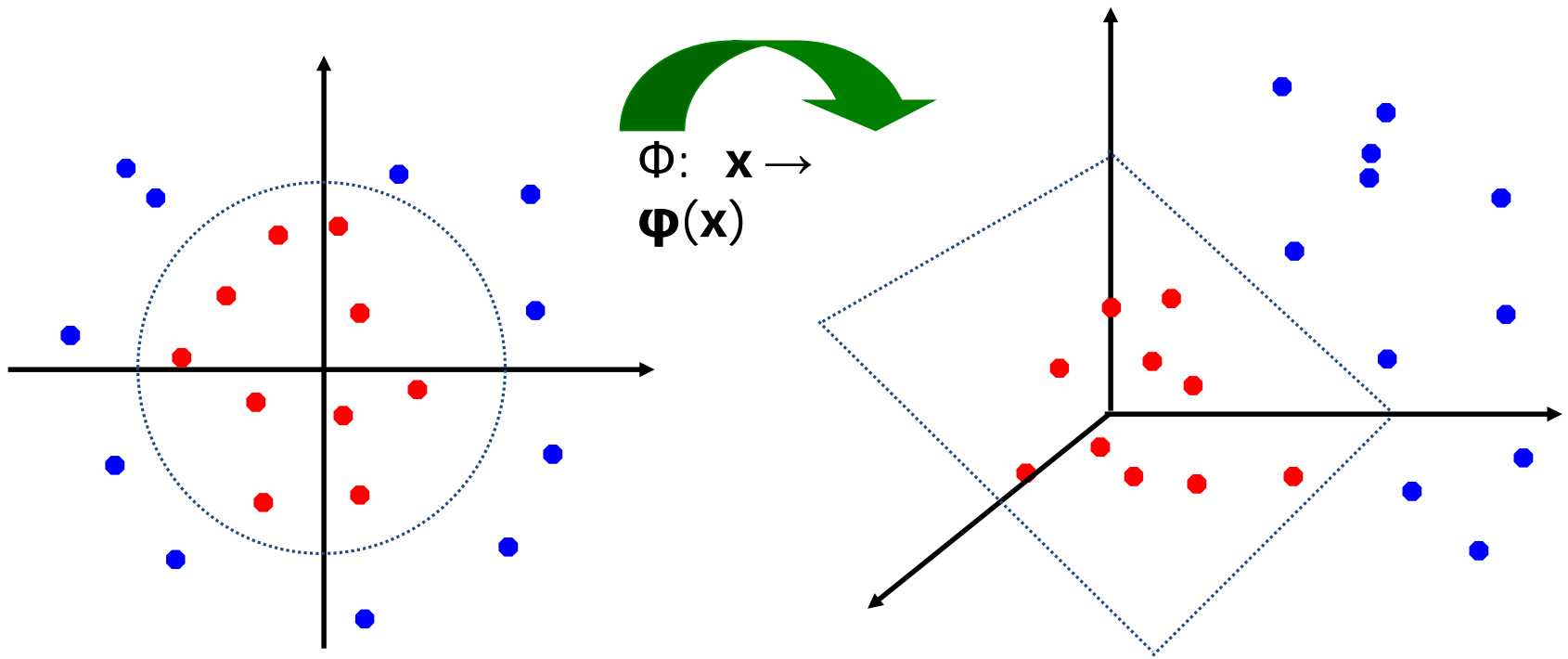
Moving data to higher dimensional space

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



Note: feature space is of higher dimension than the input space in practice

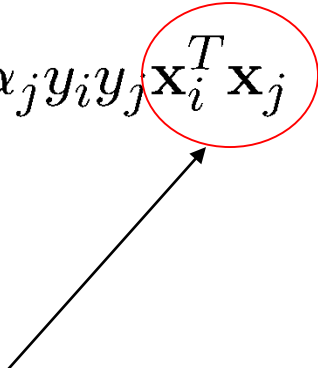
Moving data to higher dimensional space



- Computation in the feature space can be costly because it is high dimensional (feature space can be even infinite-dimensional!)
- The kernel trick comes to rescue

The Kernel Trick

- Recall the SVM optimization problem

$$\begin{aligned} \max. \quad W(\alpha) &= \sum_{i=1}^n \alpha_i - \frac{1}{2} \sum_{i=1, j=1}^n \alpha_i \alpha_j y_i y_j \mathbf{x}_i^T \mathbf{x}_j \\ \text{subject to } C &\geq \alpha_i \geq 0, \quad \sum_{i=1}^n \alpha_i y_i = 0 \end{aligned}$$


- The data points only appear as **inner product**
- **As long as we can calculate the inner product in the feature space, we do not need the mapping explicitly**
- Define the kernel function K by

$$K(\mathbf{x}_i, \mathbf{x}_j) = \phi(\mathbf{x}_i)^T \phi(\mathbf{x}_j)$$

The Kernel Trick

- Recall the SVM optimization problem

$$\begin{aligned} \max. \quad W(\boldsymbol{\alpha}) &= \sum_{i=1}^n \alpha_i - \frac{1}{2} \sum_{i=1, j=1}^n \alpha_i \alpha_j y_i y_j K(\mathbf{x}_i, \mathbf{x}_j) \\ \text{subject to } C &\geq \alpha_i \geq 0, \quad \sum_{i=1}^n \alpha_i y_i = 0 \end{aligned}$$

- Classification

$$h(\mathbf{x}) = \text{sign} \left(\sum_{i=1}^l \alpha_i \cdot y_i \cdot K(\mathbf{x}_i, \mathbf{x}) + b \right)$$

Example: Polynomial kernel

- Suppose $\phi(\cdot)$ is given as follows

$$\phi\left(\begin{bmatrix} x_1 \\ x_2 \end{bmatrix}\right) = (1, \sqrt{2}x_1, \sqrt{2}x_2, x_1^2, x_2^2, \sqrt{2}x_1x_2)$$

- The inner product in the feature space is

$$\begin{aligned} \langle \phi\left(\begin{bmatrix} x_1 \\ x_2 \end{bmatrix}\right), \phi\left(\begin{bmatrix} y_1 \\ y_2 \end{bmatrix}\right) \rangle &= (1, \sqrt{2}x_1, \sqrt{2}x_2, x_1^2, x_2^2, \sqrt{2}x_1x_2)^T (1, \sqrt{2}y_1, \sqrt{2}y_2, y_1^2, y_2^2, \sqrt{2}y_1y_2) \\ &= \dots \\ &= (1 + x_1y_1 + x_2y_2)^2 \end{aligned}$$

- So, if we define the kernel function as follows, there is no need to carry out $\phi(\cdot)$ explicitly

$$K(\mathbf{x}, \mathbf{y}) = (1 + x_1y_1 + x_2y_2)^2$$

- This use of kernel function to avoid carrying out $\phi(\cdot)$ explicitly is known as the **kernel trick**

Popular kernels

- Polynomial kernel **with degree d**

$$K(\mathbf{x}, \mathbf{y}) = (\mathbf{x}^T \mathbf{y} + 1)^d$$

- Radial basis function kernel **with width σ**

$$K(\mathbf{x}, \mathbf{y}) = \exp(-\|\mathbf{x} - \mathbf{y}\|^2 / (2\sigma^2))$$

- The feature space is infinite-dimensional
- The projection function is unknown
- ?

Kernel conditions

- All kernels has the following form

$$K(\mathbf{x}_i, \mathbf{x}_j) = \phi(\mathbf{x}_i)^T \phi(\mathbf{x}_j) = N^T N$$

- Any matrix that can be decomposed as $N^T N$ is called as **symmetric, positive definite matrix (sdp)**
- Any function $K(x,z)$ that creates a **symmetric, positive definite matrix** **is a valid kernel** (= an inner product in some space)
- ...even when we don't know projection function $\phi(\cdot)$
- This is the case of the RBF function

Choosing the Kernel Function

- Probably the most tricky part of using SVM.
- The kernel function is important because it creates the kernel matrix, which summarizes all the data
- Many principles have been proposed (diffusion kernel, Fisher kernel, string kernel, ...)
- Since the training of the SVM only needs the value of $K(x_i, x_j)$ there is no constraints about how the examples are represented
- **In practice, a low degree polynomial kernel or RBF kernel with a reasonable width is a good initial try**

Summary: Steps for Classification

- Prepare the data matrix [**numeric+normalization**]
- Select the kernel function to use
- Select the parameter of the kernel function and the value of C
 - You can use the values suggested by the SVM software, or you can set apart a validation set to determine the values of the parameter
- Execute the training algorithm and obtain the α_i
- Unseen data can be classified using the α_i and the support vectors

Strengths and Weaknesses of SVM

■ Strengths

- Training is relatively easy
 - No local optimal, unlike in neural networks
- It scales relatively well to high dimensional data
- Tradeoff between classifier complexity and error can be controlled explicitly
- Non-traditional data like strings and trees can be used as input to SVM, instead of feature vectors

■ Weaknesses

- Need to choose a “good” kernel function.

Other Types of Kernel Methods

- A lesson learnt in SVM: a linear algorithm in the feature space is equivalent to a non-linear algorithm in the input space
- Standard linear algorithms can be generalized to its non-linear version by going to the feature space
 - Kernel principal component analysis, kernel independent component analysis, kernel canonical correlation analysis, kernel k-means, 1-class SVM are some examples

Conclusion

- SVM state of the art classification algorithms
- Two key concepts of SVM: maximize the margin and the kernel trick
- Many SVM implementations are available on the web for you to try on your data set!
- Let's play!
 - www.csie.ntu.edu.tw/~cjlin/libsvm