Support Vector Machine

生醫光電所 吳育德

Chap2, Support Vector Machines for Pattern Classification, Shigeo Abe, 2005
Chap5. A First Course in Machine Learning, 2ed, Simon Rogers and Mark Girolami, 2017

Hard-Margin Support Vector Machines

- Let N d-dimensional training inputs $\boldsymbol{x}_i$ (i = 1, . . . , N) belong to Class 1 or 2 and the labels be $y_i = 1$ for Class 1 and $y_i = -1$ for Class 2.

- If data are linearly separable, we can determine the decision function:

$$D(x) = \mathbf{w}^T \boldsymbol{x} - b$$

where **w** is an d-dimensional vector, b is a bias term, i = 1, . . . , N

$$\mathbf{w}^T \boldsymbol{x_i} - b \begin{cases} > 0 & \text{for } y_i = 1, \\ < 0 & \text{for } y_i = -1 \end{cases} \quad ①$$

- Because the training data are linearly separable, no training data satisfy $\mathbf{w}^T \boldsymbol{x} - b = 0$

Hard-Margin Support Vector Machines

- To control separability, instead of ①, we consider

$$\mathbf{w}^T \boldsymbol{x_i} - b \begin{cases} > 1 & \text{for } y_i = 1, \\ < -1 & \text{for } y_i = -1 \end{cases} \quad ②$$

Here, 1 and -1 can be replaced by a constant a (> 0) and $-a$.

- ② is equivalent to

$$y_i(\mathbf{w}^T \boldsymbol{x_i} - b) \geq 1, i = 1, \ldots, N$$

- The hyperplane $D(x) = \mathbf{w}^T \boldsymbol{x} - b = \mathrm{c}$ for $-1 < c < 1$

forms a separating hyperplane that separates x_i ($i = 1, \ldots, N$) .

- When $c = 0$, the separating hyperplane is in the middle of the two hyperplanes with $c = 1$ and -1.

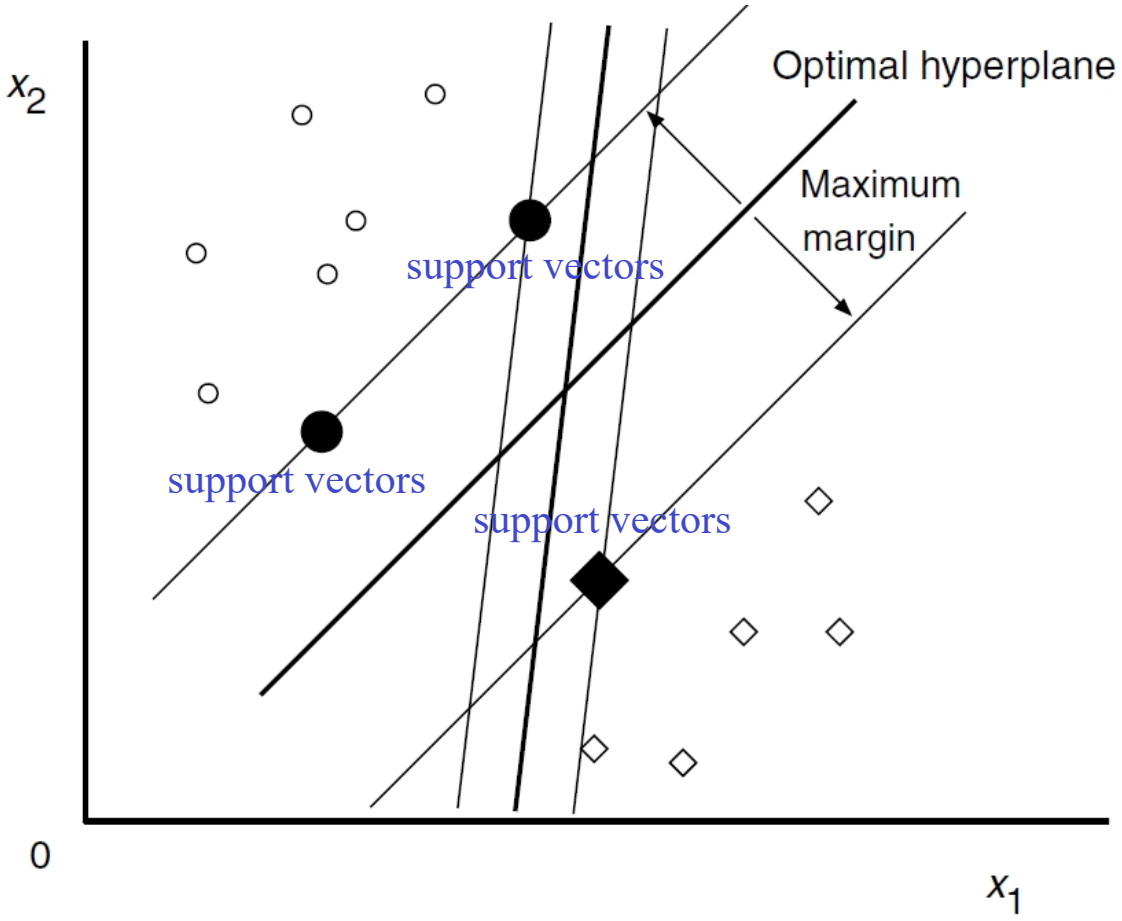
Hard-Margin Support Vector Machines

- The distance between the separating hyperplane and the training datum nearest to the hyperplane is called the ***margin***

- The hyperplane with the maximum margin is called the **optimal separating hyperplane**

- The margin is a function of **w**. Training the SVM consists of learning a **w** that maximizes the margin. So, margin is important.

Optimal separating hyperplane in a two-dimensional space



Normal distance between x and the hyperplane

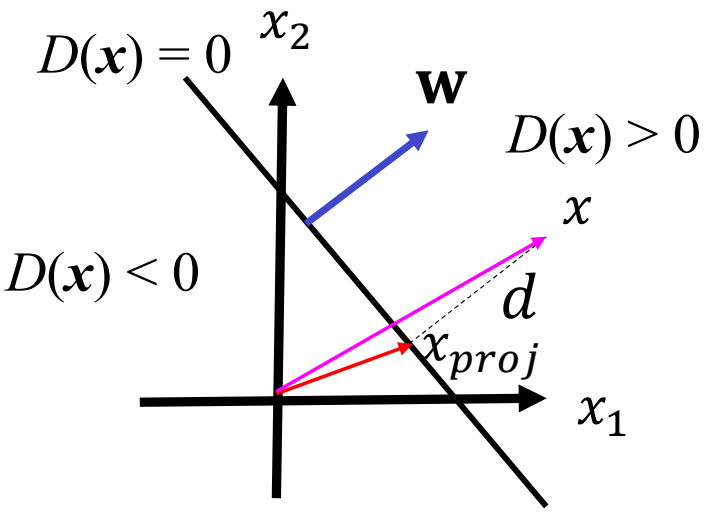
- x_{proj}: projection of x onto the hyperplane $D(\boldsymbol{x}) = 0$.
- $\boldsymbol{d}$: the normal distance between x and x_{proj}.
- $\boldsymbol{x} = \boldsymbol{x}_{proj} + \boldsymbol{d}\frac{\mathbf{w}}{||\mathbf{w}||}$

$$D(x) = \mathbf{w}^T\boldsymbol{x} - b$$

$$= \mathbf{w}^T(x_{proj} + \boldsymbol{d}\frac{\mathbf{w}}{||\mathbf{w}||}) - b$$

$$= \mathbf{w}^T x_{proj} - b + \boldsymbol{d}\frac{\mathbf{w}^T\mathbf{w}}{||\mathbf{w}||} = 0 + \boldsymbol{d}||\mathbf{w}||$$

$$\Rightarrow \boldsymbol{d} = \frac{D(x)}{||\mathbf{w}||}$$



Cost function for obtaining the optimal separating hyperplane

- $d_+ = \left|\frac{D(x_+)}{||\mathbf{w}||}\right| = \frac{+1}{||\mathbf{w}||}$, $d_- = \left|\frac{D(x_-)}{||\mathbf{w}||}\right| = \left|\frac{-1}{||\mathbf{w}||}\right| = \frac{1}{||\mathbf{w}||}$

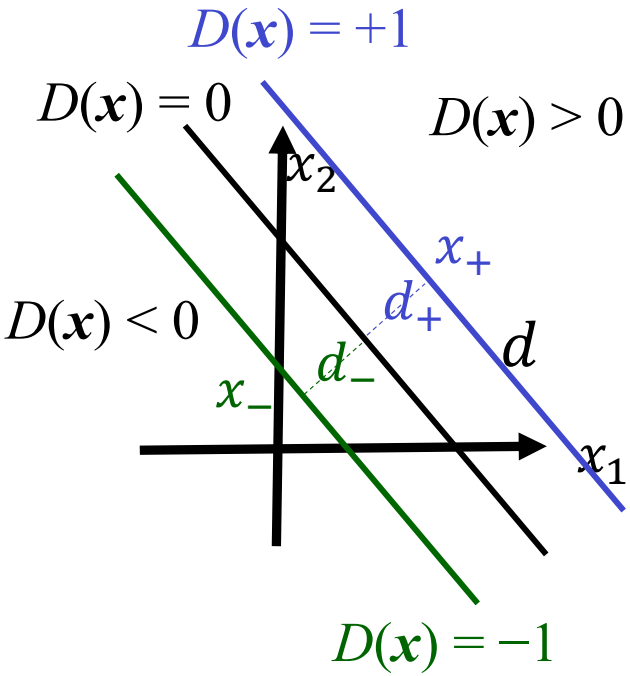
- Margin $= d_+ + d_- = \frac{2}{||\mathbf{w}||}$

- The optimal separating hyperplane can be obtained by minimizing

$$Q(\mathbf{w}) = \frac{1}{2}||\mathbf{w}||^2 \qquad ③$$

with respect to $\mathbf{w}$ and b subject to the constraints

$$y_i(\mathbf{w}^T \boldsymbol{x_i} - b) \geq 1, i = 1, \ldots, N \qquad ④$$



Comment 5.1 – Constrained optimisation with Lagrange multipliers: At various points in this book we will need to perform constrained optimisations – finding the values of a set of parameters that maximise (or minimise) an objective function but that also satisfy some constraints. This can be done using *Lagrange* multipliers. In particular, we make a new objective function which includes the original plus an additional term for each constraint. The form of these terms is chosen such that the optimum of the new function is equal to the optimum of the constrained problem.

For example, suppose we wish to minimise $f(w)$ subject to the constraint $g(w) \leq a$:

$$\begin{aligned} \underset{w}{\text{argmin}} \quad & f(w) \\ \text{subject to} \quad & g(w) \leq a. \end{aligned}$$

The new objective function is produced by adding a Lagrangian term of the form $\lambda(a - g(w))$ and optimised over both w *and* the Lagrange multiplier λ:

$$\begin{aligned} \underset{w,\lambda}{\text{argmin}} \quad & f(w) - \lambda(g(w) - a) \\ \text{subject to} \quad & \lambda > 0. \end{aligned}$$

We are not going to go into the details of how this works here. Whenever we perform constrained optimisation, we will state the necessary Lagrangian terms without any further details. For more details, see the suggested reading at the end of this chapter.

Optimization with m inequality constraints

- Find $\boldsymbol{x} = [x_1, \cdots, x_n]^T$ that

  Minimize $F(\boldsymbol{x})$ ①

  subject to $g_i(\boldsymbol{x}) \leq 0, i = 1, \cdots, m$ ②

- If x satisfies the inequality constraints ②, it is said to be *feasible*. Otherwise it is called *infeasible*

- The ith constraint $g_i(\boldsymbol{x}) \leq 0$ is said to be active at a point $\boldsymbol{x}$ if $g_i(\boldsymbol{x}) = 0$.

- The constraints ② can be converted to equality constraints by adding positive slack variables to get:

  Minimize $F(\boldsymbol{x})$ ①

  subject to $g_i(\boldsymbol{x}) + t_i^2 = 0, i = 1, \cdots, m$ ③

Optimization with m inequality constraints

- ① ③ is an optimization problem with only m equality constraints
- Let $\boldsymbol{t} = [t_1, \cdots, t_m]^T$, $\boldsymbol{\lambda} = [\lambda_1, \cdots, \lambda_m]^T$, the Lagrangian has the form:

$$L(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{\lambda}) = F(\boldsymbol{x}) + \sum_{i=1}^{m} \lambda_i (g_i(\boldsymbol{x}) + t_i^2),$$

which has n+2m unknown $\boldsymbol{x}^*$, $\boldsymbol{t}^*$ and $\boldsymbol{\lambda}^*$

- The optimal conditions are

$$\frac{\partial L}{\partial \boldsymbol{x}} = 0 \;\Rightarrow\; \frac{\partial F(\boldsymbol{x})}{\partial \boldsymbol{x}} + \sum_{i=1}^{m} \lambda_i \frac{\partial g_i(\boldsymbol{x})}{\partial \boldsymbol{x}} = 0, \qquad ④$$

$$\frac{\partial L}{\partial t_i} = 0 \;\Rightarrow\; 2\lambda_i t_i = 0, \qquad i = 1, \cdots, m \qquad ⑤$$

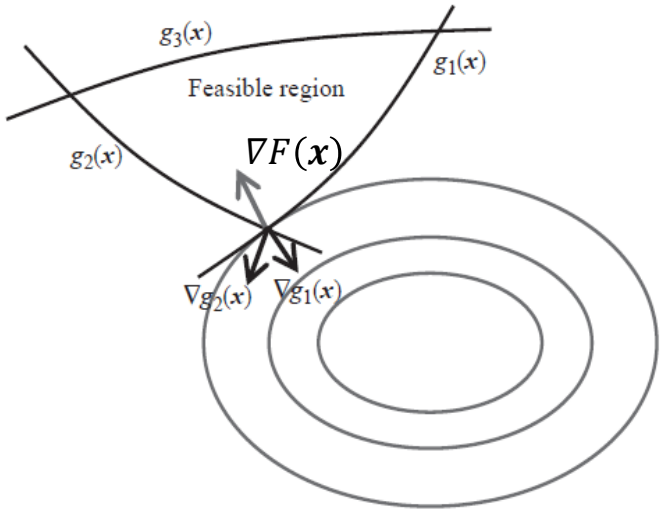
$$\frac{\partial L}{\partial \lambda_i} = 0 \;\Rightarrow\; g_i(\boldsymbol{x}) + t_i^2 = 0, \qquad i = 1, \cdots, m \qquad ⑥$$

- ④⑤ ⑥ are usually called the Karush–Kuhn–Tucker (KKT) conditions

Optimization with *m* inequality constraints

- ④ $\Rightarrow \frac{\partial F(\boldsymbol{x})}{\partial \boldsymbol{x}}$ is a linear combination of $\frac{\partial g_i(\boldsymbol{x})}{\partial \boldsymbol{x}}$ with $\lambda_i \neq 0$

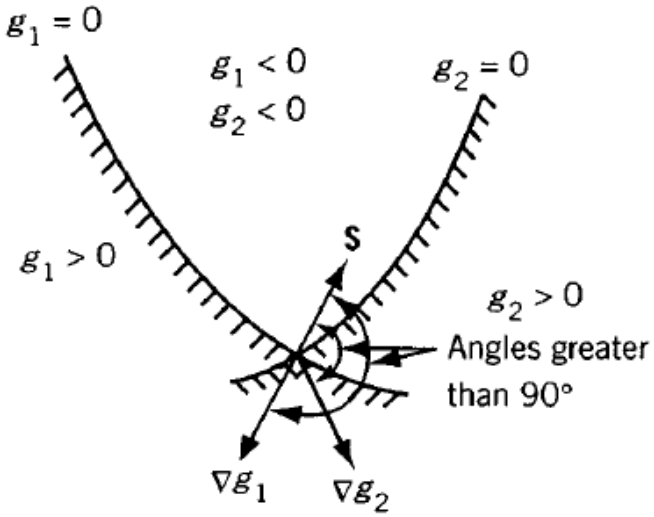
- $\lambda_i t_i = 0$ ⑤ $\Rightarrow$ either $\lambda_i = 0 \Rightarrow t_i \neq 0$ and $g_i(\boldsymbol{x}) + t_i^2 = 0 \Rightarrow g_i(\boldsymbol{x}) < 0$ (inactive)
  or $\lambda_i \neq 0 \Rightarrow t_i = 0$ and $g_i(\boldsymbol{x}) + t_i^2 = 0 \Rightarrow g_i(\boldsymbol{x}) = 0$ (active).
  $\Rightarrow \lambda_i g_i(\boldsymbol{x}) = 0$ (we will show $\lambda_i > 0$ when $g_i(\boldsymbol{x}) = 0$)
- Combining ④& ⑤, one concludes that at the optimal solution, $\frac{\partial F(\boldsymbol{x})}{\partial \boldsymbol{x}}$ is a linear combination of the gradients of active constraints.

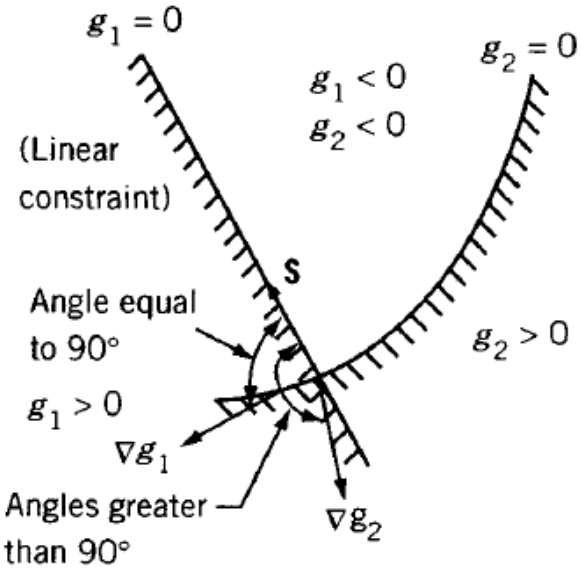


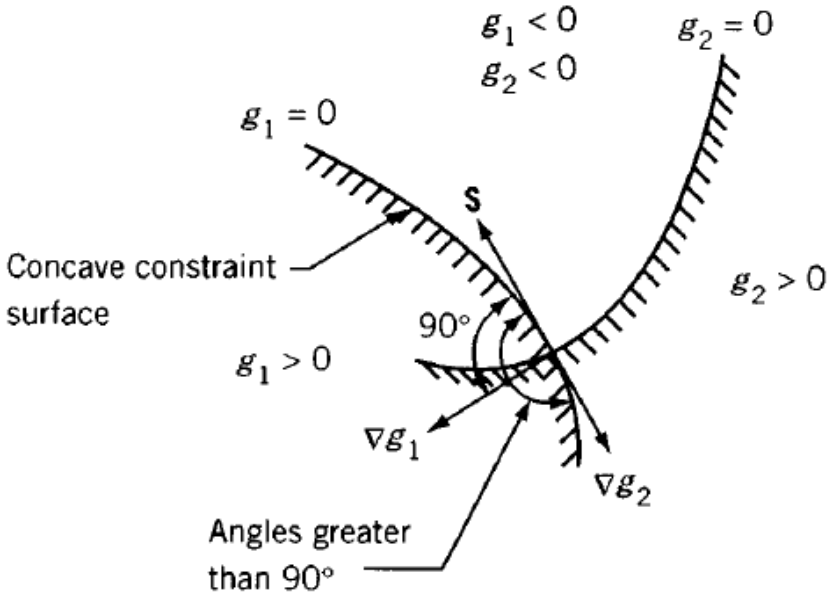
An illustration of the optimality conditions for inequality constraints; the feasible region is defined by 3 constraints and at the optimal point, $g_1(\boldsymbol{x})$ *and* $g_2(\boldsymbol{x})$ *are active. At this point,* $\nabla F(\boldsymbol{x})$ *is a linear function of the gradients of the active constraints* $\nabla g_1(\boldsymbol{x}), \nabla g_2(\boldsymbol{x})$

Feasible direction S for x on active constraint

- **S** is *a feasible direction* from $\boldsymbol{x}$ if a small step along **S** does not leave the feasible region.
- That is, **S** satisfies $g_j(\boldsymbol{x}+\mathbf{S}) \approx g_j(\boldsymbol{x})(=0$ since active constraint$) + \mathbf{S}^T \nabla g_j(\boldsymbol{x}) < 0$ is called a feasible direction.
- If constraint is linear or concave, any **S** satisfies $\mathbf{S}^T \nabla g_j(\boldsymbol{x}) < 0$ is a feasible direction.
- $\mathbf{S}^T \nabla g_j(\boldsymbol{x}) = ||\mathbf{S}^T||||\nabla g_j|| cos\theta < 0 \Rightarrow$ S makes an obtuse angle with all the ∇g_j.
- In linear or concave constraints, the angle can be as low as 90°.







Why $\lambda_j > 0$ when constraints are active $g_i(\boldsymbol{x}) = 0$

- Suppose only two constraints are active ($m = 2$) at the optimum point $\boldsymbol{x}^*$

$$\nabla_x F(\boldsymbol{x}^*) + \lambda_1^* \nabla g_1(\boldsymbol{x}^*) + \lambda_2^* \nabla g_2(\boldsymbol{x}^*) = 0$$
$$\Rightarrow \nabla_x F(\boldsymbol{x}^*) = -\lambda_1^* \nabla g_1(\boldsymbol{x}^*) - \lambda_2^* \nabla g_2(\boldsymbol{x}^*)$$

- Let $\mathbf{S}$ be a feasible direction (i.e. $g_j(\boldsymbol{x}^* + \mathbf{S}) < 0$) at the optimum point $\boldsymbol{x}^*$,

$\Rightarrow\ 0 > g_j(\boldsymbol{x}^* + \mathbf{S}) \approx g_j(\boldsymbol{x}^*) (= 0\ \because \text{active constrain}) + \mathbf{S}^T \nabla g_j(\boldsymbol{x})$

$\Rightarrow\ \mathbf{S}^T \nabla g_1(\boldsymbol{x}^*) < 0$ and $\mathbf{S}^T \nabla g_2(\boldsymbol{x}^*) < 0$

- But $\boldsymbol{x}^*$ is an optimal solution, if there exists $\mathbf{S} \neq \mathbf{0}$ we must have

$$f(\boldsymbol{x}^* + \mathbf{S}) > f(\boldsymbol{x}^*) \Rightarrow f(\boldsymbol{x}^* + \mathbf{S}) - f(\boldsymbol{x}^*) \approx \mathbf{S}^T\ \nabla f(\boldsymbol{x}^*) > 0$$

- Thus if $\lambda_1^* > 0$ and $\lambda_2^* > 0$, $\Rightarrow\ \mathbf{S}^T \nabla_x F(\boldsymbol{x}^*) = -\lambda_1^* \mathbf{S}^T \nabla g_1(\boldsymbol{x}^*) - \lambda_2^* \mathbf{S}^T \nabla g_2(\boldsymbol{x}^*) > 0.$

- $\mathbf{S}^T \nabla f > 0 \Rightarrow f \uparrow$ as we move along the direction $\mathbf{S}$.
- Hence if $\lambda_1^* > 0$ and $\lambda_2^* > 0$, we will not be able to find any direction in the feasible domain along which the function value can be decreased further.

SINGIRESU S. RAO, ENGINEERING OPTIMIZATION Theory and Practice Third Edition, 2009

Optimization with *m* inequality constraints

- The necessary KKT condition for inequality constraints can thus be cast in the standard form

$$\frac{\partial F(\boldsymbol{x})}{\partial x_i} + \sum_{j=1}^{m} \lambda_j \frac{\partial g_j(\boldsymbol{x})}{\partial x_i} = 0, \qquad i = 1, \cdots, n \qquad ⑦$$

$$\lambda_j g_j(\boldsymbol{x}) = 0, \quad \textit{complementarity condition} \qquad j = 1, \cdots, m \qquad ⑧$$

$$g_j(\boldsymbol{x}) \leq 0, \qquad j = 1, \cdots, m \qquad ⑨$$

$$\lambda_j \geq 0, \qquad j = 1, \cdots, m \qquad ⑩$$

- Condition $\lambda_j \geq 0$ ⑩ for the inequality constraints $g_j(\boldsymbol{x}) \leq 0$ ensures F will not be reduced by a move off any of the active constraints at $\boldsymbol{x}^*$ to the interior of the feasible region.

Example: 2 inequality constraints

Minimize $f(x_1, x_2) = x_1^2 + x_2^2 - 14x_1 - 6x_2$,

subject to $g_1(x_1, x_2) = x_1 + x_2 - 2 \le 0$,

$g_2(x_1, x_2) = x_1 + 2x_2 - 3 \le 0$,

- ⑦ $\Rightarrow \nabla_{\boldsymbol{x}} L(\boldsymbol{x}^*, \lambda_1^*) = \nabla f(\boldsymbol{x}^*) + \lambda_1^* \nabla g_1(\boldsymbol{x}^*) + \lambda_2^* \nabla g_2(\boldsymbol{x}^*) = 0$

$$\Rightarrow \begin{bmatrix} 2x_1^* - 14 \\ 2x_2^* - 6 \end{bmatrix} + \lambda_1^* \begin{bmatrix} 1 \\ 1 \end{bmatrix} + \lambda_2^* \begin{bmatrix} 1 \\ 2 \end{bmatrix} = 0$$

- ⑧ $\Rightarrow \lambda_1^*(x_1^* + x_2^* - 2) = 0;\ \lambda_2^*(x_1^* + 2x_2^* - 3) = 0$
- ⑨ $\Rightarrow x_1^* + x_2^* - 2 \le 0;\ x_1^* + 2x_2^* - 3 \le 0$; ⑩ $\Rightarrow \lambda_1^* \ge 0;\ \lambda_2^* \ge 0$
- Four possibilities: 1 : both constraints are active, 2 :only the first one is active, 3:only the second one is active, 4: none of the constraints is active.

- 1: $g_1 = 0$ & $g_2 = 0 \Rightarrow \begin{bmatrix} 2x_1^* - 14 \\ 2x_2^* - 6 \end{bmatrix} + \lambda_1^* \begin{bmatrix} 1 \\ 1 \end{bmatrix} + \lambda_2^* \begin{bmatrix} 1 \\ 2 \end{bmatrix} = 0, x_1^* + x_2^* = 2, x_1^* + 2x_2^* = 3$

$\Rightarrow x_1^* = 1, x_2^* = 1 \Rightarrow \lambda_1^* = 20, \lambda_2^* = -8 < 0$, violate $\lambda_2^* > 0$

- 2: $g_1 = 0$ & $g_2 < 0 \Rightarrow \begin{bmatrix} 2x_1^* - 14 \\ 2x_2^* - 6 \end{bmatrix} + \lambda_1^* \begin{bmatrix} 1 \\ 1 \end{bmatrix} = 0, x_1^* + x_2^* = 2$

$\Rightarrow \lambda_1^* = 8 \Rightarrow x_1^* = 3, x_2^* = -1 \Rightarrow g_2(\boldsymbol{x}^*) = -2 < 0 \Rightarrow$ Satisfy KKT condition

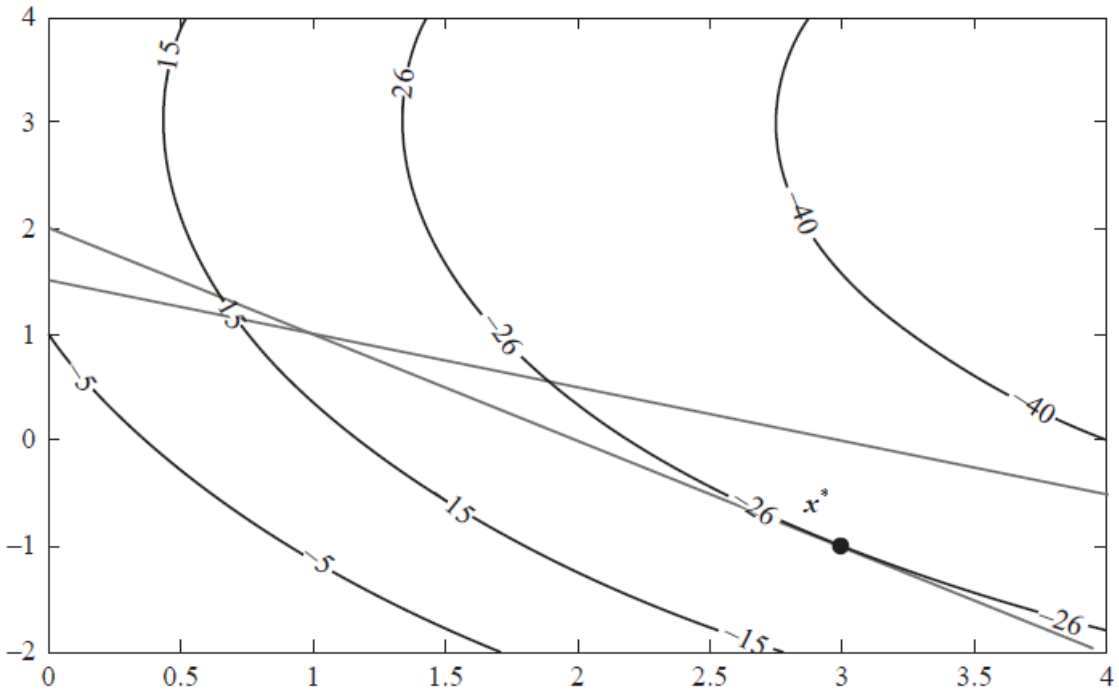
Example: 2 inequality constraints

- 3: $g_1 < 0$ & $g_2 = 0 \Rightarrow \begin{bmatrix} 2x_1^* - 14 \\ 2x_2^* - 6 \end{bmatrix} + \lambda_2^* \begin{bmatrix} 1 \\ 2 \end{bmatrix} = 0, x_1^* + 2x_2^* = 3 \Rightarrow \lambda_2^* = 4$

  $\Rightarrow x_1^* = 5, x_2^* = -1 \Rightarrow g_1(\boldsymbol{x}^*) = 2 > 0$, violate $g_1(\boldsymbol{x}^*) \leq 0$

- 4: $g_1 < 0$ & $g_2 < 0 \Rightarrow \begin{bmatrix} 2x_1^* - 14 \\ 2x_2^* - 6 \end{bmatrix} = 0,$

  $\Rightarrow x_1^* = 7, x_2^* = 3 \Rightarrow g_1(\boldsymbol{x}^*) = 8 > 0$, $g_2(\boldsymbol{x}^*) = 10 > 0$, violate $g_1(\boldsymbol{x}^*) \leq 0$ & $g_2(\boldsymbol{x}^*) \leq 0$



Convert constrained into unconstrained optimization

- The square of the Euclidean norm **w** in ③ is to make the optimization problem quadratic programming.

- The assumption of linear separability means that there exist **w** and b that satisfy ④. We call the solutions that satisfy ④ **feasible solutions**.

- We first convert the constrained problem given by ③ and ④ into the unconstrained problem

$$Q(\mathbf{w}, b, \boldsymbol{\alpha}) = \frac{1}{2}\mathbf{w}^T\mathbf{w} + \sum_{i=1}^{N} \alpha_i\{1 - t_i(\mathbf{w}^T\boldsymbol{x_i} + b)\} \qquad ⑤$$

where $\boldsymbol{\alpha} = (\alpha_1, \ldots, \alpha_N)^T$ and α_i are the nonnegative Lagrange multipliers.

Karush-Kuhn-Tucker (KKT) conditions

- The optimal solution of ⑤ is given by minimizing w.r.t $\mathbf{w}$ and b and maximizing w.r.t α_i (≥ 0) satisfying the following KKT conditions

$$\frac{\partial Q(\mathbf{w}, b, \boldsymbol{\alpha})}{\partial \mathbf{w}} = \mathbf{w} - \sum_{i=1}^{N} \alpha_i t_i \boldsymbol{x_i} = 0 \Rightarrow \mathbf{w} = \sum_{i=1}^{N} \alpha_i t_i \boldsymbol{x_i} \quad (*)$$

$$\frac{\partial Q(\mathbf{w}, b, \boldsymbol{\alpha})}{\partial b} = -\sum_{i=1}^{N} \alpha_i t_i = 0 \quad (**)$$

$$\alpha_i\{1 - t_i(\mathbf{w}^T \boldsymbol{x_i} + b)\} = 0, \; i = 1, \ldots, N \quad ⑥$$

$$\alpha_i \geq 0, i = 1, \ldots, N$$

- ⑥ are called KKT **complementarity conditions**: $\alpha_i = 0$, or $\alpha_i > 0$ and $t_i(\mathbf{w}^T \boldsymbol{x_i} + b) = 1$ must be satisfied.

- The training data $\boldsymbol{x_i}$ with $\alpha_i > 0$ are called support vectors

Dual Problem

• Substituting (∗) and (∗∗) into ⑤, we obtain the dual problem.

Maximize

$$Q(\mathbf{w}, b, \boldsymbol{\alpha}) = \frac{1}{2}\mathbf{w}^T\mathbf{w} + \sum_{i=1}^{N}\alpha_i\{1 - t_i(\mathbf{w}^T\boldsymbol{x_i} + b)\}$$

$$= \frac{1}{2}\sum_{i=1}^{N}\alpha_i\, t_i\, \boldsymbol{x}_i^T \sum_{j=1}^{N}\alpha_j\, t_j \boldsymbol{x_j} + \sum_{i=1}^{N}\alpha_i\{1 - t_i\,(\sum_{j=1}^{N}\alpha_j\, t_j \boldsymbol{x}_j^T \boldsymbol{x}_i + b)\}$$

$$= \sum_{i=1}^{N}\alpha_i - \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N}\alpha_i\alpha_j\, t_i t_j \boldsymbol{x}_i^T \boldsymbol{x}_j - b\sum_{i=1}^{N}\alpha_i\, t_i$$

$$= \sum_{i=1}^{N}\alpha_i - \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N}\alpha_i\alpha_j\, t_i t_j \boldsymbol{x}_i^T \boldsymbol{x}_j - b \times 0$$

w.r.t. α_i subject to

$$\sum_{i=1}^{N}\alpha_i\, t_i = 0, \quad \alpha_i \geq 0, \qquad i = 1, \ldots, N$$

• This is the *dual problem* and it is in terms of α_i's only

⇒ α_i's are used to get optimal $\mathbf{w}$ and b

Quadratic Programming (QP)

$$\min_x \frac{1}{2}\boldsymbol{x}^T P\boldsymbol{x} + \boldsymbol{q}^T\boldsymbol{x}$$
$$\text{subject to } \mathrm{G}\boldsymbol{x} \le \boldsymbol{h}$$
$$\mathrm{A}\boldsymbol{x} = \boldsymbol{b}$$

- To convert the dual problem into QP, let's work out a simple example

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

$$= (\alpha_1 y_1 \boldsymbol{x}_1^T + \alpha_2 y_2 \boldsymbol{x}_2^T)(\alpha_1 y_1 \boldsymbol{x}_1 + \alpha_2 y_2 \boldsymbol{x}_2) = [\alpha_1 \quad \alpha_2]\begin{bmatrix} y_1\boldsymbol{x}_1^T \\ y_2\boldsymbol{x}_2^T \end{bmatrix}[y_1\boldsymbol{x}_1 \quad y_2\boldsymbol{x}_2]\begin{bmatrix}\alpha_1 \\ \alpha_2\end{bmatrix}$$

Note $\begin{bmatrix} y_1\boldsymbol{x}_1^T \\ y_2\boldsymbol{x}_2^T \end{bmatrix}[y_1\boldsymbol{x}_1 \quad y_2\boldsymbol{x}_2] = \left(\begin{bmatrix}y_1 \\ y_2\end{bmatrix}[y_1 \quad y_2]\right).* \begin{bmatrix} \boldsymbol{x}_1^T\boldsymbol{x}_1 & \boldsymbol{x}_1^T\boldsymbol{x}_2 \\ \boldsymbol{x}_2^T\boldsymbol{x}_1 & \boldsymbol{x}_2^T\boldsymbol{x}_2 \end{bmatrix} = (\boldsymbol{y}\otimes y).*\boldsymbol{K}$

$$\Rightarrow \sum_{i=1}^{2}\sum_{j=1}^{2}\alpha_i\alpha_j y_i y_j \boldsymbol{x}_i^T\boldsymbol{x}_j = [\alpha_1 \quad \alpha_2](\boldsymbol{y}\otimes y).*\boldsymbol{K}\begin{bmatrix}\alpha_1 \\ \alpha_2\end{bmatrix}$$

Convert dual problem into QP notation

$$\boldsymbol{\alpha} = \begin{bmatrix} \alpha_1 \\ \vdots \\ \alpha_N \end{bmatrix}, \boldsymbol{y} = \begin{bmatrix} y_1 \\ \vdots \\ y_N \end{bmatrix}, \mathbf{1}_N = \begin{bmatrix} 1 \\ \vdots \\ 1 \end{bmatrix}, \underbrace{\boldsymbol{X}^T}_{d \times N} = [\boldsymbol{x}_1 \;\; \boldsymbol{x}_2 \;\; \cdots \boldsymbol{x}_N], \underbrace{\boldsymbol{X}}_{N \times d} = \begin{bmatrix} \boldsymbol{x}_1^T \\ \vdots \\ \boldsymbol{x}_N^T \end{bmatrix}$$

$$\Rightarrow \sum_{i=1}^{N} \alpha_i = \mathbf{1}_N^T \boldsymbol{\alpha},$$

$$\left(\underbrace{\sum_{i=1}^{N} \alpha_i y_i \boldsymbol{x}_i^T}_{1xd} \right) \left(\underbrace{\sum_{j=1}^{N} \alpha_j y_j \boldsymbol{x}_j}_{d \times 1} \right) = \underbrace{\boldsymbol{\alpha}^T}_{1 \times N} \left(\underbrace{\boldsymbol{y} \otimes y}_{N \times N} \right) .* \underbrace{\boldsymbol{K}}_{N \times N} \underbrace{\boldsymbol{\alpha}}_{N \times 1}$$

Where $\boldsymbol{y} \otimes y = \begin{bmatrix} y_1 \\ \vdots \\ y_N \end{bmatrix} [y_1 \quad \cdots \quad y_N]$, $\underbrace{\boldsymbol{K}}_{N \times N} = \underbrace{\boldsymbol{X}}_{N \times d} \underbrace{\boldsymbol{X}^T}_{d \times N}$

Convert dual problem into QP notation

$$Q(\mathbf{w}, b, \boldsymbol{\alpha}) = \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_i \alpha_j y_i y_j \boldsymbol{x}_i^T \boldsymbol{x}_j$$

$$= \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \left(\sum_{i=1}^{N} \alpha_i y_i \boldsymbol{x}_i^T \right) \left(\sum_{j=1}^{N} \alpha_j y_j \boldsymbol{x}_j \right)$$

$$= -\frac{1}{2} \underbrace{\boldsymbol{\alpha}^T}_{1 \times N} \left(\underbrace{\boldsymbol{y} \otimes \boldsymbol{y}}_{N \times N} \right) .* \underbrace{\boldsymbol{K}}_{N \times N} \underbrace{\boldsymbol{\alpha}}_{N \times 1} + \mathbf{1}_N^T \boldsymbol{\alpha}$$

subject to

$$\sum_{i=1}^{N} \alpha_i y_i = \boldsymbol{y}^T \boldsymbol{\alpha} = 0, \quad \alpha_i \geq 0, \qquad i = 1, \ldots, N$$

Convert dual problem into QP notation

$$\min_{\boldsymbol{x}} \frac{1}{2}\boldsymbol{x}^T P\boldsymbol{x} + \boldsymbol{q}^T\boldsymbol{x}$$
$$\text{subject to } \mathrm{G}\boldsymbol{x} \le \boldsymbol{h}$$
$$\mathrm{A}\boldsymbol{x} = \boldsymbol{b}$$

$$\max_{\boldsymbol{\alpha}} -\left(\frac{1}{2}\underbrace{\boldsymbol{\alpha}^T}_{1\times N}\left(\underbrace{\boldsymbol{y}\otimes\boldsymbol{y}}_{N\times N}\right).*\underbrace{\boldsymbol{K}}_{N\times N}\underbrace{\boldsymbol{\alpha}}_{N\times 1} - \mathbf{1}_N^T\boldsymbol{\alpha}\right)$$
$$\text{subject to } \boldsymbol{\alpha} \ge 0$$
$$\boldsymbol{y}^T\boldsymbol{\alpha}=0$$

$$P = (\boldsymbol{y}\otimes\boldsymbol{y}).*\boldsymbol{X}\boldsymbol{X}^T$$
$$\boldsymbol{q} = -\mathbf{1}_N$$
$$\mathrm{G} = -\mathbf{I}_{N\times N}$$
$$\boldsymbol{h} = \mathbf{0}$$
$$\mathrm{A} = \boldsymbol{y}^T$$
$$\boldsymbol{b} = \mathbf{0}$$

Optimal $\mathbf{w} = \sum_{i=1}^{N} \alpha_i t_i \boldsymbol{x}_i$ **and** $b = t_i - \sum_{x_j \in S} \alpha_j t_j \boldsymbol{x}_j^T \boldsymbol{x}_i$, $\alpha_j > 0$

- We may estimate $\boldsymbol{\alpha}$ vector to obtain the *global optimum*. $\mathbf{w} = \sum_{i=1}^{N} \alpha_i t_i \boldsymbol{x}_i$.

- Many of the α_i are 0. Support Vectors (SVs) are the $\boldsymbol{x}_i$' s corresponding to the nonzero α_i' s. Let $S = \{x_i | \alpha_i > 0\}$ be the set of SVs.

a. By *complementary slackness condition*,

$\boldsymbol{x}_i \in S \Rightarrow \alpha_i > 0 \Rightarrow t_i(\mathbf{w}^T \boldsymbol{x}_i + b) = 1 \Rightarrow \boldsymbol{x}_i$ is the closest to the decision boundary.

b. Optimal $\mathbf{w} = \sum_{i=1}^{N} \alpha_i t_i \boldsymbol{x}_i = \sum_{x_i \in S} \alpha_i t_i \boldsymbol{x}_i$ is a linear combination of SVs.

c. $t_i \times t_i(\mathbf{w}^T \boldsymbol{x}_i + b) = t_i \Rightarrow b = t_i - \mathbf{w}^T \boldsymbol{x}_i = t_i - \sum_{x_j \in S} \alpha_j t_j \boldsymbol{x}_j^T \boldsymbol{x}_i$, $\alpha_j > 0$.

d. It is better to average the SVs : $b = \frac{1}{\#(\boldsymbol{x}_i \in S)} \sum_{x_i \in S} (t_i - \mathbf{w}^T \boldsymbol{x}_i)$

Making Prediction

- Data associated with α_i's > 0 are support vectors for Classes 1 and 2.

- $\mathbf{w} = \sum_{i=1}^{N} \alpha_i t_i \boldsymbol{x}_i$ $(*)$, the decision function is (do not need to use $\mathbf{w}$ and b explicitly, use $\alpha_i > 0$, t_i and $\boldsymbol{x}_i$ only)

$$D(\boldsymbol{x}_{\text{new}}) = \mathbf{w}^T \boldsymbol{x}_{\text{new}} + b = \sum_{\boldsymbol{x}_i \in S} \alpha_i t_i \boldsymbol{x}_i{}^T \boldsymbol{x}_{\text{new}} + (t_i - \sum_{\boldsymbol{x}_j \in S} \alpha_j t_j \boldsymbol{x}_j{}^T \boldsymbol{x}_i)$$

- Then unknown datum $\boldsymbol{x}_{\text{new}}$ is classified into:

$$\begin{cases} \text{Class 1 , if } D(\boldsymbol{x}_{\text{new}}) > 0 \\ \text{Class 2 , if } D(\boldsymbol{x}_{\text{new}}) < 0 \end{cases}$$

If $D(\boldsymbol{x}_{\text{new}}) = 0$, $\boldsymbol{x}_{\text{new}}$ is on the boundary and thus is unclassifiable

Example

- Consider a linearly separable case shown in Fig. 2.2, $(x_1, t_1) = (-1,1)$, $(x_2, t_2) = (0,-1)$, $(x_3, t_3) = (1,-1)$, The inequality constraints given by $t_i(\mathbf{w}^T \boldsymbol{x_i} + b) \geq 1, i = 1, \ldots, 3$ are

$$-w + b \geq 1\ , -b \geq 1\ , -(w + b) \geq 1 \qquad (***)$$

- The region of (w, b) that satisfies $(***)$ are given by the shaded region in Fig. 2.3. Thus the solution that minimizes $||\mathrm{w}||^2$ is given by

$b = -1, w = -2.$

- The decision function is $D(x) = -2x - 1$
- The class boundary is $x = -1/2$
- $x = 0$ and -1 are support vectors

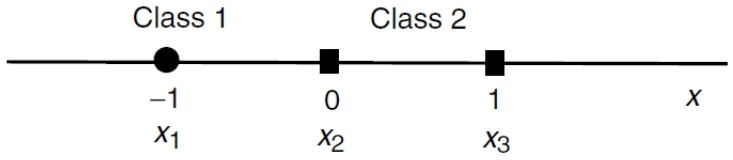


Fig. 2.2. Linearly separable one-dimensional case

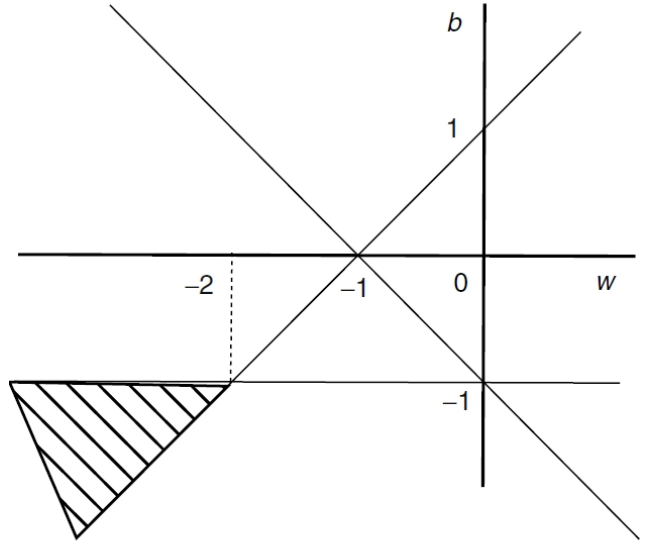


Fig. 2.3. Region that satisfies constraints

Example

- The dual problem is to maximize

$$Q(\boldsymbol{\alpha}) = \sum_{i=1}^{3} \alpha_i - \frac{\mathbf{1}}{\mathbf{2}} \sum_{i=1}^{3} \sum_{j=1}^{3} \alpha_i \alpha_j \, t_i t_j \boldsymbol{x}_i^T \boldsymbol{x}_j, \quad (x_1, t_1) = (-1,1), (x_2, t_2) = (0,-1), (x_3, t_3) = (1,-1)$$

$$= \alpha_1 + \alpha_2 + \alpha_3 - \frac{\mathbf{1}}{\mathbf{2}} \{\alpha_1^2 1^2 (-1)^2 + \alpha_1 \alpha_2 (-1) 0 + \alpha_1 \alpha_3 (-1)(-1) +$$
$$\alpha_2 \alpha_1 (-1) 0 + \alpha_2^2 (-1)^2 (0)^2 + \alpha_2 \alpha_3 (-1)(-1) 0 +$$
$$\alpha_3 \alpha_1 (-1)(-1) + \alpha_3 \alpha_2 (-1)^2 0 + \alpha_3^2 (-1)^2 1\}$$

$$= \alpha_1 + \alpha_2 + \alpha_3 - \frac{\mathbf{1}}{\mathbf{2}} (\alpha_1 + \alpha_3)^2 \qquad (****)$$

subject to

$$\sum_{i=1}^{3} \alpha_i \, t_i = \alpha_1 - \alpha_2 - \alpha_3 = 0, \; \alpha_i \geq 0, i = 1, \ldots, 3$$

- Substituting $\alpha_2 = \alpha_1 - \alpha_3$ into $(****)$, we obtain

$Q(\boldsymbol{\alpha}) = 2\alpha_1 - \frac{\mathbf{1}}{\mathbf{2}} (\alpha_1 + \alpha_3)^2$ subject to $\alpha_i \geq 0, i = 1, \ldots, 3$

which is maximized when $\alpha_3 = 0$, since $\alpha_3 \geq 0$

Example

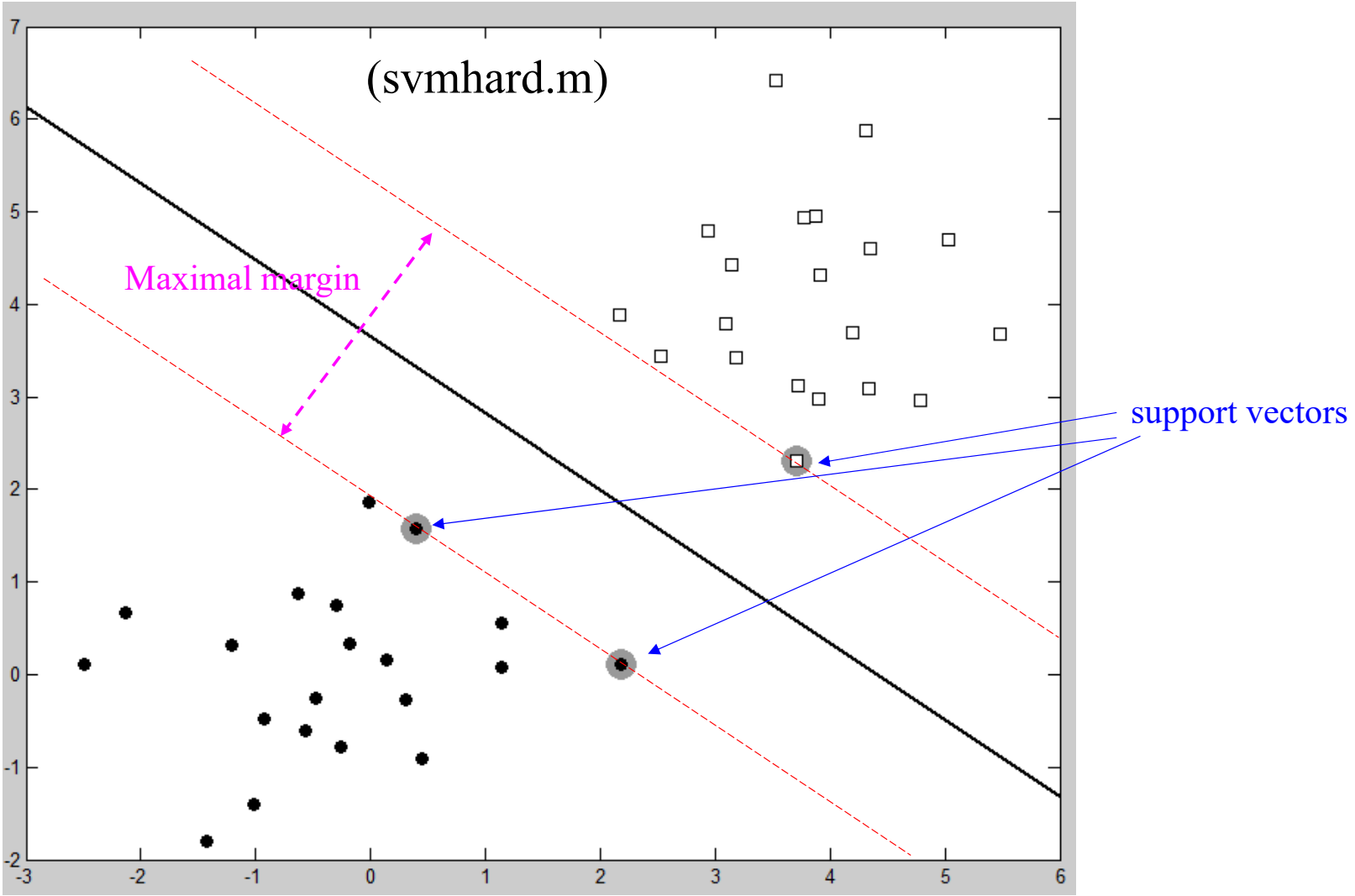
- Now $Q(\boldsymbol{\alpha}) = 2\alpha_1 - \frac{\mathbf{1}}{\mathbf{2}}{\alpha_1}^2 = -\frac{\mathbf{1}}{\mathbf{2}}(\alpha_1 - 2)^2 + 2,\ \alpha_1 \geq 0$

which is maximized for $\alpha_1 = 2$.

- The optimal solution for $(****)$ is $\alpha_1 = 2, \alpha_3 = 0, \alpha_2 = \alpha_1 - \alpha_3 = 2$
- Therefore $x = -1$ ($\alpha_1 = 2 > 0$) and 0 ($\alpha_2 = 2 > 0$) are support vectors and $w = \sum_{i=1}^{3} \alpha_i\, t_i x_i = 2(1)(-1) + 2(-1)0 + 0(-1)(1) = -2$ and $b = t_i - w^T x_i = t_1 - (-2)x_1 = 1 - (-2)(-1) = -1$ ($\alpha_1 = 2 > 0$, x_1 is a support vector $\Rightarrow\ t_1(w^T x_1 + b) = 1$), which are the same as the solution obtained by solving the primary problem.

- Consider changing the label of x_3 into that of the opposite class, i.e., $t_3 = 1$. Then the problem becomes inseparable and last inequality in $(***)$ becomes $w + b \geq 1$. Thus, from Fig 2.3 there is no feasible solution.

Decision Boundary and Support Vectors for a Linear SVM



MATLAB script: svmhard.m

```
%% svmhard.m
% From A First Course in Machine Learning, Chapter 5.
% Hard margin SVM

clear all;close all;
%% Generate the data
x = [randn(20,2);randn(20,2)+4];
t = [repmat(-1,20,1);repmat(1,20,1)];

%% Plot the data
ma = {'ko','ks'};
fc = {[0 0 0],[1 1 1]};
tv = unique(t);
figure(1); hold off
for i = 1:length(tv)
    pos = find(t==tv(i));
    plot(x(pos,1),x(pos,2),ma{i},'markerfacecolor',fc{i});
    hold on
end
```

```
%% Setup the optimisation problem
N = size(x,1);
K = x*x';
H = (t*t').*K + 1e-5*eye(N);
f = repmat(1,N,1);
A = [];b = [];
LB = repmat(0,N,1); UB = repmat(inf,N,1);
Aeq = t';beq = 0;

% Following line runs the SVM
alpha = quadprog(H,-f,A,b,Aeq,beq,LB,UB);

% Compute the bias
fout = sum(repmat(alpha.*t,1,N).*K,1)';
pos = find(alpha>1e-6);
bias = mean(t(pos)-fout(pos));
```

α_i's > 0 are support vectors

$$Q(\mathbf{w}, b, \boldsymbol{\alpha}) = \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_i \alpha_j t_i t_j \boldsymbol{x}_i^T \boldsymbol{x}_j$$

$$\sum_{i=1}^{N} \alpha_i t_i = 0, \quad \alpha_i \geq 0, \quad i = 1, \ldots, N$$

$$\mathbf{w} = \sum_{i=1}^{N} \alpha_i t_i \boldsymbol{x}_i$$

$$b = \frac{1}{\#(\boldsymbol{x}_i \in S)} \sum_{x_i \in S} (t_i - \mathbf{w}^T \boldsymbol{x}_i)$$

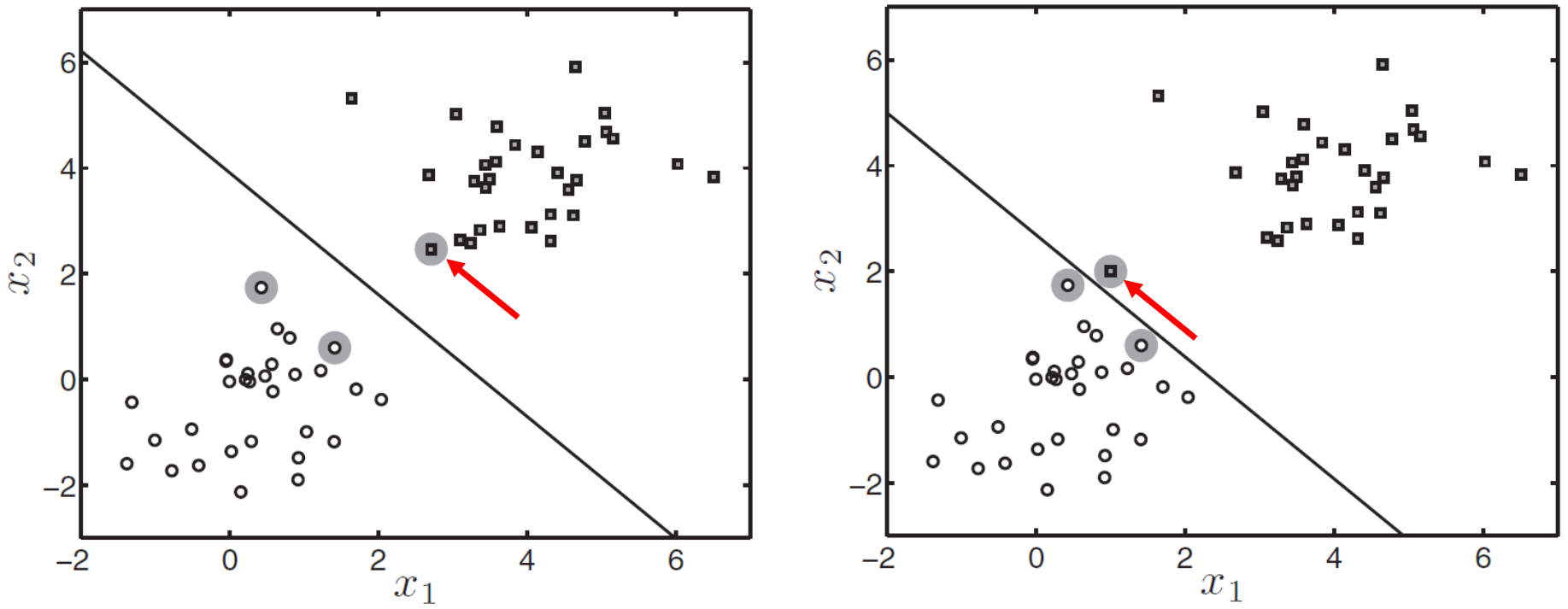
$S = \{x_i | \alpha_i > 0\}$ be the set of SVs

```
%% Plot the data, decision boundary and Support vectors
figure(1);hold off
pos = find(alpha>1e-6);
plot(x(pos,1),x(pos,2),'ko','markersize',15,'markerfacecolor',[0.6 0.6 0.6],...
    'markeredgecolor',[0.6 0.6 0.6]);
hold on
for i = 1:length(tv)
    pos = find(t==tv(i));
    plot(x(pos,1),x(pos,2),ma{i},'markerfacecolor',fc{i});
end

xp = xlim;

% Because this is a linear SVM, we can compute w and plot the decision
% boundary exactly.

w = sum(repmat(alpha.*t,1,2).*x,1)';
yp = -(bias + w(1)*xp)/w(2);
plot(xp,yp,'k','linewidth',2)
```

α_i's > 0 are support vectors for Classes 1 and 2

$$\mathbf{w} = \sum_{i=1}^{N} \alpha_i \, t_i \, \boldsymbol{x_i}$$

Move Support from One Class Closer to the Other Class



- Left: Decision boundary and support vectors for a linear SVM.
- Right: Moving the grey-square support vector appears to have a large effect on the position of the decision boundary. This is another example of over-fitting: we are allowing the data to have too much influence.

Soft-Margin Support Vector Machines

- When linearly inseparable, there is no feasible solution, and the hard-margin support vector machine is unsolvable.

- The SVM is extended to inseparable case.

- Introduce slack variables $\xi_i \geq 0$ into $t_i(\mathbf{w}^T \boldsymbol{x_i} + b) \geq 1$.

$\Rightarrow \quad t_i(\mathbf{w}^T \boldsymbol{x_i} + b) \geq 1 - \xi_i, i = 1, \ldots, N$

If $0 \leq \xi_i \leq 1$, this data is correctly classified.

If $\xi_i > 1$, this data is misclassified.

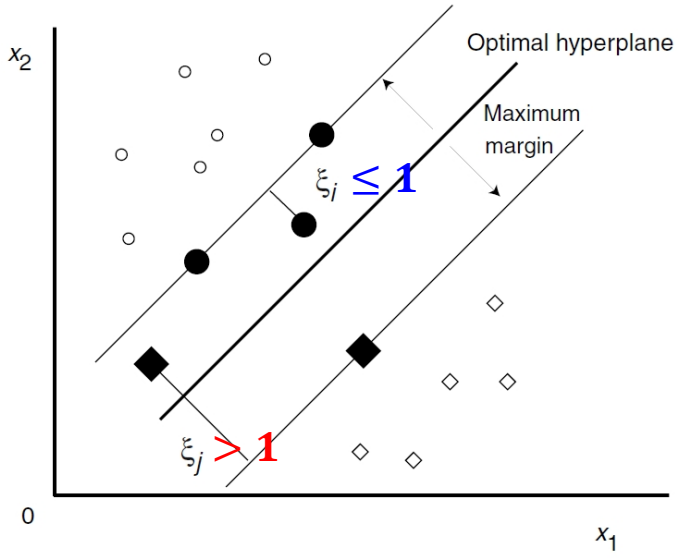


Fig. 2.4. Inseparable case in a two-dimensional space

Constrained Optimization in Soft-Margin SVM

- We minimize $Q(\mathbf{w}, b, \boldsymbol{\xi}) = \frac{1}{2}||\mathbf{w}||^2 + C\sum_{i=1}^{N} \xi_i^p$,

  subject to $\xi_i \geq 0$ and $t_i(\mathbf{w}^T \boldsymbol{x_i} + b) \geq 1 - \xi_i, i = 1, \ldots, N$

where

  - $\boldsymbol{\xi} = (\xi_1, \ldots, \xi_N)^{\mathbf{T}}$
  - C controls to what extent to allow points to sit within the margin band or on the wrong side of the decision boundary
  - $p = 1$ (l_1 soft-margin SVM), or 2(l_2 soft-margin SVM)

- We call the obtained hyperplane the **soft-margin hyperplane**.

Constrained Optimization in Soft-Margin SVM

- Introduce the nonnegative Lagrange multipliers α_i and β_i, we obtain (p=1)

$$Q(\mathbf{w}, b, \boldsymbol{\xi}, \boldsymbol{\alpha}, \boldsymbol{\beta}) = \frac{1}{2}\mathbf{w}^T\mathbf{w} + \mathrm{C}\sum_{i=1}^{N} \xi_i + \sum_{i=1}^{N} \alpha_i\{1 - \xi_i - t_i(\mathbf{w}^T\boldsymbol{x_i} + b)\} + \sum_{i=1}^{N} \beta_i(-\xi_i), i = 1, \ldots, N \quad ①$$

- For the optimal solution, the following KKT conditions are satisfied

$$\frac{\partial Q(\mathbf{w}, b, \xi, \boldsymbol{\alpha}, \boldsymbol{\beta})}{\partial \mathbf{w}} = \mathbf{w} - \sum_{i=1}^{N} \alpha_i t_i \boldsymbol{x_i} = 0 \Rightarrow \mathbf{w} = \sum_{i=1}^{N} \alpha_i t_i \boldsymbol{x_i} \quad (*)$$

$$\frac{\partial Q(\mathbf{w}, b, \xi, \boldsymbol{\alpha}, \boldsymbol{\beta})}{\partial b} = -\sum_{i=1}^{N} \alpha_i t_i = 0 \quad (**)$$

$$\frac{\partial Q(\mathbf{w}, b, \xi, \boldsymbol{\alpha}, \boldsymbol{\beta})}{\partial \xi_i} = \mathrm{C} - \alpha_i - \beta_i = 0 \Rightarrow \alpha_i + \beta_i = \mathrm{C}, i = 1, \ldots, N \quad (***)$$

$$\alpha_i\{1 - \xi_i - t_i(\mathbf{w}^T\boldsymbol{x_i} + b)\} = 0, \; i = 1, \ldots, N \quad ②$$

$$\beta_i\xi_i = 0, \quad i = 1, \ldots, N \quad ③$$

$$\alpha_i \geq 0, \beta_i \geq 0, \xi_i \geq 0, \quad i = 1, \ldots, N$$

Quadratic Programming Problem: in Soft-Margin SVM

- Substituting $(*)$, $(**)$, $(***)$ into ①, we obtain the dual problem.

Maximize

$$Q(\mathbf{w}, b, \boldsymbol{\xi}, \boldsymbol{\alpha}, \boldsymbol{\beta}) = \frac{1}{2}\mathbf{w}^T\mathbf{w} + \sum_{i=1}^{N} \xi_i(\alpha_i + \beta_i) + \sum_{i=1}^{N} \alpha_i\{1 - \xi_i - t_i(\mathbf{w}^T\boldsymbol{x_i} + b)\} + \sum_{i=1}^{N} \beta_i(-\xi_i),$$

$$= \frac{1}{2}\mathbf{w}^T\mathbf{w} + \sum_{i=1}^{N} \alpha_i\{1 - t_i(\mathbf{w}^T\boldsymbol{x_i} + b)\}$$

$$= \sum_{i=1}^{N} \alpha_i - \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N} \alpha_i\alpha_j\, t_i t_j \boldsymbol{x}_i^T\boldsymbol{x}_j$$

with respect to α_i subject to the constraints

$\sum_{i=1}^{N} \alpha_i\, t_i = 0$, $0 \le \alpha_i \le C, i = 1, \ldots, N$

- The only difference between l_1 soft-margin SVM and hard margin SVM is that α_i cannot exceed C (since $\alpha_i + \beta_i = C$, $\beta_i \ge 0$).

Quadratic Programming Problem: in Soft-Margin SVM

- ② and ③ are called KKT (complementarity) conditions

- From $\alpha_i + \beta_i$=C, $\beta_i \xi_i = 0$ and ②: $\alpha_i\{1 - \xi_i - t_i(\mathbf{w}^T \boldsymbol{x_i} + b)\}$=0,

  1. $\alpha_i = 0$. $\Rightarrow \beta_i$=C, $\Rightarrow \xi_i = 0$. $\Rightarrow$ $\boldsymbol{x_i}$ is correctly classified

  2. $0 < \alpha_i < C$,

  ② $\Rightarrow$ $t_i(\mathbf{w}^T \boldsymbol{x_i} + b) - 1 + \xi_i = 0$, and $\beta_i = C - \alpha_i \neq 0, \Rightarrow \xi_i = \frac{0}{\beta_i} = 0$.

  $\Rightarrow$ $t_i(\mathbf{w}^T \boldsymbol{x_i} + b) = 1$ and $\boldsymbol{x_i}$ is a support vector.

  We call the support vector with $0 < \alpha_i < C$ an unbounded (good) SV.

  3. $\alpha_i = C$.

  ② $\Rightarrow$ $t_i(\mathbf{w}^T \boldsymbol{x_i} + b) - 1 + \xi_i = 0$ and $\xi_i \geq 0$. Thus $\boldsymbol{x_i}$ is a support vector.

  We call the support vector with $\alpha_i = C$ a bounded (may be bad) SV.

  If $0 \leq \xi_i \leq 1$, $\boldsymbol{x_i}$ is correctly classified.

  If $\xi_i > 1$, $\boldsymbol{x_i}$ is misclassified

Convert dual problem into QP notation

$$\min_{\boldsymbol{x}} \frac{1}{2}\boldsymbol{x}^T P\boldsymbol{x} + \boldsymbol{q}^T\boldsymbol{x}$$
$$\text{subject to } \mathrm{G}\boldsymbol{x} \le \boldsymbol{h}$$
$$\mathrm{A}\boldsymbol{x} = \boldsymbol{b}$$

$\Longleftrightarrow$

$$\max_{\boldsymbol{\alpha}} -\left(\frac{1}{2}\underbrace{\boldsymbol{\alpha}^T}_{1\times N}\left(\underbrace{\boldsymbol{y}\otimes\boldsymbol{y}}_{N\times N}\right).*\underbrace{\boldsymbol{K}}_{N\times N}\underbrace{\boldsymbol{\alpha}}_{N\times 1} - \mathbf{1}_N^T\boldsymbol{\alpha}\right)$$
$$\text{subject to } -\boldsymbol{\alpha} \le 0$$
$$\boldsymbol{\alpha} \le C$$
$$\boldsymbol{y}^T\boldsymbol{\alpha}=0$$

$$P = (\boldsymbol{y}\otimes\boldsymbol{y}).*\boldsymbol{X}\boldsymbol{X}^T$$
$$\boldsymbol{q} = -\mathbf{1}_{N\times 1}$$
$$\mathrm{G} = \begin{bmatrix} -\mathbf{I}_{N\times N} \\ \mathbf{I}_{N\times N} \end{bmatrix}$$
$$\boldsymbol{h} = \begin{bmatrix} \mathbf{0}_{N\times 1} \\ C\times\mathbf{1}_{N\times 1} \end{bmatrix}$$
$$\mathrm{A} = \boldsymbol{y}^T$$
$$\boldsymbol{b} = \mathbf{0}$$

Making Predictions in Soft-Margin SVM

- Data associated with $S = \{\boldsymbol{x}_i \mid 0 < \alpha_i \leq C\}$ are SVs for Classes 1 and 2.

  From the optimal $\mathbf{w} = \sum_{i=1}^{N} \alpha_i \, t_i \, \boldsymbol{x}_i \quad (*)$, the decision function is

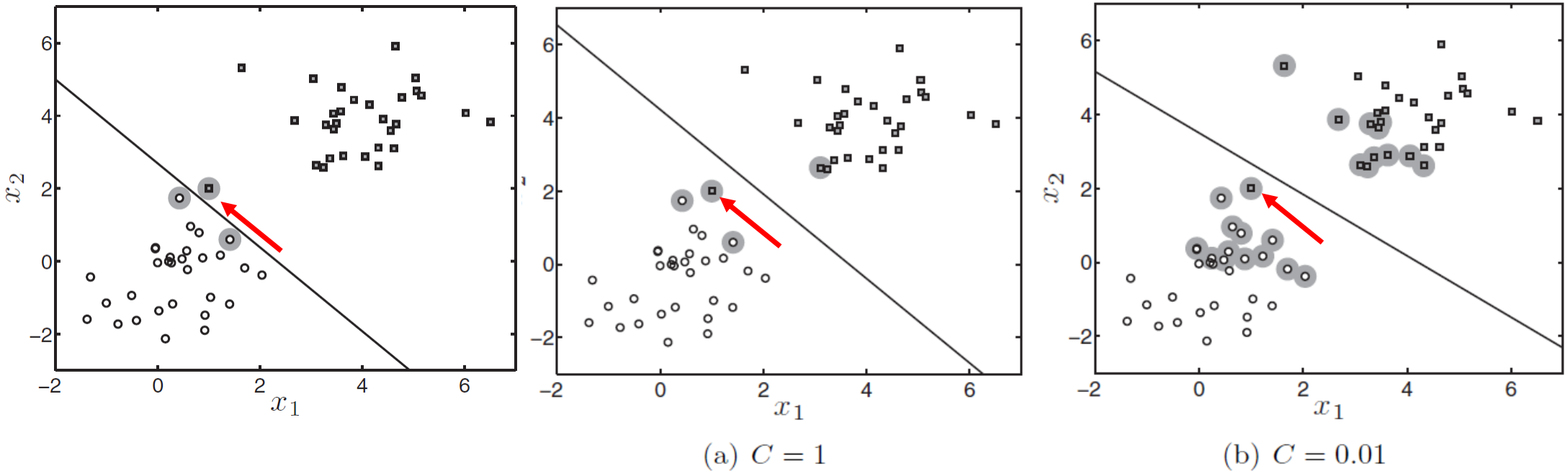
$$D(\boldsymbol{x}_{\text{new}}) = \mathbf{w}^T \boldsymbol{x}_{\text{new}} + b = \sum_i \alpha_i t_i \boldsymbol{x}_i{}^T \boldsymbol{x}_{\text{new}} + b$$

- For the unbounded α_i, $b = t_i - \mathbf{w}^T \boldsymbol{x}_i$ is satisfied.
- However, we can no longer use any SV to compute b, as they will not all satisfy $t_i \, (\boldsymbol{w}^T \boldsymbol{x}_i + b) = 1$. SVs within the margin band or on the wrong side will have $t_i \, (\boldsymbol{w}^T \boldsymbol{x}_i + b) < 1$.
- To overcome this problem, find the SV with the highest value of $\boldsymbol{w}^T \boldsymbol{x}_i$ or $\sum_j \alpha_j t_j \boldsymbol{x}_j{}^T \boldsymbol{x}_i$ to compute b
- Then unknown datum $\boldsymbol{x}_{\text{new}}$ is classified into:

$$\begin{cases} \text{Class 1 , if } D(\boldsymbol{x}_{\text{new}}) > 0 \\ \text{Class 2 , if } D(\boldsymbol{x}_{\text{new}}) < 0 \end{cases}$$

If $D(\boldsymbol{x}_{\text{new}}) = 0$, $\boldsymbol{x}_{\text{new}}$ is on the boundary and thus is unclassifiable

Soft-Margin SVM with two Different *C*



- (a) and (b): Decision boundary and support vectors for a linear SVM with a soft margin for two values of the margin parameter *C*. The influence of the stray support vector has been reduced.
- As *C* decreases, the maximum potential influence of each training point is eroded so more and more of them become active in the decision function

MATLAB script: svmsoft.m

```
% From A First Course in Machine Learning, Chapter 5.
% % Soft margin SVM
clear all;close all;
%% Generate the data
x = [randn(20,2);randn(20,2)+4];
t = [repmat(-1,20,1);repmat(1,20,1)];
% Add a bad point
x = [x;2 1];
t = [t;1];
%% Plot the data
ma = {'ko','ks'};
fc = {[0 0 0],[1 1 1]};
tv = unique(t);
figure(1); hold off
for i = 1:length(tv)
    pos = find(t==tv(i));
    plot(x(pos,1),x(pos,2),ma{i},'markerfacecolor',fc{i});
    hold on
end
```

```
%% Setup the optimisation problem
N = size(x,1);
K = x*x';
H = (t*t').*K + 1e-5*eye(N);
f = repmat(1,N,1);
A = [];b = [];
LB = repmat(0,N,1);
UB = repmat(inf,N,1);
Aeq = t';beq = 0;

%% Loop over various values of the margin parameter
Cvals = [10 5 2 1 0.5 0.1 0.05 0.01];
for cv = 1:length(Cvals);
    %%
    UB = repmat(Cvals(cv),N,1);
    % Following line runs the SVM
    alpha = quadprog(H,-f,A,b,Aeq,beq,LB,UB);
    % Compute the bias
    fout = sum(repmat(alpha.*t,1,N).*K,1)';
    pos = find(alpha>1e-6);
    bias = mean(t(pos)-fout(pos));
```

α_i's > 0 are support vectors

$$Q(\mathbf{w}, b, \boldsymbol{\alpha}) = \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_i \alpha_j \, t_i t_j \boldsymbol{x}_i^T \boldsymbol{x}_j$$

$$\sum_{i=1}^{N} \alpha_i \, y_i = 0, \quad C \geq \alpha_i \geq 0, \qquad i = 1, \ldots, N$$

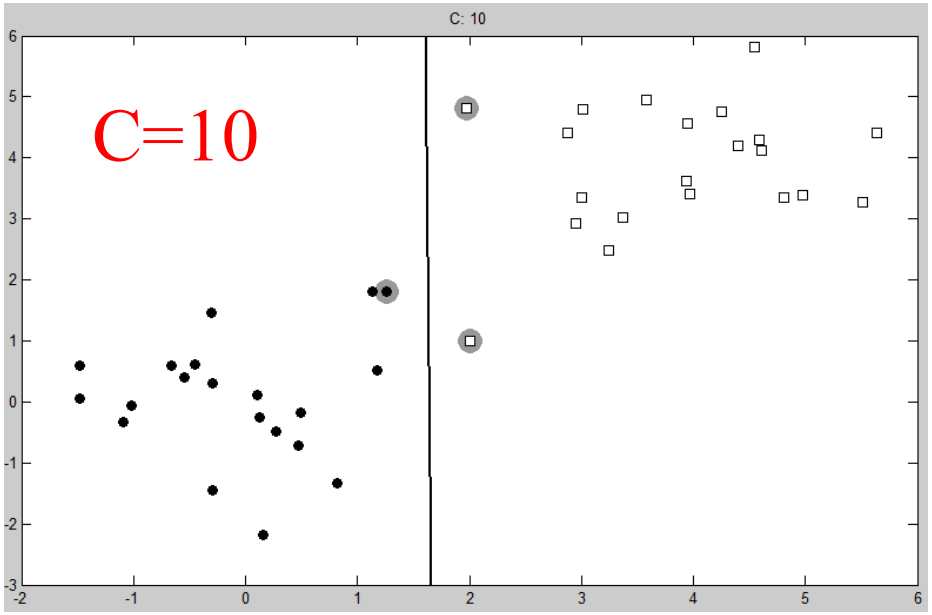
$$\mathbf{w} = \sum_{i=1}^{N} \alpha_i \, t_i \, \boldsymbol{x}_i$$

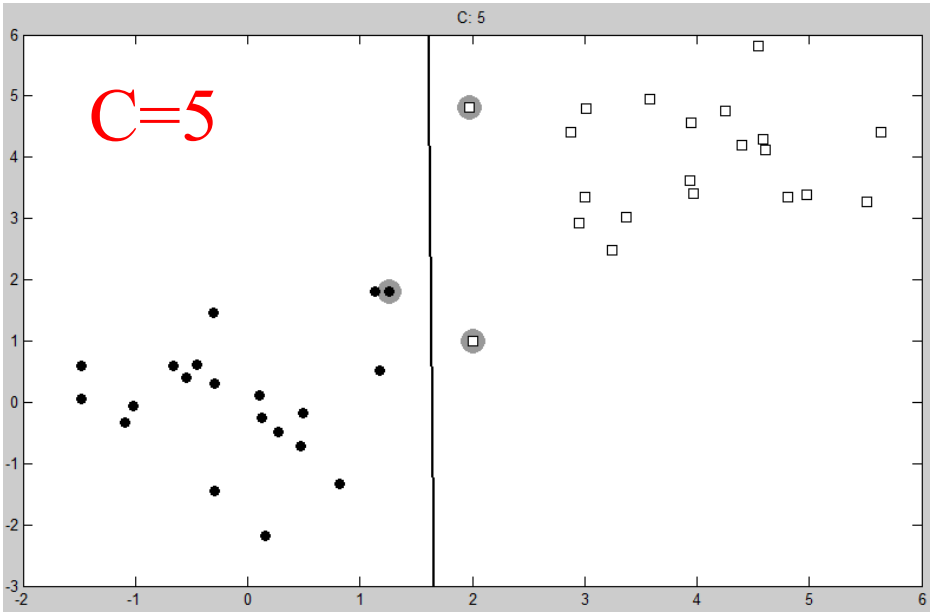
$$b = \frac{1}{\#(\boldsymbol{x}_i \in G)} \sum_{x_i \in G} (y_i - \mathbf{w}^T \boldsymbol{x}_i)$$

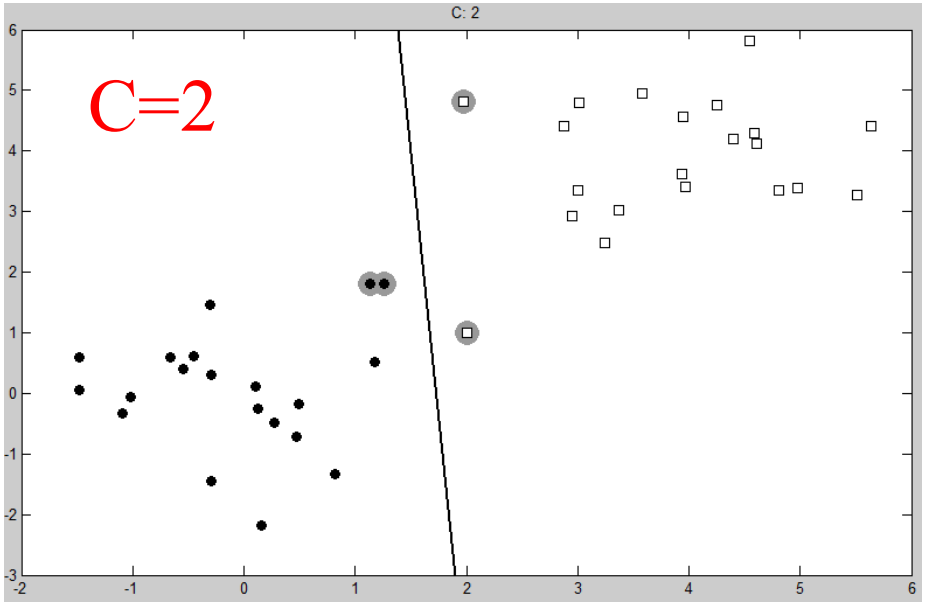
$$G = \{x_i | C > \alpha_i > 0\}$$

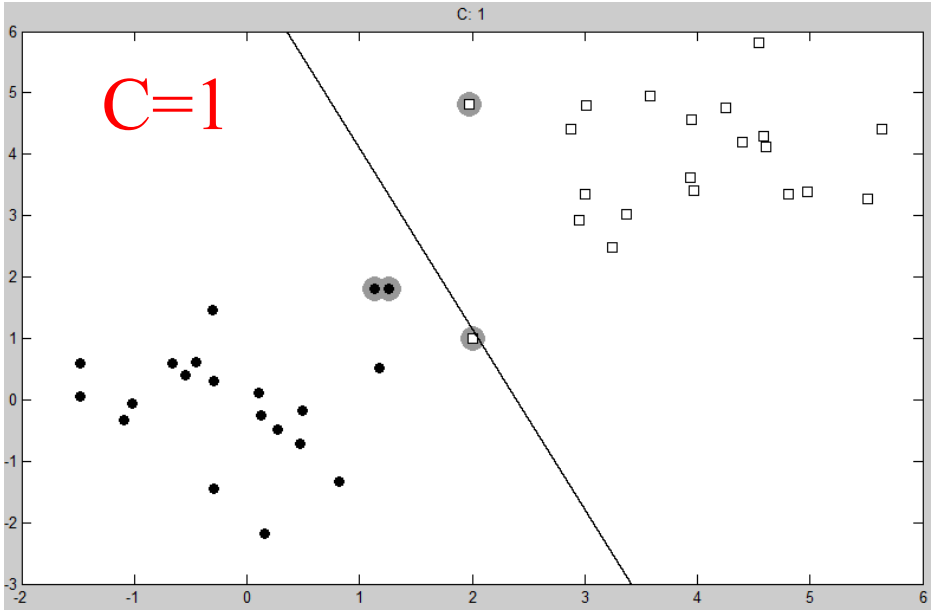
```
 %% Plot the data, decision boundary and Support vectors
    figure(1);hold off
    pos = find(alpha>1e-6);
    plot(x(pos,1),x(pos,2),'ko','markersize',15,'markerfacecolor',[0.6 0.6 0.6],...
        'markeredgecolor',[0.6 0.6 0.6]);
    hold on
    for i = 1:length(tv)
        pos = find(t==tv(i));
        plot(x(pos,1),x(pos,2),ma{i},'markerfacecolor',fc{i});
    end
    xp = xlim;
    yl = ylim;
    % Because this is a linear SVM, we can compute w and plot the decision
    % boundary exactly.
    w = sum(repmat(alpha.*t,1,2).*x,1)';
    yp = -(bias + w(1)*xp)/w(2);
    plot(xp,yp,'k','linewidth',2);
    ylim(yl);
    ti = sprintf('C: %g',Cvals(cv));
    title(ti);
    pause
end
```

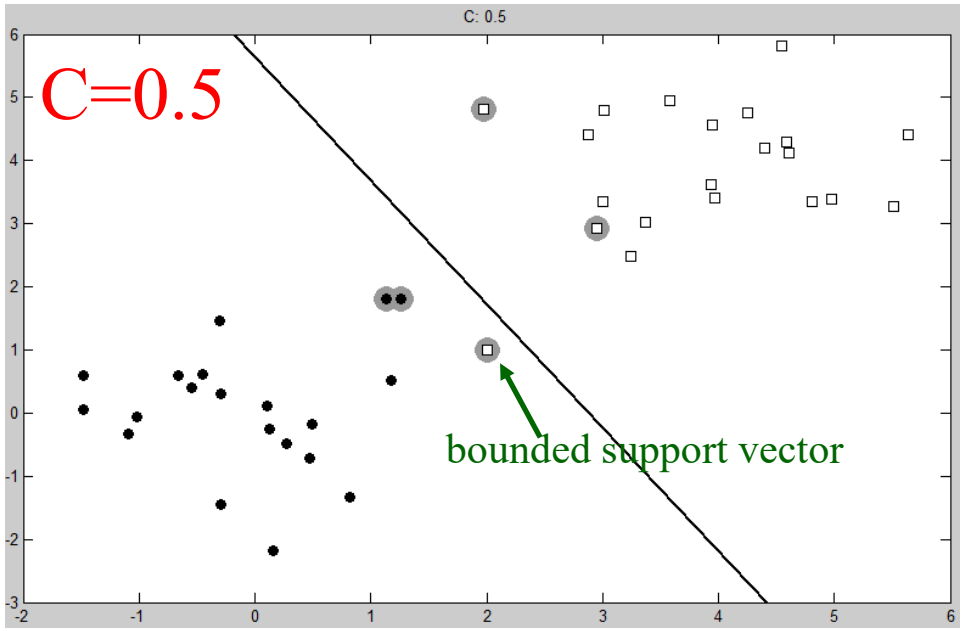
$$\mathbf{w} = \sum_{i=1}^{N} \alpha_i \, t_i \, \boldsymbol{x_i}$$

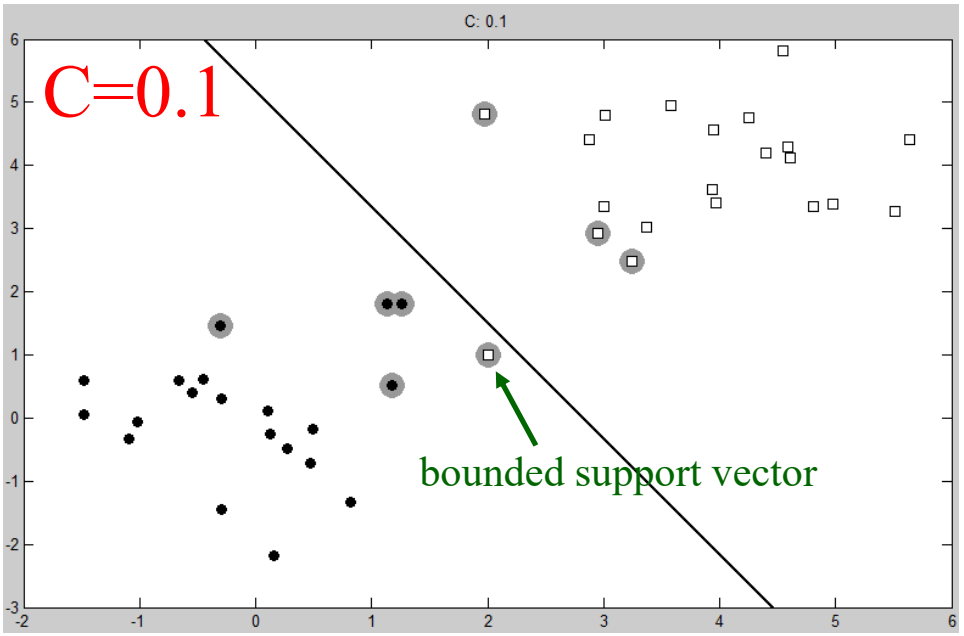

C: 10
C=10

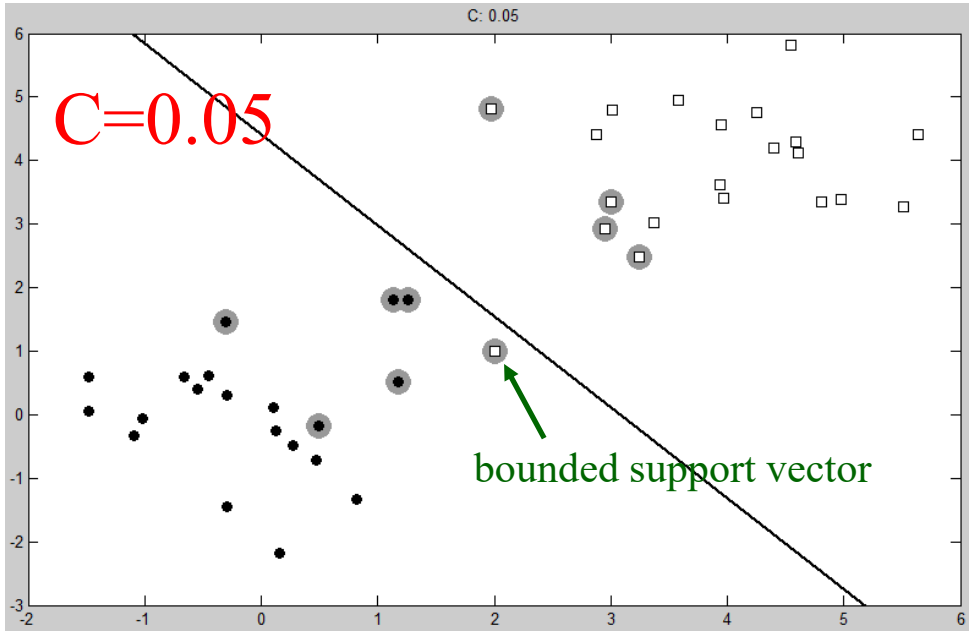

C: 5
C=5

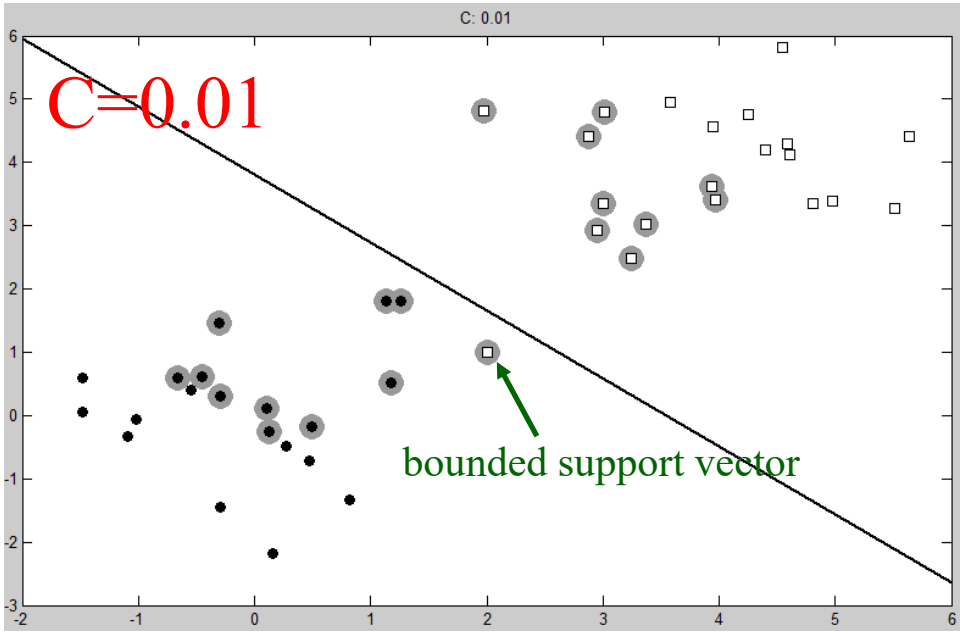

C: 2
C=2


C: 1
C=1


C: 0.5
C=0.5
bounded support vector

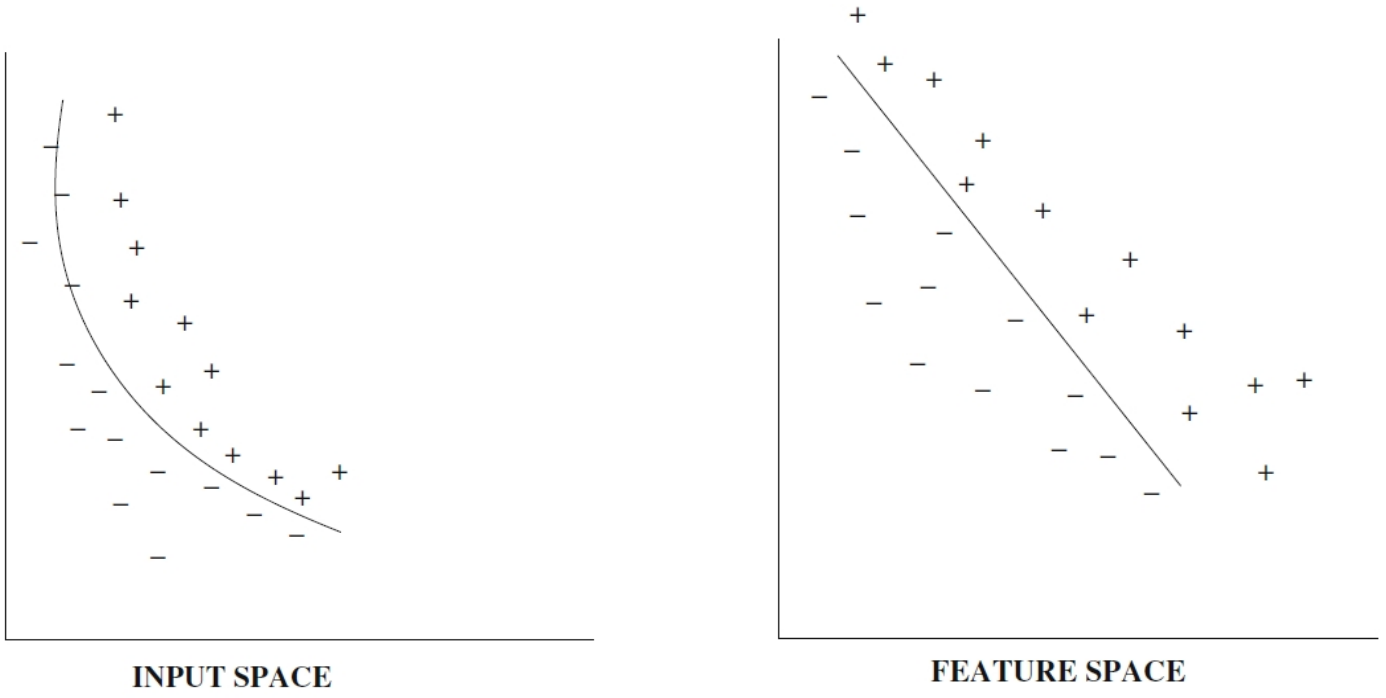

C: 0.1
C=0.1
bounded support vector


C: 0.05
C=0.05
bounded support vector


C: 0.01
C=0.01
bounded support vector

Mapping to a High-Dimensional Space: Kernel Tricks

- If the training data are not linearly separable, to enhance linear separability, the original input space is mapped into a high-dimensional dot-product space called the **feature space**.



Nonlinear decision boundary

Hilbert-Schmidt Theory

- Using a nonlinear $\boldsymbol{g(x)} = (g_1(\boldsymbol{x}), \ldots, g_l(\boldsymbol{x}))^{\mathbf{T}}$, that maps the d-dimensional input vector $\boldsymbol{x}$ into the l-dimensional feature space

- The linear decision function

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

where $\mathbf{w} \in \mathbb{R}^l$ and b is a bias term.

- According to the Hilbert-Schmidt theory, if a symmetric $K(\boldsymbol{x}, \boldsymbol{x}')$ satisfies

$$\sum_{i,j=1}^{N} h_i \, h_j K(\boldsymbol{x_i}, \boldsymbol{x_j}) \geq 0 \qquad ①$$

for all N, $\boldsymbol{x_i}$, and h_i, where $h_i \in \mathbb{R}$, $\exists$ a $\boldsymbol{g}(\boldsymbol{x})$ that maps $\mathbf{x}$ into the dot-product feature space

$$K(\boldsymbol{x}, \boldsymbol{x}') = \boldsymbol{g}(\boldsymbol{x})^{\,T} \boldsymbol{g}(\boldsymbol{x}') \qquad ②$$

Dual Problem in Feature Space Using Kernel

- If ② is satisfied,

$$\sum_{i,j=1}^{N} h_i\, h_j K(\boldsymbol{x_i}, \boldsymbol{x_j}) = \left(\sum_{i=1}^{N} \boldsymbol{g}(\boldsymbol{x_i})^{T} h_i\right)\left(\sum_{j=1}^{N} \boldsymbol{g}(\boldsymbol{x_j}) h_j\right) \geq 0 \qquad ③$$

- ① or ③ is called **Mercer's condition**, and function satisfies ① or ③ is called positive semidefinite kernel or the Mercer kernel or simply the kernel.

- Using the kernel, the dual problem in the feature space is

Maximize $Q(\boldsymbol{\alpha}) = \sum_{i=1}^{N} \alpha_i - \frac{\mathbf{1}}{\mathbf{2}} \sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_i \alpha_j\, t_i t_j K(\boldsymbol{x_i}, \boldsymbol{x_j})$

subject to $\sum_{i=1}^{N} \alpha_i\, t_i = 0,\ 0 \leq \alpha_i \leq \mathrm{C}, i = 1, \ldots, N$

- Because $K(\boldsymbol{x}, \boldsymbol{x'})$ is a positive semidefinite kernel, the optimization problem is a convex quadratic programming problem.

Making Predictions in Nonlinear SVM

- Decision function is

$$D(\boldsymbol{x}_{\text{new}}) = \mathbf{w}^T \boldsymbol{g}(\boldsymbol{x}_{\text{new}}) + b = \sum_{x_i \in S} \alpha_i t_i K(\boldsymbol{x_i}, \boldsymbol{x}_{\text{new}}) + b$$

$b = t_i - \sum_j \alpha_j t_j K(\boldsymbol{x_j}, \boldsymbol{x_i})$, $\boldsymbol{x_i}$ is an unbounded support vector

- Similar to the soft-margin SVM, we find the SV with the highest value of $\sum_j \alpha_j t_j K(\boldsymbol{x_j}^T \boldsymbol{x}_i)$ to compute b

- Then unknown datum $\boldsymbol{x}_{\text{new}}$ is classified into:

$$\begin{cases} \text{Class 1 , if } D(\boldsymbol{x}_{\text{new}}) > 0 \\ \text{Class 2 , if } D(\boldsymbol{x}_{\text{new}}) < 0 \end{cases}$$

If $D(\boldsymbol{x}_{\text{new}}) = 0$, $\boldsymbol{x}_{\text{new}}$ is unclassifiable

Kernels used in SVM

- **Linear Kernels:**

If the problem is linearly separable, we use linear kernels: $K(\boldsymbol{x}, \boldsymbol{x}') = \boldsymbol{x}^{\boldsymbol{T}}\boldsymbol{x}'$

- **Polynomial Kernels:**

The polynomial kernel with degree $m \geq 1$ is $K(\boldsymbol{x}, \boldsymbol{x}') = (\boldsymbol{x}^{\boldsymbol{T}}\boldsymbol{x}' + 1)^{\boldsymbol{m}}$

When $m = 1$, the kernel is the linear kernel by adjusting 1 into b

When $m = 2, d = 2$,

$$K(\boldsymbol{x}, \boldsymbol{x}') = 1 + 2x_1x_1' + 2x_2x_2' + 2x_1x_1'x_2x_2' + x_1^2x_1'^2 + x_2^2x_2'^2$$

$$= \boldsymbol{g}(\boldsymbol{x})^{T}\boldsymbol{g}(\boldsymbol{x}') \geq 0 \quad \text{satisfy Mercer's condition}$$

where $\boldsymbol{g}(\boldsymbol{x}) = (1, \sqrt{2}x_1, \sqrt{2}x_2, \sqrt{2}x_1x_2, x_1^2, x_2^2)^{\boldsymbol{T}}$

- In general, polynomial kernels satisfy Mercer's condition

Kernels used in SVM

- **Radial Basis Function (RBF) Kernels:**

$K(\boldsymbol{x}, \boldsymbol{x}') = \exp(-\gamma||\boldsymbol{x} - \boldsymbol{x}'||^2), \gamma > 0$ controlling the radius

$$= \exp(-\gamma||\boldsymbol{x}||^2)\exp(-\gamma||\boldsymbol{x}'||^2)\exp(2\gamma\boldsymbol{x}^T\boldsymbol{x}') \qquad (*)$$

Because $\exp(2\gamma\boldsymbol{x}^T\boldsymbol{x}')=1+2\gamma\boldsymbol{x}^T\boldsymbol{x}' + 2\gamma^2(\boldsymbol{x}^T\boldsymbol{x}')^2 + \frac{2\gamma^3}{3!}(\boldsymbol{x}^T\boldsymbol{x}')^3 + \cdots$

is an infinite summation of polynomials ⇒ it is a kernel.

$\exp(-\gamma\left|\left|\boldsymbol{x}\right|\right|^2)$ and $\exp(-\gamma||\boldsymbol{x}'||^2)$ are proved to be kernels and the product of kernels is also a kernel. Thus $(*)$ is a kernel.

- The decision function is

$D(\boldsymbol{x}_{\text{new}}) = \sum_{x_i \in S} \alpha_i t_i K(\boldsymbol{x_i}, \boldsymbol{x}_{\text{new}}) + b = \sum_{x_i \in S} \alpha_i t_i \exp(-\gamma||\boldsymbol{x_i} - \boldsymbol{x}_{\text{new}}||^2) + b$

Here, the support vectors are the centers of the radial basis functions.

Convert dual problem into QP notation

$$\min_{\boldsymbol{x}} \frac{1}{2}\boldsymbol{x}^T P\boldsymbol{x} + \boldsymbol{q}^T\boldsymbol{x}$$
$$\text{subject to } \mathrm{G}\boldsymbol{x} \le \boldsymbol{h}$$
$$\mathrm{A}\boldsymbol{x} = \boldsymbol{b}$$

$\Longleftrightarrow$

$$\max_{\boldsymbol{\alpha}} -\left(\frac{1}{2}\underbrace{\boldsymbol{\alpha}^T}_{1\times N}\left(\underbrace{\boldsymbol{y}\otimes\boldsymbol{y}}_{N\times N}\right).*\underbrace{\boldsymbol{K}}_{N\times N}\underbrace{\boldsymbol{\alpha}}_{N\times 1} - \mathbf{1}_N^T\boldsymbol{\alpha}\right)$$
$$\text{subject to } -\boldsymbol{\alpha} \le 0$$
$$\boldsymbol{\alpha} \le C$$
$$\boldsymbol{y}^T\boldsymbol{\alpha}=0$$

$$P = (\boldsymbol{y}\otimes\boldsymbol{y}).*\boldsymbol{K}$$
$$\boldsymbol{q} = -\mathbf{1}_{N\times 1}$$
$$\mathrm{G} = \begin{bmatrix} -\mathbf{I}_{N\times N} \\ \mathbf{I}_{N\times N} \end{bmatrix}$$
$$\boldsymbol{h} = \begin{bmatrix} \mathbf{0}_{N\times 1} \\ C\times\mathbf{1}_{N\times 1} \end{bmatrix}$$
$$\mathrm{A} = \boldsymbol{y}^T$$
$$\boldsymbol{b} = \mathbf{0}$$

MATLAB script: svmgauss.m

```
%% svmgauss.m
% From A First Course in Machine Learning, Chapter 5.
% SVM with Gaussian kernel
clear all;close all;
%% Load the data
load t.csv
load X.csv
% Put in class order for visualising the kernel
[t I] = sort(t);
X = X(I,:);

%% Plot the data
ma = {'ko','ks'};
fc = {[0 0 0],[1 1 1]};
tv = unique(t);
figure(1); hold off
for i = 1:length(tv)
    pos = find(t==tv(i));
    plot(X(pos,1),X(pos,2),ma{i},'markerfacecolor',fc{i});hold on
    pause
end
```

```
%% Compute Kernel and test Kernel
[Xv Yv] = meshgrid(-3:0.1:3,-3:0.1:3);
testX = [Xv(:) Yv(:)];
N = size(X,1);
Nt = size(testX,1);
K = zeros(N);
testK = zeros(N,Nt);
% Set kernel parameter
gamvals = [0.01 0.1 1 5 10 50];
for gv = 1:length(gamvals)
    %%
    gam = gamvals(gv);
    for n = 1:N
        for n2 = 1:N
            K(n,n2) = exp(-gam*sum((X(n,:)-X(n2,:)).^2));
        end
        for n2 = 1:Nt
            testK(n,n2) = exp(-gam*sum((X(n,:)-testX(n2,:)).^2));
        end
    end
    figure(1);hold off
    imagesc(K);
    ti = sprintf('Gamma: %g',gam);
    title(ti);
```

$$K(\boldsymbol{x}_i, \boldsymbol{x}_j) = \exp(-\gamma ||\boldsymbol{x}_i - \boldsymbol{x}_j||^2)$$

```
% Construct the optimisation
    H = (t*t').*K + 1e-5*eye(N);
    f = repmat(1,N,1);
    A = [];b = [];
    LB = repmat(0,N,1);
    UB = repmat(inf,N,1);
    Aeq = t';beq = 0;

    % Fix C
    C = 10;
    UB = repmat(C,N,1);
    % Following line runs the SVM
    alpha = quadprog(H,-f,A,b,Aeq,beq,LB,UB);

    fout = sum(repmat(alpha.*t,1,N).*K,1)';
    pos = find(alpha>1e-6);
    bias = mean(t(pos)-fout(pos));

    % Compute the test predictions
    testpred = (alpha.*t)'*testK + bias;
    testpred = testpred';
```

$$Q(\mathbf{w}, b, \boldsymbol{\alpha}) = \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_i \alpha_j \, t_i t_j K(\boldsymbol{x}_i, \boldsymbol{x}_j)$$

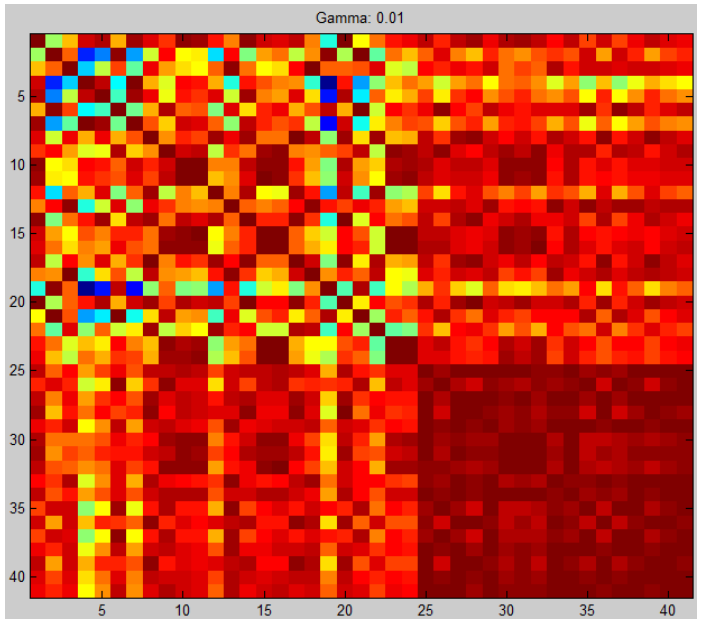
$$\sum_{i=1}^{N} \alpha_i \, y_i = 0, \quad C \geq \alpha_i \geq 0, \qquad i = 1, \ldots, N$$

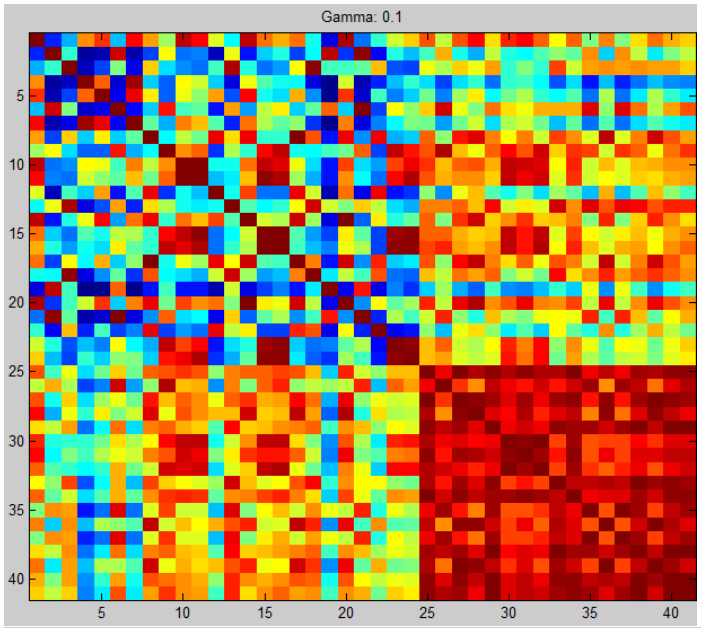
α_i's > 0 are support vectors

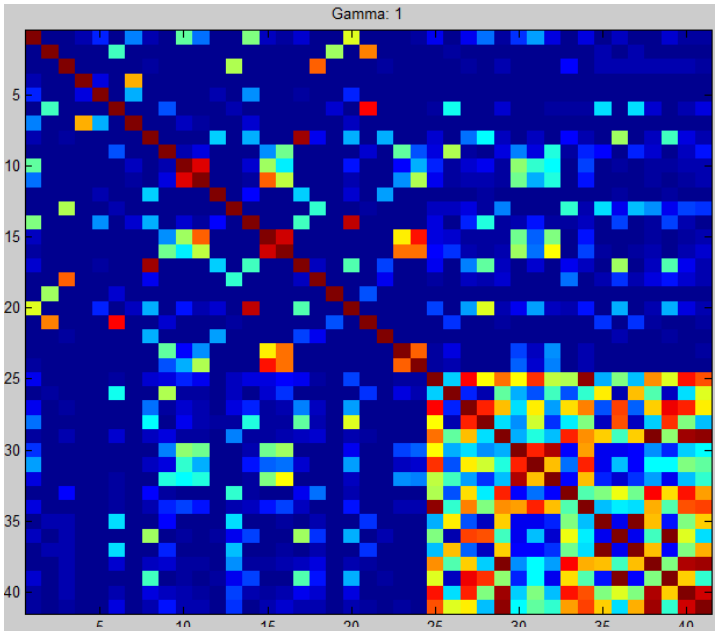
$$\sum_{x_i \in S} \alpha_i t_i \exp(-\gamma ||\boldsymbol{x}_i - \boldsymbol{x}||^2) + b$$

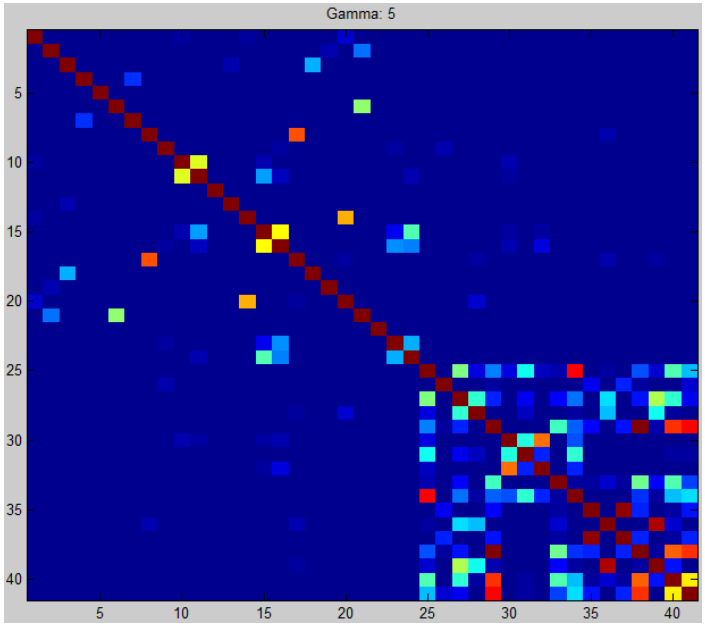
```
 % Plot the data, support vectors and decision boundary
    figure(2);hold off
    pos = find(alpha>1e-6);
    plot(X(pos,1),X(pos,2),'ko','markersize',15,'markerfacecolor',[0.6 0.6 0.6],...
        'markeredgecolor',[0.6 0.6 0.6]);
    hold on
    for i = 1:length(tv)
        pos = find(t==tv(i));
        plot(X(pos,1),X(pos,2),ma{i},'markerfacecolor',fc{i});
    end
    contour(Xv,Yv,reshape(testpred,size(Xv)),[0 0],'k');
    ti = sprintf('Gamma: %g',gam);
    title(ti);
    pause
end
```

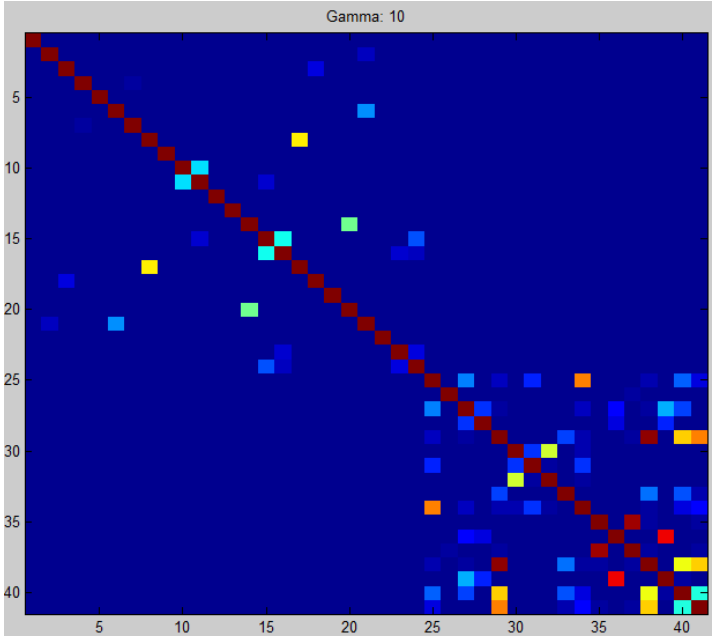
α_i's > 0 are support vectors

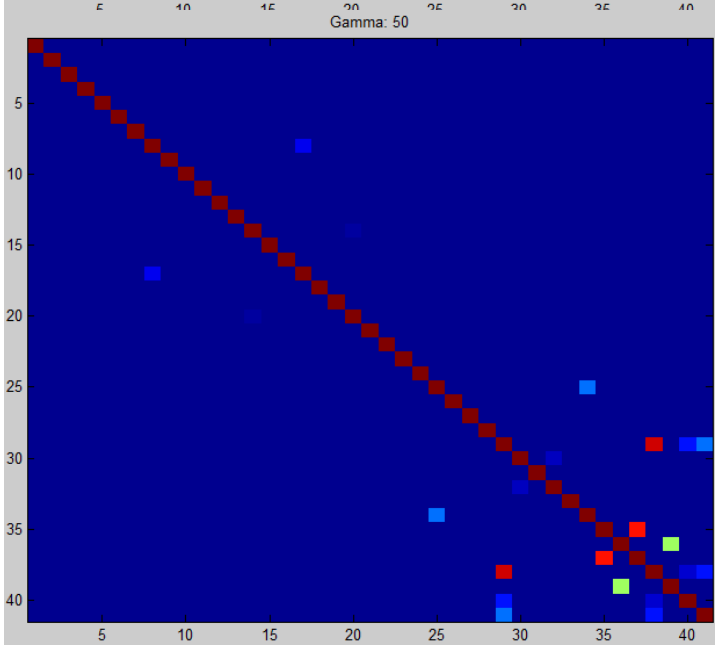

Gamma: 0.01

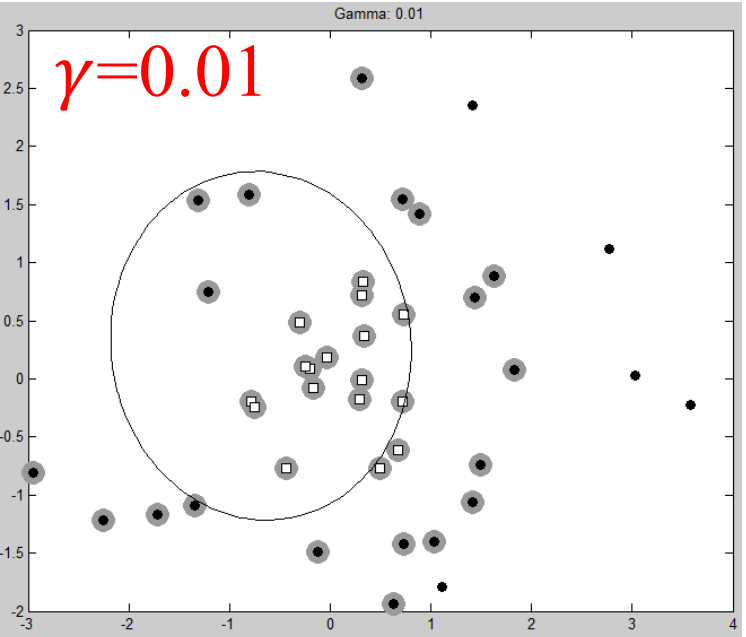

Gamma: 0.1

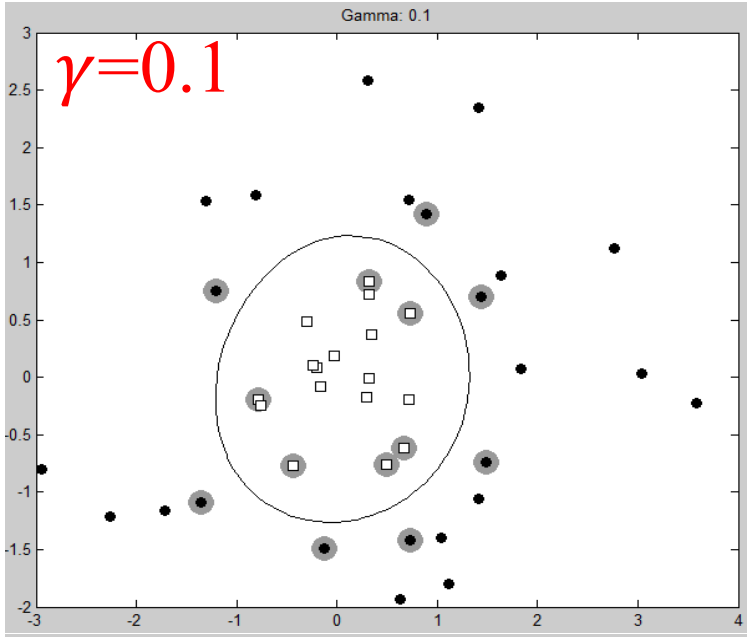

Gamma: 1

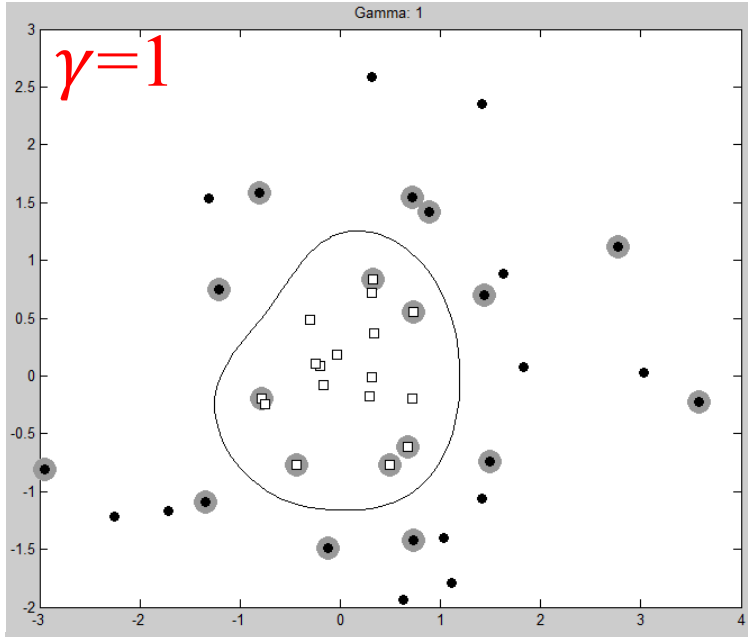

Gamma: 5

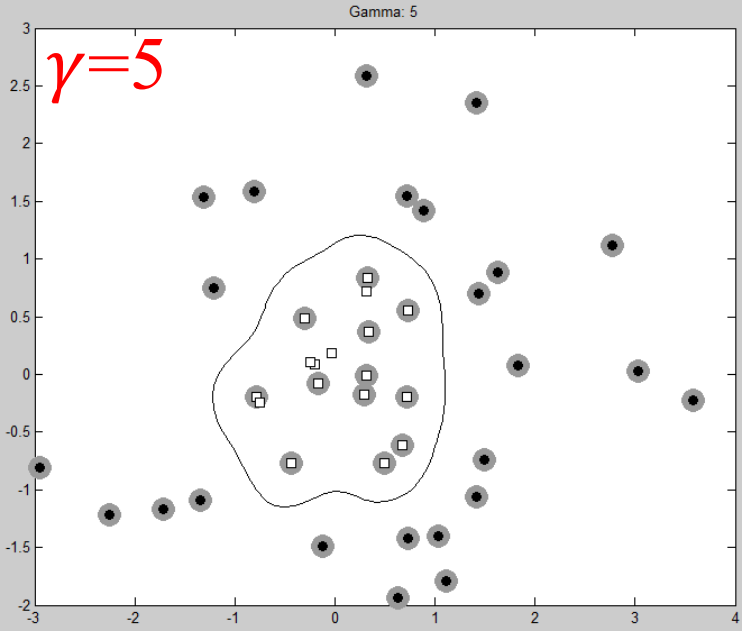

Gamma: 10

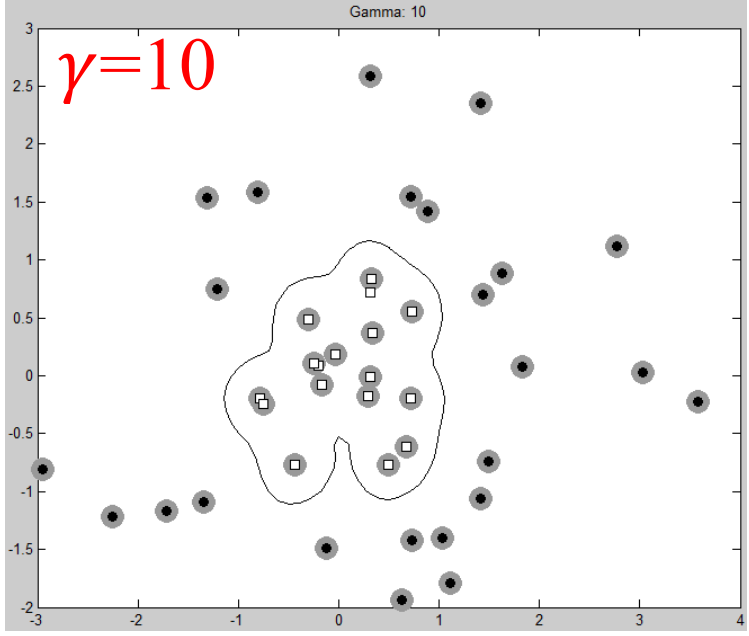

Gamma: 50

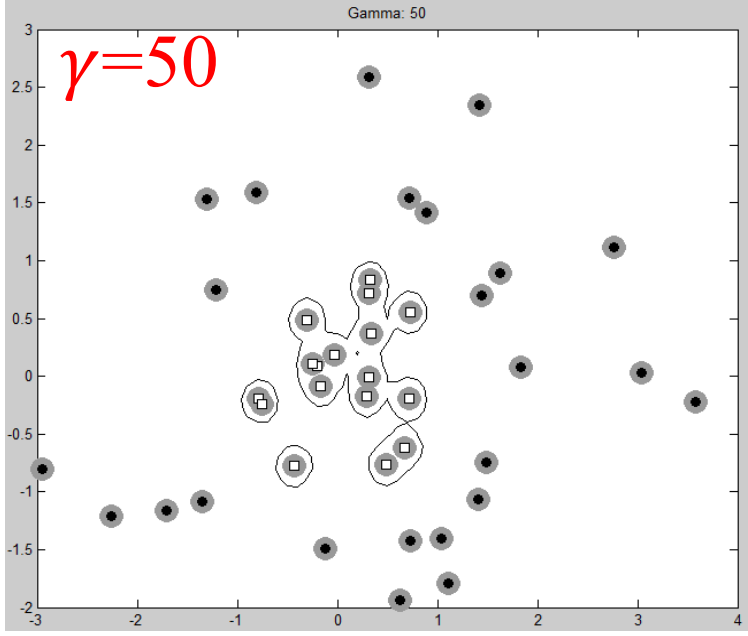

Gamma: 0.01
γ=0.01


Gamma: 0.1
γ=0.1


Gamma: 1
γ=1


Gamma: 5
γ=5


Gamma: 10
γ=10


Gamma: 50
γ=50

Summary of Kernel Trick

- A kernel function, $H : \mathbb{R}^{l} \times \mathbb{R}^{l} \rightarrow \mathbb{R}$ where $K(\boldsymbol{x}, \boldsymbol{x}') = \boldsymbol{g}(\boldsymbol{x})^{T} \boldsymbol{g}(\boldsymbol{x}')$

- $\mathbf{w} = \sum_{x_i \in S} \alpha_i t_i \, \boldsymbol{g}(\boldsymbol{x}_i)$, where S is the set of support vectors.

- Given a test pattern $\boldsymbol{x}_{\text{new}}$, we can classify it based on $D(\boldsymbol{x}_{\text{new}}) = \mathbf{w}^{T} \boldsymbol{g}(\boldsymbol{x}_{\text{new}}) + b$ by $\sum_{x_i \in S} \alpha_i t_i \, \boldsymbol{g}(\boldsymbol{x}_i)^{T} \boldsymbol{g}(\boldsymbol{x}_{\text{new}}) + b$

- b is obtained by

  $b = t_i - \sum_j \alpha_j t_j \, K(\boldsymbol{x}_j, \boldsymbol{x}_i)$, $\boldsymbol{x}_i$ is a good support vector

Assessing Classification Performance

- Sensitivity (True positive Rate) is the proportion of the diseased people (TP +FN) that we correctly classify as being diseased (TP).
- Specificity (True Negative Rate) is the proportion of all of the healthy people (TN + FP) that we correctly classify as being healthy (TN).
- Consider the rare disease example, if we diagnosed everyone as healthy, we have a specificity of 1, but a sensitivity of 0 which is very bad.
- We do not want to misdiagnose any diseased people but tolerate some healthy people as diseased, we reduce specificity to increase sensitivity
- The area under the receiver operating characteristic (ROC) curve combine sensitivity and specificity into a single value.

		True Status			
		Yes	No		
Predicted status	Yes	True Positive (TP)	False Positive (FP) Type I error	Positive Predictive Rate, Precision TP/(TP+FP)	False Discovery Rate FP/(TP+FP)
	No	False Negative (FN) Type II error	True Negative (TN)	False Omission Rate FN/(FN+TN)	Negative Predictive Rate TN/(FN+TN)
Total number		True positive Rate Sensitivity, Recall TP/(TP+FN)	False positive Rate FP/(FP+TN)	F1 score =2*precision*Recall/ (precision+Recall)	
Accuracy (TP+TN)/T		False Negative Rate FN/(TP+FN)	True Negative Rate Specificity TN/(FP+TN)		

MATLAB script: svmroc.m

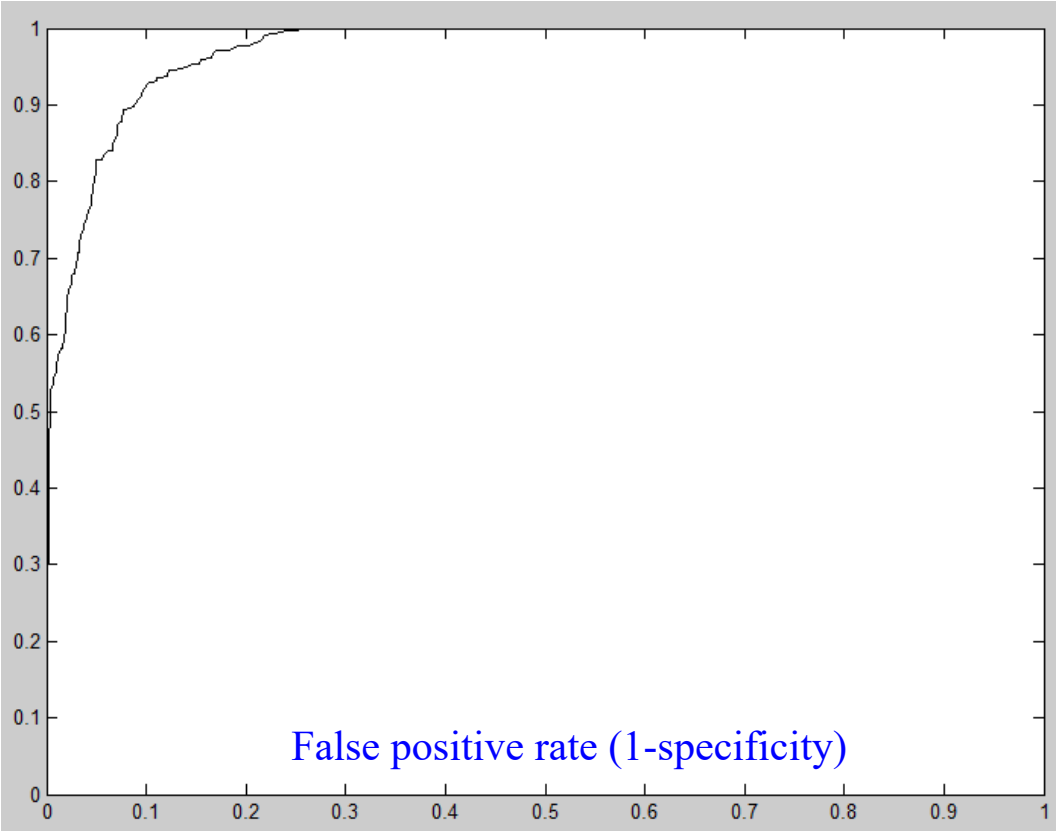
```
%% svmroc.m
% From A First Course in Machine Learning, Chapter 5.
% ROC analysis of SVM
clear all;close all;
%% Load the data
load ../data/SVMdata2/X.csv
load ../data/SVMdata2/t.csv
load ../data/SVMtest/testX.csv
load ../data/SVMtest/testt.csv
%% Compute the kernels
gam = 10; % Experiment with this value
N = size(X,1);
Nt = size(testX,1);
for n = 1:N
    for n2 = 1:N
        K(n,n2) = exp(-gam*sum((X(n,:)-X(n2,:)).^2));
    end
    for n2 = 1:Nt
        testK(n,n2) = exp(-gam*sum((X(n,:)-testX(n2,:)).^2));
    end
end
```

```
%% Train the SVM
H = (t*t').*K + 1e-5*eye(N);
f = repmat(1,N,1);
A = [];b = [];
LB = repmat(0,N,1); UB = repmat(inf,N,1);
Aeq = t';beq = 0;

% Fix C
C = 10;
UB = repmat(C,N,1);
% Following line runs the SVM
alpha = quadprog(H,-f,A,b,Aeq,beq,LB,UB);

fout = sum(repmat(alpha.*t,1,N).*K,1)';
pos = find(alpha>1e-6);
bias = mean(t(pos)-fout(pos));

%% Compute the test predictions
testpred = (alpha.*t)'*testK + bias;
testpred = testpred';
```

```
%% Do the ROC analysis
th_vals = [min(testpred):0.01:max(testpred)+0.01];
sens = []; spec = [];
for i = 1:length(th_vals)
    b_pred = testpred>=th_vals(i);
    % Compute true positives, false positives, true negatives, true
    % positives
    TP = sum(b_pred==1 & testt == 1);
    FP = sum(b_pred==1 & testt == -1);
    TN = sum(b_pred==0 & testt == -1);
    FN = sum(b_pred==0 & testt == 1);
    % Compute sensitivity and specificity
    sens(i) = TP/(TP+FN);
    spec(i) = TN/(TN+FP);
end
%% Plot the ROC curve
figure(1);hold off
cspec = 1-spec;
cspec = cspec(end:-1:1);
sens = sens(end:-1:1);
plot(cspec,sens,'k')
%% Compute the AUC
AUC = sum(0.5*(sens(2:end)+sens(1:end-1)).*(cspec(2:end) - cspec(1:end-1)));
fprintf('\n AUC: %g\n',AUC);
```

ROC curve (svmroc.m)

True positive rate (sensitivity)



The ROC curve traces out two types of error as we vary the threshold value for the prediction values $\sum_{x_i \in S} \alpha_i y_i \exp(-\gamma ||\boldsymbol{x_i} - \boldsymbol{x}||^2) + b$. The actual thresholds are not shown. The true positive rate is the sensitivity: the fraction of test data (labeled 1) that are correctly identified, using a given threshold value. The false positive rate is 1-specificity: the fraction of test data (labeled -1) that we classify incorrectly as 1, using that same threshold value. The ideal ROC curve hugs the top left corner, indicating a high true positive rate and a low false positive rate.

Confusion matrix

TABLE 5.3 Confusion matrix for the 20-class newsgroup data.

Predicted class \ True class	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
1	242	3	3	0	1	0	0	1	0	4	2	0	2	10	4	7	1	12	7	47
2	0	296	33	8	8	42	9	1	1	0	0	4	18	7	8	2	0	1	1	3
3	0	6	209	15	9	8	4	0	0	0	0	1	0	1	0	1	0	0	0	0
4	0	12	60	303	36	12	46	2	0	1	0	1	28	3	0	0	0	0	0	0
5	0	8	10	22	277	2	21	0	0	1	0	2	7	0	0	1	1	0	0	0
6	1	21	30	2	2	304	0	1	0	3	0	1	3	0	1	2	0	0	1	0
7	0	1	0	5	5	1	235	5	1	2	0	1	1	0	0	0	1	0	0	0
8	0	3	1	6	4	0	31	356	25	3	1	0	9	4	0	0	2	2	1	0
9	0	2	2	0	1	2	5	4	353	1	0	0	2	0	1	0	1	1	0	1
10	0	0	2	0	1	1	0	2	2	348	4	0	0	1	0	0	1	1	0	0
11	1	0	1	1	0	0	1	0	0	16	382	0	1	0	1	0	1	1	0	0
12	1	16	16	5	4	10	3	1	1	2	0	360	45	0	4	1	3	4	3	1
13	1	4	1	24	16	0	9	5	1	2	0	3	260	3	4	0	0	0	0	0
14	2	3	4	0	8	0	2	0	1	0	2	2	6	324	4	1	1	0	3	3
15	3	7	4	1	2	3	3	2	0	0	1	0	4	3	336	0	2	0	7	5
16	39	4	5	0	0	1	3	1	1	3	2	2	5	17	4	376	3	7	2	68
17	4	0	0	0	3	1	1	5	4	1	0	9	0	3	1	3	325	3	95	19
18	7	1	0	0	0	1	3	1	2	2	1	0	2	6	2	1	2	325	4	5
19	7	2	9	0	6	2	5	8	5	8	4	8	0	10	21	1	16	19	185	7
20	10	0	1	0	0	0	1	0	0	0	0	1	0	1	1	2	4	0	1	92

- High values on the diagonals tell us, on the whole, the classifier is doing pretty well.
- High off-diagonal elements tell us about mistakes that are being made.

Linear Model

- $\mathbf{w}^T \boldsymbol{x} - b = 0$
- $\mathbf{w}^T \boldsymbol{x_i} - b \begin{cases} > 1 & \text{for } y_i = 1, \\ < -1 & \text{for } y_i = -1 \end{cases}$ is equivalent to
- $y_i(\mathbf{w}^T \boldsymbol{x_i} - b) \geq 1, i = 1, \ldots, N$

Cost Function with data within and outside the margin boundary

- For data points that are on the correct side of the margin boundary satisfy $y_i(\mathbf{w}^T \boldsymbol{x_i} - b) \geq 1$, we have $\xi_i = 0$, and for the remaining points we have $\xi_i = 1 - y_i(\mathbf{w}^T \boldsymbol{x_i} - b)$.

- Thus the objective function can be written in the form

$$Q(\mathbf{w}, \mathrm{b}, \boldsymbol{\xi}) = \lambda||\mathbf{w}||^2 + \frac{1}{N}\sum_{i=1}^{N} \max(0, 1 - y_i(\mathbf{w}^T \boldsymbol{x_i} - b))$$

where $$\max(0, 1 - y_i(\mathbf{w}^T \boldsymbol{x_i} - b)) = \begin{cases} 0, & \text{if } y_i(\mathbf{w}^T \boldsymbol{x_i} - b) \geq 1, \\ y_i(\mathbf{w}^T \boldsymbol{x_i} - b), & \text{otherwise} \end{cases}$$

is the hinge loss

Cost function with regularization

- Minimize

$$l(\boldsymbol{w}, b) = \lambda||\mathbf{w}||^2 + \frac{1}{N}\sum_{i=1}^{N} \max(0, 1 - y_i(\mathbf{w}^T \boldsymbol{x_i} - b))$$

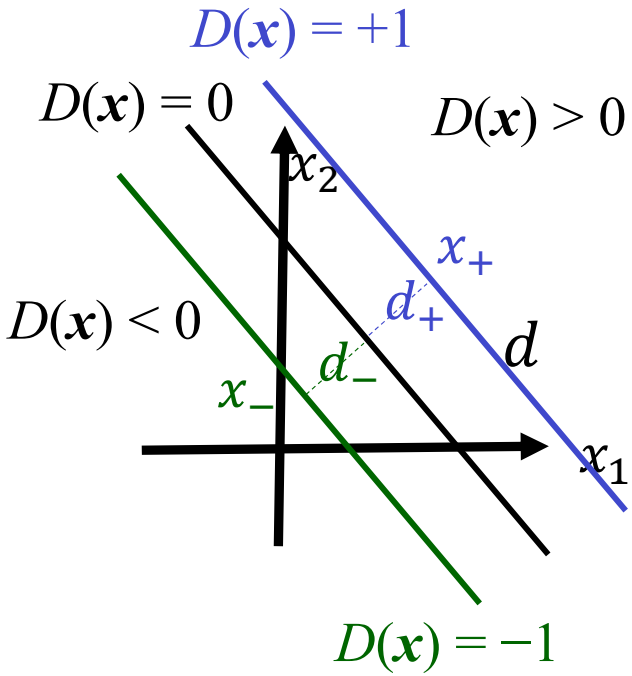
with respect to **w** and b.

- if $y_i(\mathbf{w}^T \boldsymbol{x_i} - b) \geq 1$

$$l(\boldsymbol{w}, b) = \lambda||\mathbf{w}||^2$$

else

$$l(\boldsymbol{w}, b) = \lambda||\mathbf{w}||^2 + 1 - y_i(\mathbf{w}^T \boldsymbol{x_i} - b)$$



Gradient Descent of w and b

- if $y_i(\mathbf{w}^T \boldsymbol{x_i} - b) \geq 1$

$$\frac{\partial l}{\partial \boldsymbol{w}} = 2\lambda \boldsymbol{w}$$

$$\frac{\partial l}{\partial b} = 0$$

else

$$\frac{\partial l}{\partial \boldsymbol{w}} = 2\lambda \boldsymbol{w} - y_i \boldsymbol{x_i}$$

$$\frac{\partial l}{\partial b} = y_i$$

- Update rule: for each sample $\boldsymbol{x_i}$, $i = 1, \ldots, N$

$$\boldsymbol{w} = \boldsymbol{w} - \alpha \frac{\partial l}{\partial \boldsymbol{w}}$$

$$b = b - \alpha \frac{\partial l}{\partial b}$$

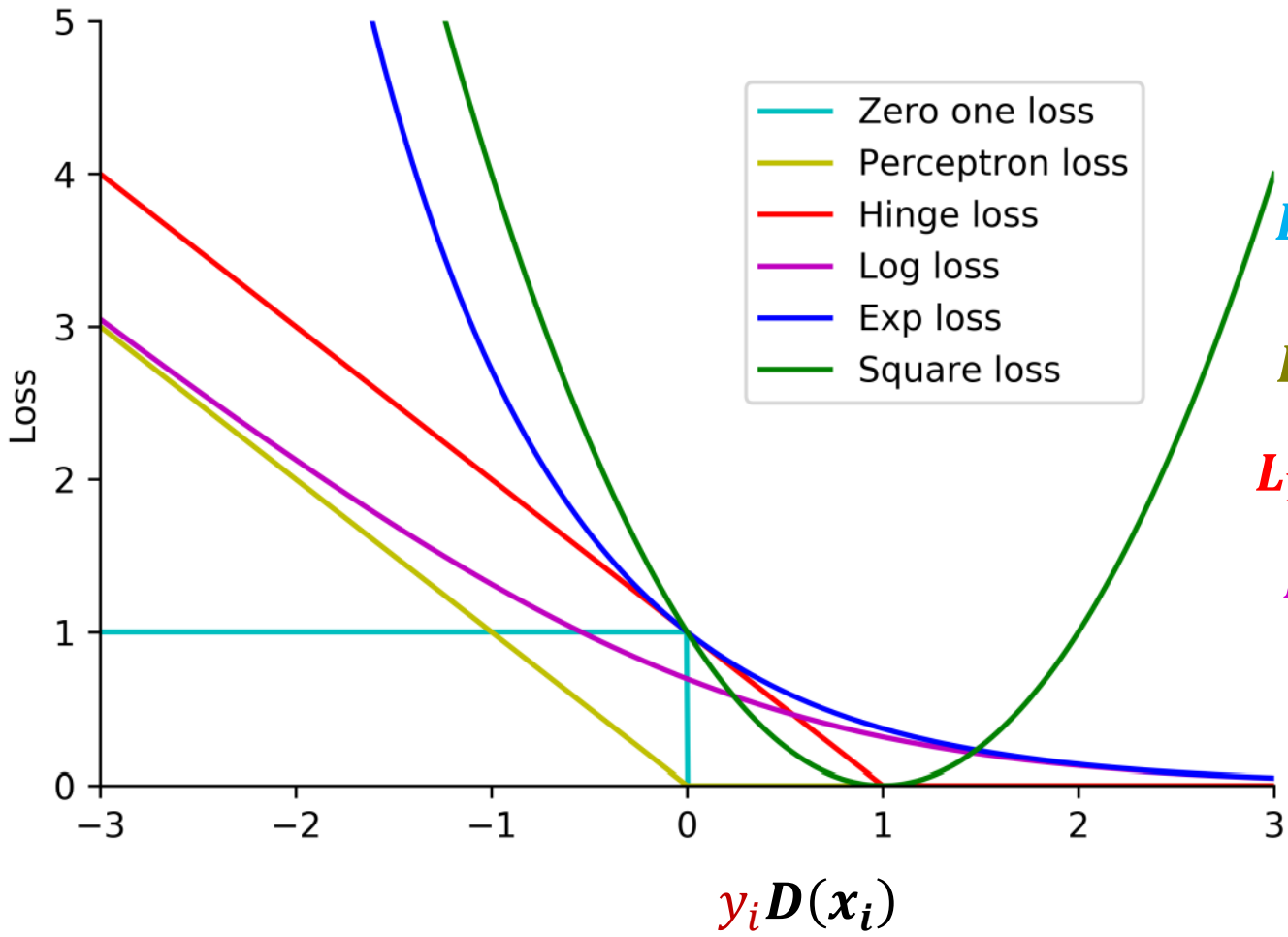
Implementation trick

$$\underbrace{\boldsymbol{X}}_{N_i \times d} = \begin{bmatrix} \boldsymbol{x}_1^T \\ \vdots \\ \boldsymbol{x}_{\boldsymbol{N_i}}^T \end{bmatrix} = \begin{bmatrix} x_{1,1} & x_{1,2} & \cdots & x_{1,d} \\ x_{2,1} & x_{2,2} & \cdots & x_{2,d} \\ \vdots & \vdots & \ddots & \vdots \\ x_{N_i,1} & x_{N_i,2} & \cdots & x_{N_i,d} \end{bmatrix}, \boldsymbol{w} = \begin{bmatrix} w_1 \\ \vdots \\ w_d \end{bmatrix}$$

$$\mathbf{w}^T \boldsymbol{x_i} - b = \boldsymbol{x}_i^T \boldsymbol{w} - b$$

$$\Rightarrow \quad \boldsymbol{Xw} - \boldsymbol{b}, \ \boldsymbol{b} = \begin{bmatrix} b \\ \vdots \\ b \end{bmatrix}$$

Comparison of different loss functions for binary classification



The loss function to be minimized is

$$\frac{1}{N}\sum_{i=1}^{N} L_{\mathbf{fun}}\left(y_i, D(x_i)\right)$$

$$L_{0|1}\left(y_i, D(x_i)\right) = \begin{cases} 1, \text{if } y_i D(x_i) < 0 \\ 0, \quad \text{otherwise} \end{cases}$$

$$L_{\mathbf{per}}\left(y_i, D(x_i)\right) = \max(0, -y_i D(x_i))$$

$$L_{\mathbf{hin}}\left(y_i, D(x_i)\right) = \max(0, 1 - y_i D(x_i))$$

$$L_{\mathbf{log}}\left(y_i, D(x_i)\right) = \log(1 + e^{-y_i D(x_i)})$$

$$L_{\mathbf{exp}}\left(y_i, D(x_i)\right) = e^{-y_i D(x_i)}$$

$$L_{\mathbf{squ}}\left(y_i, D(x_i)\right) = (1 - y_i D(x_i))^2$$

Comparison of different loss functions

- In the zero-one loss, if a data sample is predicted correctly ($y_i \boldsymbol{D}(\boldsymbol{x_i}) > 0$), it results in zero penalties; otherwise, there is a penalty of one. For any data sample that is not predicted correctly, it receives the same loss.
- For the perceptron loss, the penalty for each wrong prediction is proportional to the extent of violation. For other losses, a data sample can still incur penalty even if it is classified correctly.
- The log loss is similar to the hinge loss but it is a smooth function which can be optimized with the gradient descent method.
- While log loss grows slowly for negative values, exponential loss and square loss are more aggressive.
- Note that, in all of these loss functions, square loss will penalize correct predictions severely when the value of $y_i \boldsymbol{D}(\boldsymbol{x_i})$ is large.
- In addition, zero-one loss is not convex while the other loss functions are convex. Note that the hinge loss and perceptron loss are not strictly convex.

Discriminative and Generative Classifiers

- Generative classifiers define a model for each class and then assign new objects to the model that suits them best.
- Discriminative classifiers explicitly define a decision boundary between classes.
- The Bayesian classier is an example of a generative classifier.
- The SVM and logistic regression are examples of discriminative classifiers.

Homework

觀看 Prof. Hung-yi Lee 講解SVM
https://www.youtube.com/watch?v=QSEPStBgwRQ