Support Vector Machine

生醫光電所 吳育德

Credit:
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 −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 $\mathbf{w}^T \boldsymbol{x} + b = 0$

- 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 ($i = 1, \ldots, N$) .

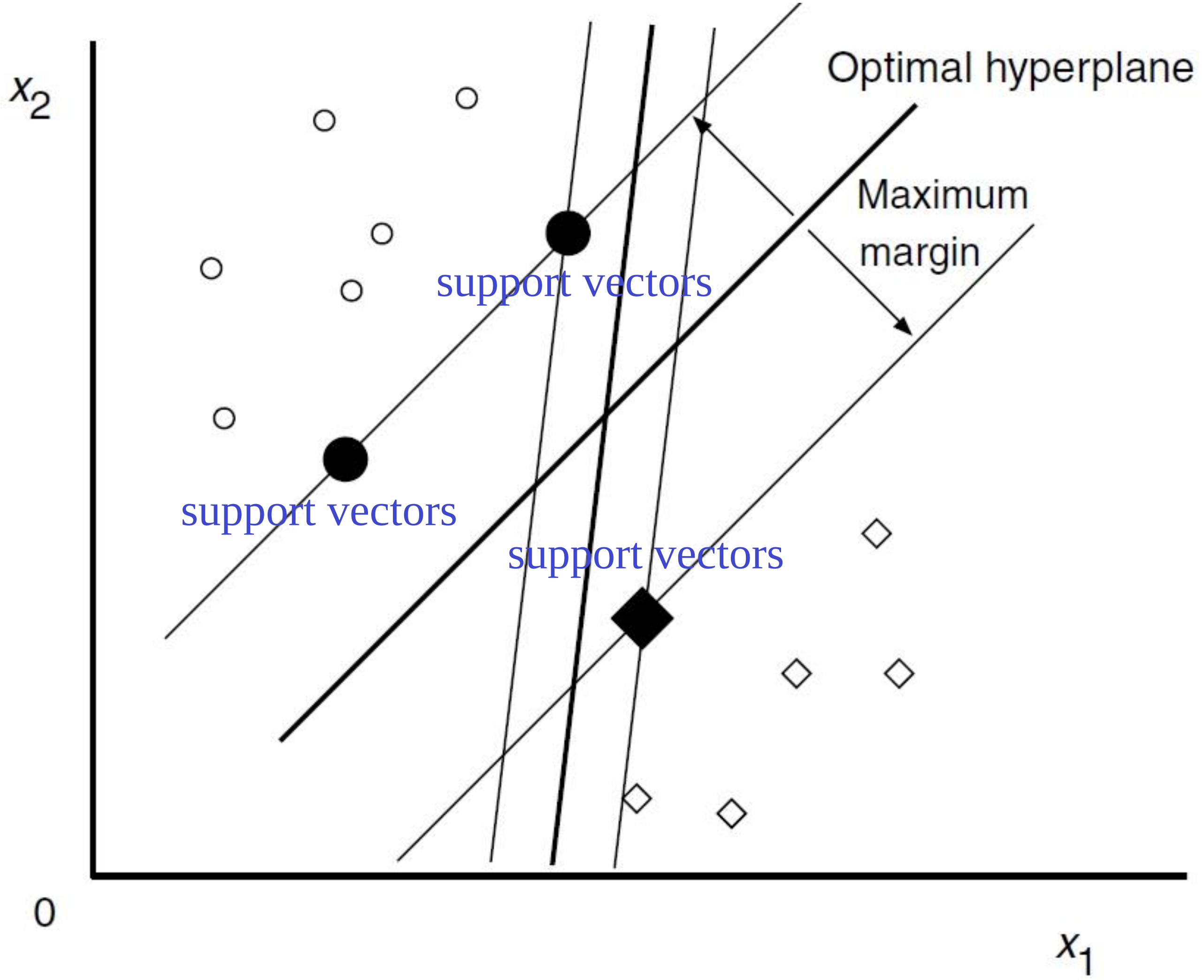
- When $c = 0$, the separating hyperplane is in the middle of the two hyperplanes with $c = 1$ and -1.

- 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

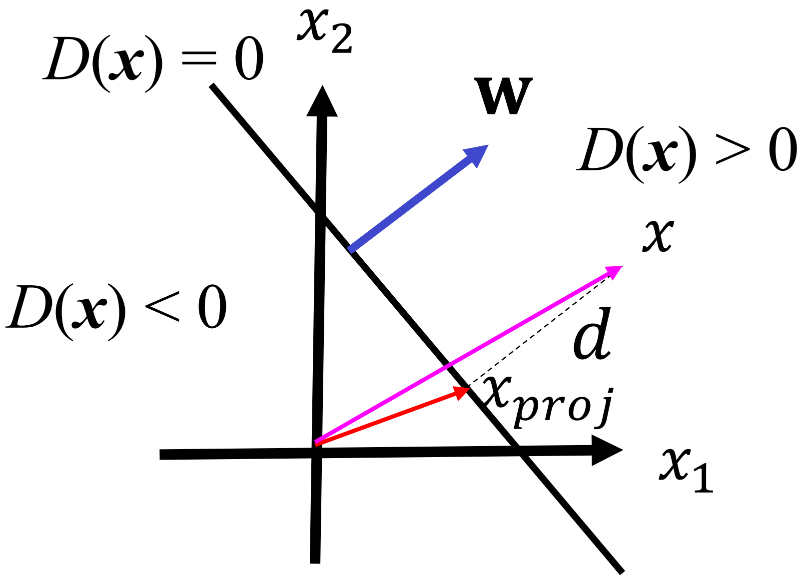
- x_{proj}: projection of x onto the hyperplane $D(\boldsymbol{x}) = 0$.
- $\boldsymbol{d}$: the normal distance between x and x_{proj}.
- $\boldsymbol{x} = \boldsymbol{x}_{proj} + \boldsymbol{d}\frac{\mathbf{w}}{||\mathbf{w}||}$

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

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

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

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



Cost function for obtaining the optimal separating hyperplane

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

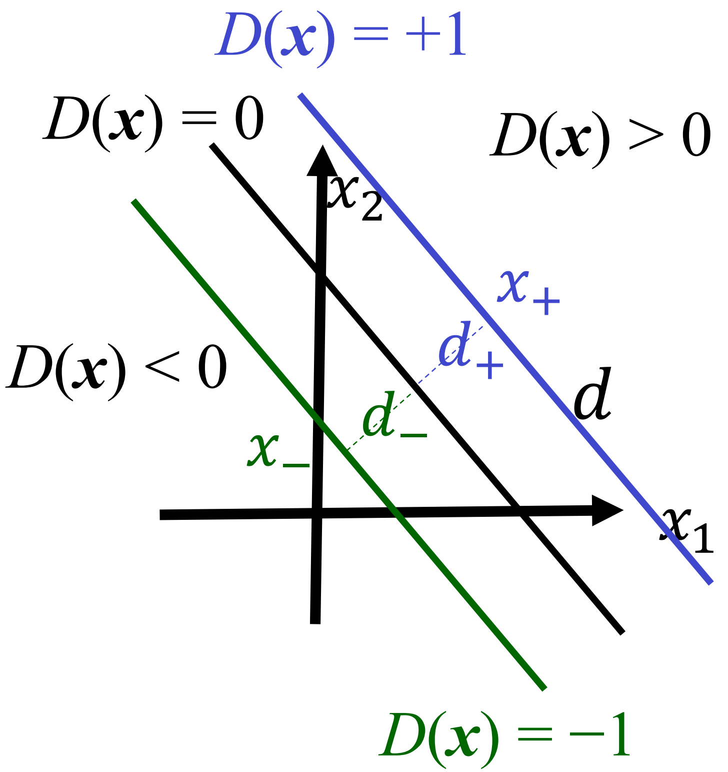
- Margin $= d_+ + d_- = \frac{2}{||\mathbf{w}||}$

- The optimal separating hyperplane can be obtained by minimizing

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

with respect to **w** and *b* subject to the constraints

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



Optimization with *m* inequality constraints

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

$$\text{Minimize } F(\boldsymbol{x}) \qquad ①$$

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

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

- The ith constraint $g_i(\boldsymbol{x}) \le 0$ 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:

$$\text{Minimize } F(\boldsymbol{x}) \qquad ①$$

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

Optimization with *m* inequality constraints

- ① ③ is an optimization problem with only m equality constraints
- Let $\boldsymbol{y} = [y_1, \cdots, y_m]^T$, $\boldsymbol{\lambda} = [\lambda_1, \cdots, \lambda_m]^T$, the Lagrangian has the form:

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

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

- The optimal conditions are

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

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

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

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

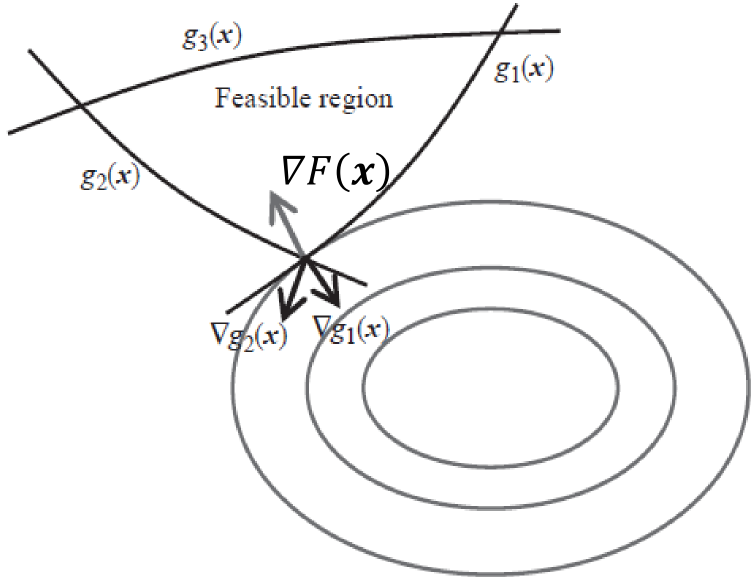
Optimization with *m* inequality constraints

- 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 $\lambda_i \neq 0$
- $\lambda_i y_i = 0$ ⑤ $\Rightarrow$ either $\lambda_i = 0 \Rightarrow y_i \neq 0$ and $g_i(\boldsymbol{x}) + y_i^2 = 0 \Rightarrow g_i(\boldsymbol{x}) < 0$ (inactive)

  or $\lambda_i \neq 0 \Rightarrow y_i = 0$ and $g_i(\boldsymbol{x}) + y_i^2 = 0 \Rightarrow g_i(\boldsymbol{x}) = 0$ (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 $\nabla g_1(\boldsymbol{x}), \nabla g_2(\boldsymbol{x})$



Optimization with m inequality constraints

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

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

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

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

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

- 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.

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

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

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 **w** and b and maximizing w.r.t α_i (≥0) satisfying the following KKT conditions

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

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

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

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

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

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

- Substituting (*) and (**) into ⑤, we obtain the dual problem.

Maximize

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

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

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

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

w.r.t. α_i subject to

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

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

$\Rightarrow$ α_i's are used to get optimal **w** and b

- This is a *convex optimization problem*. It is possible to obtain $\boldsymbol{\alpha}$ vector corresponding to the *global optimum*. $\mathbf{w} = \sum_{i=1}^{N} \alpha_i \, y_i \, \boldsymbol{x}_i$.

- 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*,

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

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

c. $y_i \times y_i(\mathbf{w}^T \boldsymbol{x}_i + b) = y_i \Rightarrow b = y_i - \mathbf{w}^T \boldsymbol{x}_i$ where i is such that $\alpha_i > 0$.

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

Making Prediction

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

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

$$D(\boldsymbol{x}) = \mathbf{w}^T \boldsymbol{x} + b = \sum_{x_i \in S} \alpha_i y_i \boldsymbol{x}_i{}^T \, \boldsymbol{x} + (y_i - \sum_{x_i \in S} \alpha_i y_i \boldsymbol{x}_i{}^T \, \boldsymbol{x}_i)$$

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

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

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

Example

- Consider a linearly separable case shown in Fig. 2.2, (x_1, y_1) = (-1,1), (x_2, y_2) = (0,-1), (x_3, y_3) = (1,-1), The inequality constraints given by $y_i(\mathbf{w}^T \boldsymbol{x_i} + b) \geq 1, i = 1, \ldots, 3$ 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

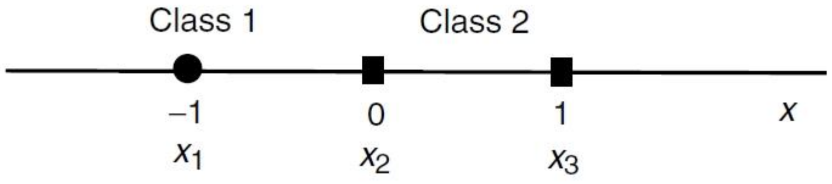


Fig. 2.2. Linearly separable one-dimensional case

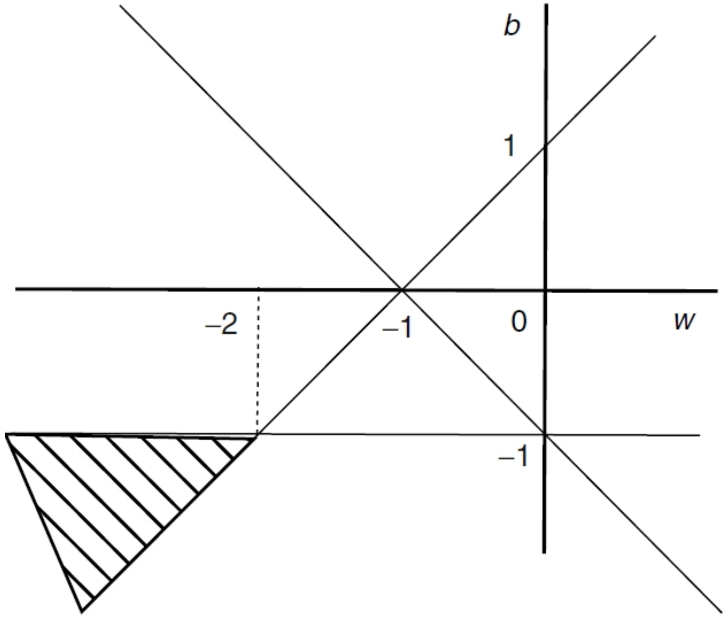


Fig. 2.3. Region that satisfies constraints

- The dual problem is to maximize

$$Q(\boldsymbol{\alpha})=\sum_{i=1}^{3}\alpha_i-\frac{\mathbf{1}}{\mathbf{2}}\sum_{i=1}^{3}\sum_{j=1}^{3}\alpha_i\alpha_j\,y_iy_j\boldsymbol{x}_i^T\boldsymbol{x}_j,$$ **(x_1, y_1) = (-1,1), (x_2, y_2) = (0,-1), (x_3, y_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

$$\sum_{i=1}^{3}\alpha_i\,y_i=\alpha_1-\alpha_2-\alpha_3=0,\ \alpha_i\geq 0, i = 1, . . . , 3$$

- 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 $\alpha_i\geq 0, i = 1, . . . , 3$

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

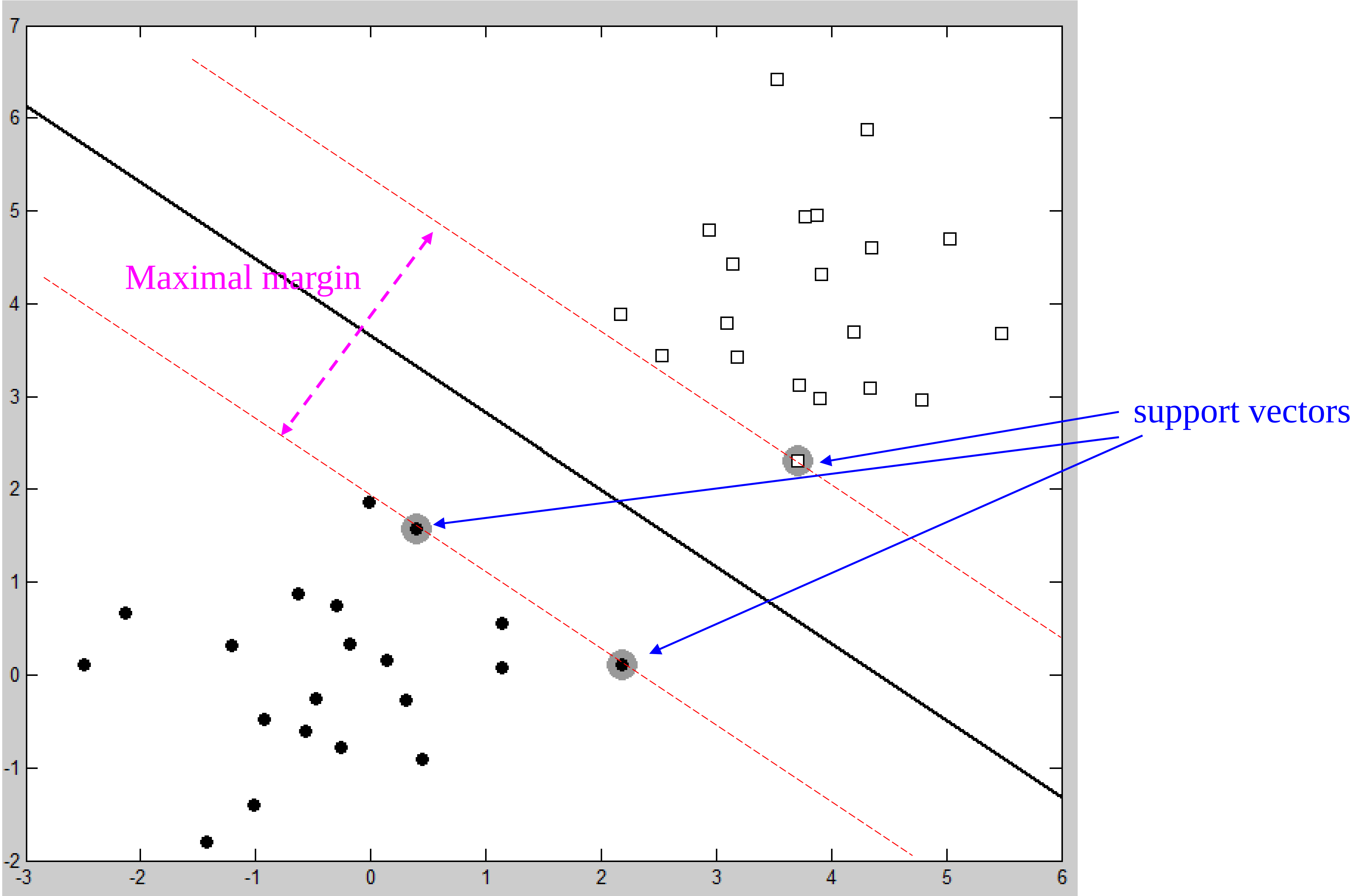
- Now $Q(\boldsymbol{\alpha}) = 2\alpha_1 - \frac{\mathbf{1}}{\mathbf{2}}\alpha_1{}^2 = -\frac{\mathbf{1}}{\mathbf{2}}(\alpha_1 - 2)^2 + 2,\ \alpha_1 \geq 0$

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 $w = \sum_{i=1}^{3} \alpha_i\, y_i x_i = 2(1)(-1) + 2(-1)0 + 0(-1)(1) = -2$ and $b = y_i - w^T x_i = y_1 - (-2)x_1 = 1 - (-2)(-1) = -1$ $(\alpha_1 = 2 > 0, x_1$ is a support vector $\Rightarrow$ $y_1(w^T x_1 + b) = 1)$, 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., $y_3 = 1$. 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 (svmhard.m)



```matlab
%% svmhard.m
% From A First Course in Machine Learning, Chapter 5.
% Simon Rogers, 01/11/11 [simon.rogers@glasgow.ac.uk]
% 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

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

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

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

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

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

```matlab
%% 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

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

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

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

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

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

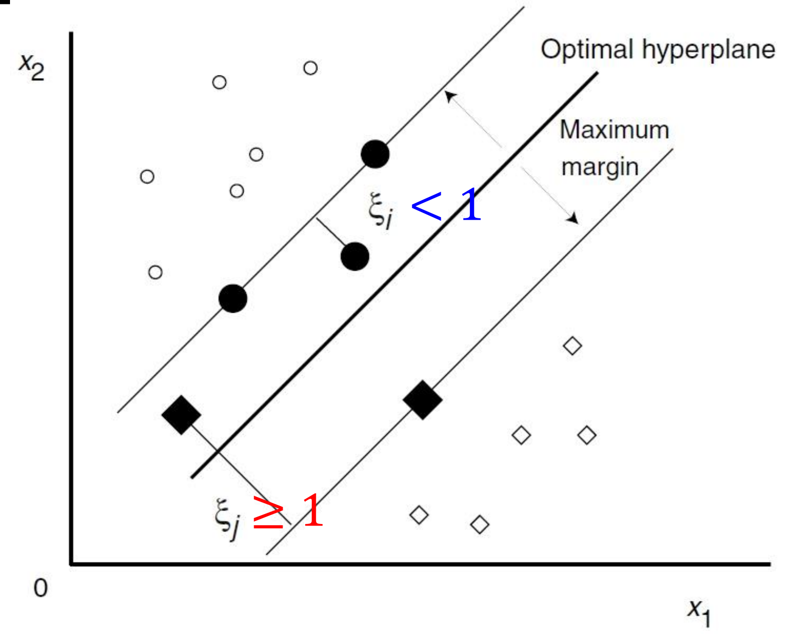


Fig. 2.4. Inseparable case in a two-dimensional space

- Minimize $Q(\mathbf{w}) = \frac{1}{2}||\mathbf{w}||^2 + \sum_{i=1}^{N} \theta(\xi_i), \theta(\xi_i) = \begin{cases} 1, \text{for } \xi_i > 0 \\ 0, \text{for } \xi_i = 0 \end{cases}$
  subject to $y_i(\mathbf{w}^T \boldsymbol{x_i} + b) \geq 1 - \xi_i, i = 1, \ldots, N$

- This is a combinatorial optimization and difficult to solve

- Instead, we minimize $Q(\mathbf{w}, \mathrm{b}, \boldsymbol{\xi}) = \frac{1}{2}||\mathbf{w}||^2 + \mathrm{C}\sum_{i=1}^{N} \xi_i^{\,p},\ \xi_i \geq 0$
  subject to $y_i(\mathbf{w}^T \boldsymbol{x_i} + b) \geq 1 - \xi_i, i = 1, \ldots, N$

where $\boldsymbol{\xi} = (\xi_1, \ldots, \xi_N)^{\mathbf{T}}$, C determines the trade-off between the maximization of margin and minimization of classification error, and $p = 1$ (l_1 soft-margin SVM), or 2(l_2 soft-margin SVM)

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

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

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

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

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

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

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

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

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

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

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

Maximize

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

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

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

with respect to α_i subject to the constraints

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

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

- Especially, ② and ③ are called KKT (complementarity) conditions

- From $\alpha_i + \beta_i$=C, $\beta_i \xi_i = 0$ and ② there are three cases for α_i:

1. $\alpha_i = 0$. Then β_i=C, $\xi_i = 0$. Thus $\boldsymbol{x_i}$ is correctly classified

2. $0 < \alpha_i < C$. Then ② $\Rightarrow y_i(\mathbf{w}^T \boldsymbol{x_i} + b) - 1 + \xi_i = 0$, and $\beta_i \neq 0, \Rightarrow \xi_i = 0$. Therefore, $y_i(\mathbf{w}^T \boldsymbol{x_i} + b) = 1$ and $\boldsymbol{x_i}$ is a support vector. We call the support vector with $C > \alpha_i > 0$ a good (unbounded) SV.

3. $\alpha_i = C$. Then ② $\Rightarrow y_i(\mathbf{w}^T \boldsymbol{x_i} + b) - 1 + \xi_i = 0$ and $\xi_i \geq 0$. Thus $\boldsymbol{x_i}$ is a support vector. We call the support vector with $\alpha_i = C$ a bad (bounded) SV.

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

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

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

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

$$D(\boldsymbol{x}) = \mathbf{w}^T \boldsymbol{x} + b = \sum_{x_i \in S} \alpha_i y_i \boldsymbol{x}_i{}^T \, \boldsymbol{x} + b$$

- For the unbounded α_i, $b = y_i - \mathbf{w}^T \boldsymbol{x}_i$ is satisfied.
- To ensure the precision of calculations, we take the average of b that is calculated for unbounded support vectors, $b = \frac{1}{\#(\boldsymbol{x}_i \in G)} \sum_{x_i \in G} (y_i - \mathbf{w}^T \boldsymbol{x}_i)$

  where G is the set of good support vector
- Then unknown datum $\boldsymbol{x}$ is classified into:

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

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

```
% From A First Course in Machine Learning, Chapter 5.
% Simon Rogers, 01/11/11 [simon.rogers@glasgow.ac.uk]
% 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

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

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

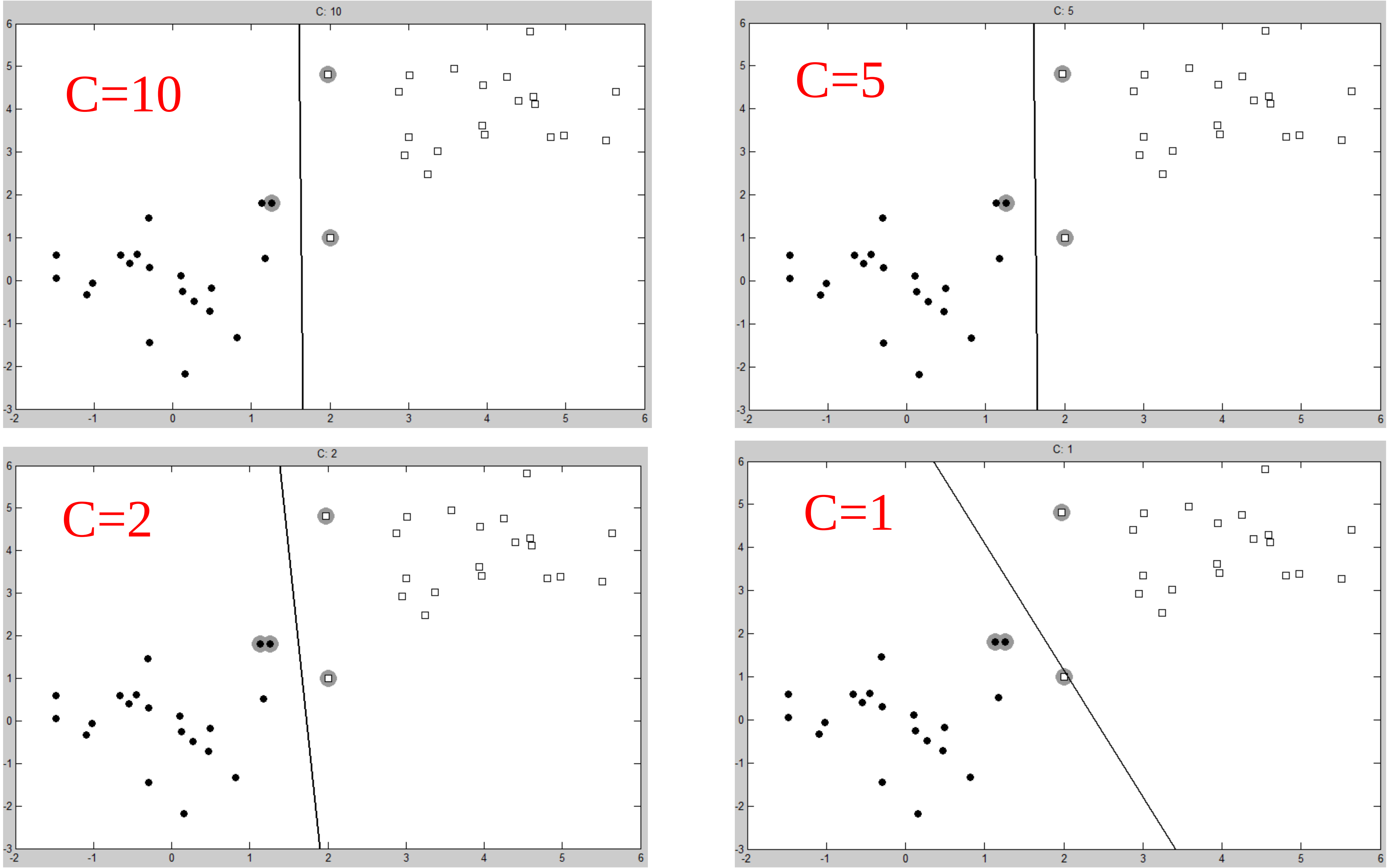
$$\mathbf{w} = \sum_{i=1}^{N} \alpha_i y_i \boldsymbol{x}_i$$

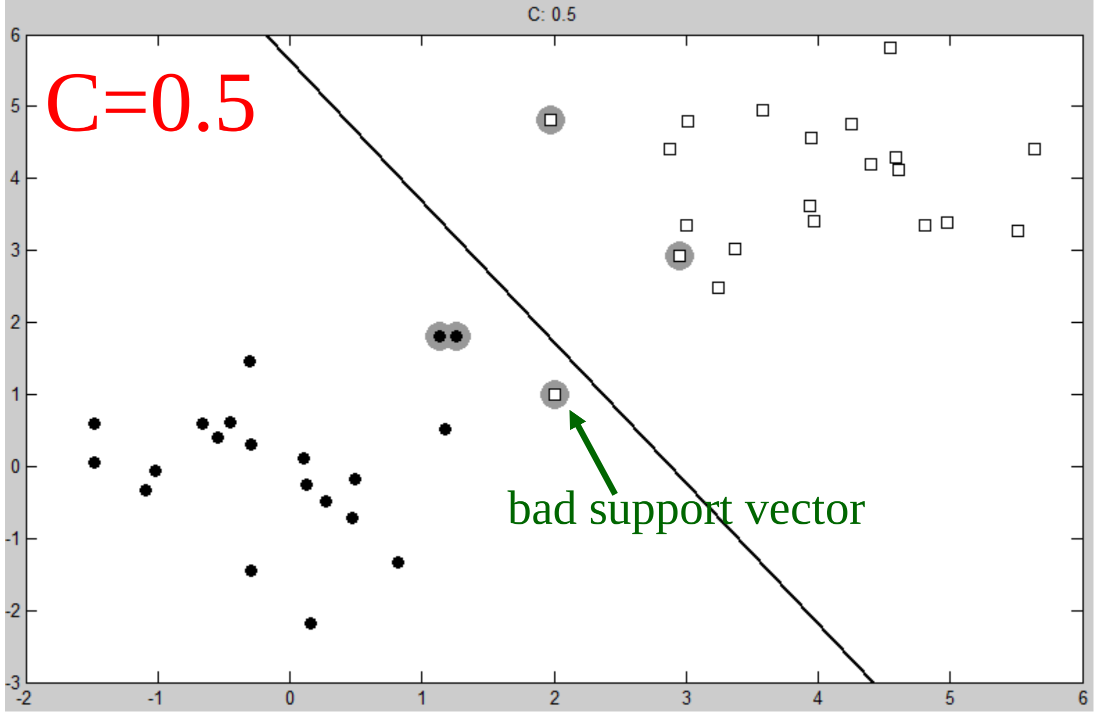
$$b = \frac{1}{\#(\boldsymbol{x}_i \in G)} \sum_{\boldsymbol{x}_i \in G} (y_i - \mathbf{w}^T \boldsymbol{x}_i)$$

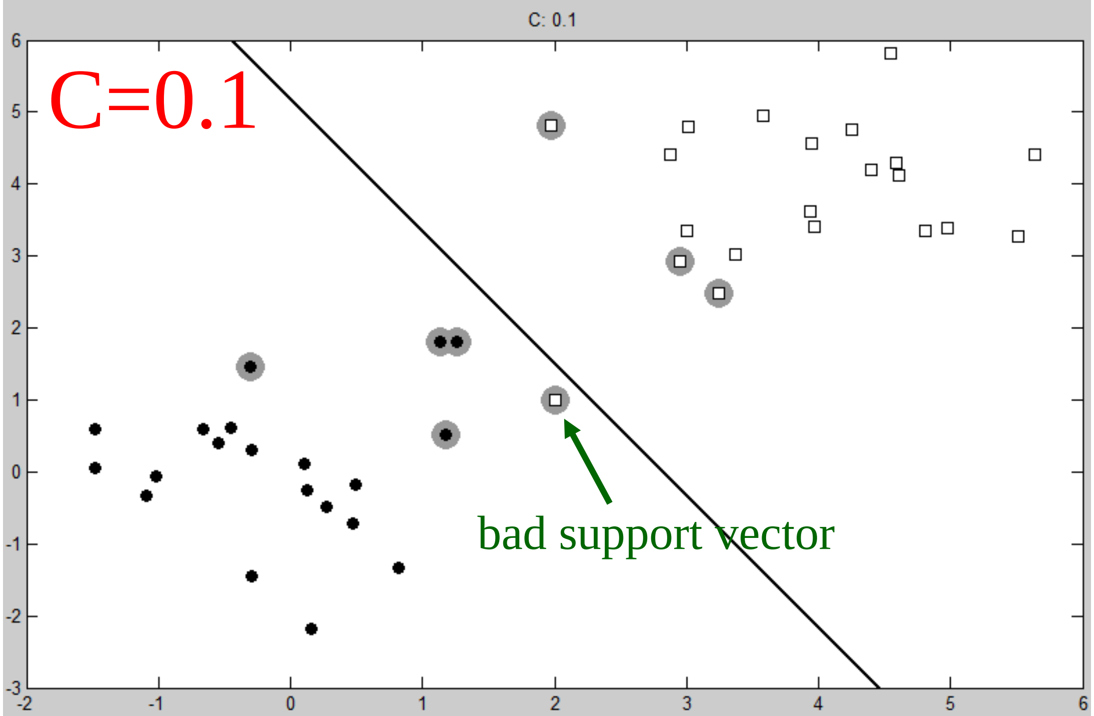
$$G = \{x_i | C > \alpha_i > 0\}$$

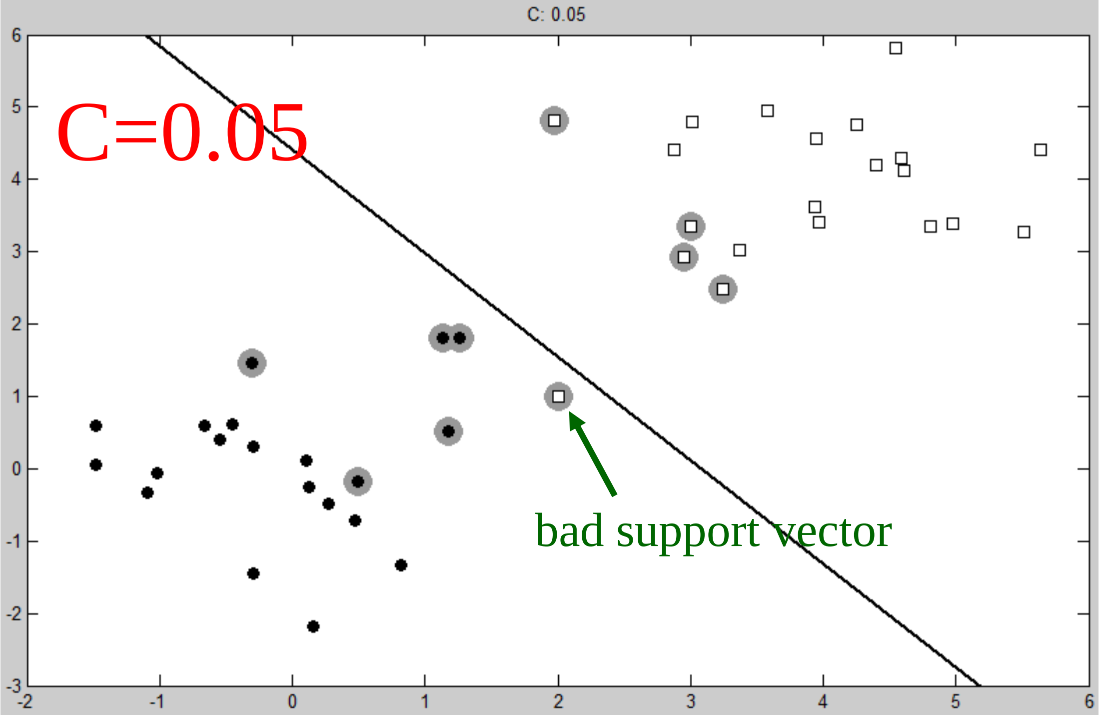
```
%% Plot the data, decision boundary and Support vectors
    figure(1);hold off
    pos = find(alpha>1e-6);
    plot(x(pos,1),x(pos,2),'ko','markersize',15,'markerfacecolor',[0.6 0.6 0.6],...
        'markeredgecolor',[0.6 0.6 0.6]);
    hold on
    for i = 1:length(tv)
        pos = find(t==tv(i));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

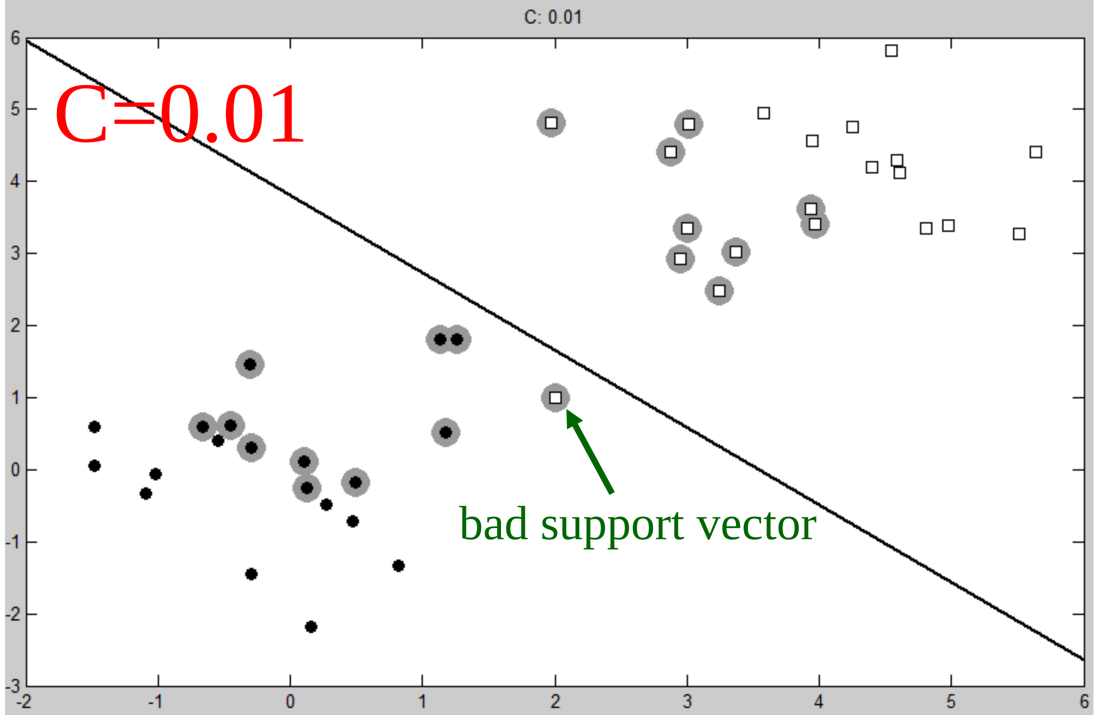
$$\mathbf{w} = \sum_{i=1}^{N} \alpha_i \, y_i \, \boldsymbol{x}_i$$


C: 10
C=10
C: 5
C=5
C: 2
C=2
C: 1
C=1


C: 0.5
C=0.5
bad support vector

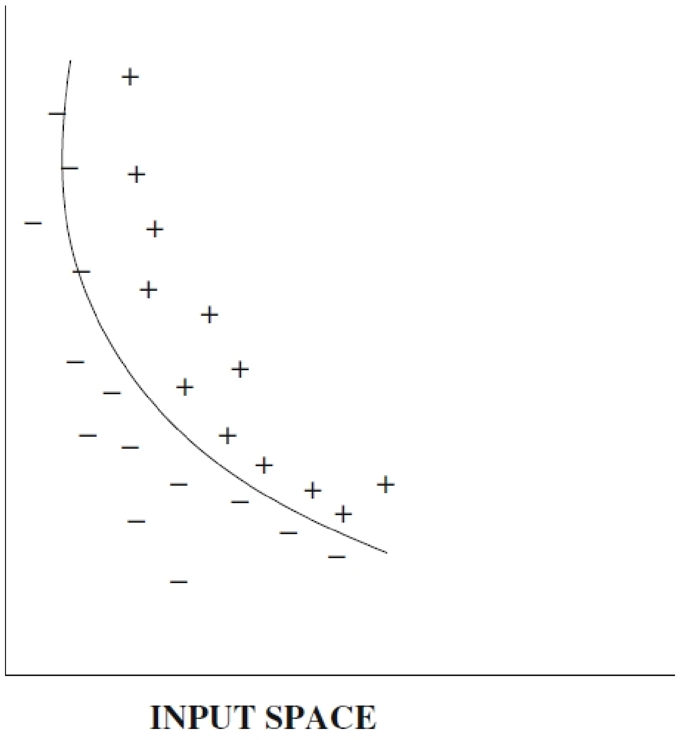

C: 0.1
C=0.1
bad support vector

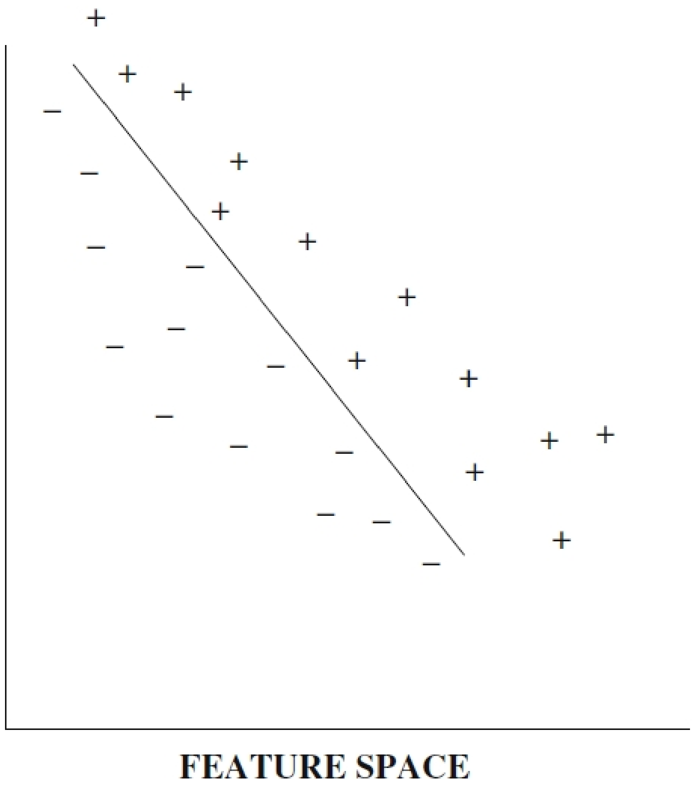

C: 0.05
C=0.05
bad support vector


C: 0.01
C=0.01
bad 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

- Using a nonlinear $\boldsymbol{g}(\boldsymbol{x}) = (g_1(\boldsymbol{x}), \ldots, g_l(\boldsymbol{x}))^{\mathbf{T}}$, 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 $H(\boldsymbol{x}, \boldsymbol{x}')$ satisfies

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

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

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

- If ② is satisfied,

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

- 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 $Q(\boldsymbol{\alpha}) = \sum_{i=1}^{N} \alpha_i - \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N} \alpha_i \alpha_j y_i y_j H(\boldsymbol{x_i}, \boldsymbol{x_j})$

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

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

- Decision function is

$$D(\boldsymbol{x}) = \mathbf{w}^T \boldsymbol{g}(\boldsymbol{x}) + b = \sum_{x_i \in S} \alpha_i y_i H(\boldsymbol{x_i}, \boldsymbol{x}) + b$$

$b = y_j - \sum_{x_i \in S} \alpha_i y_i H(\boldsymbol{x_i}, \boldsymbol{x_j})$, $\boldsymbol{x_j}$ is an unbounded support vector

- To ensure stability of calculations, we take the average:

$$b = \frac{1}{\#(\boldsymbol{x}_j \in G)} \sum_{x_j \in G} \left(y_j - \sum_{x_i \in S} \alpha_i y_i H(\boldsymbol{x_i}, \boldsymbol{x_j})\right)$$

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

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

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

Kernels used in SVM

- **Linear Kernels:**

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

- **Polynomial Kernels:**

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

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

When $m = 2, d = 2$,

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

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

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

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

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

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

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 $\Rightarrow$ it is a kernel.

$\exp(-\gamma||\boldsymbol{x}||^2)$ 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

$$D(\boldsymbol{x}) = \sum_{x_i \in S} \alpha_i y_i H(\boldsymbol{x_i}, \boldsymbol{x}) + b = \sum_{x_i \in S} \alpha_i y_i \exp(-\gamma||\boldsymbol{x_i} - \boldsymbol{x}||^2) + b$$

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

```
%% svmgauss.m
% From A First Course in Machine Learning, Chapter 5.
% Simon Rogers, 01/11/11 [simon.rogers@glasgow.ac.uk]
% 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);
```

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

```
% 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';
```

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

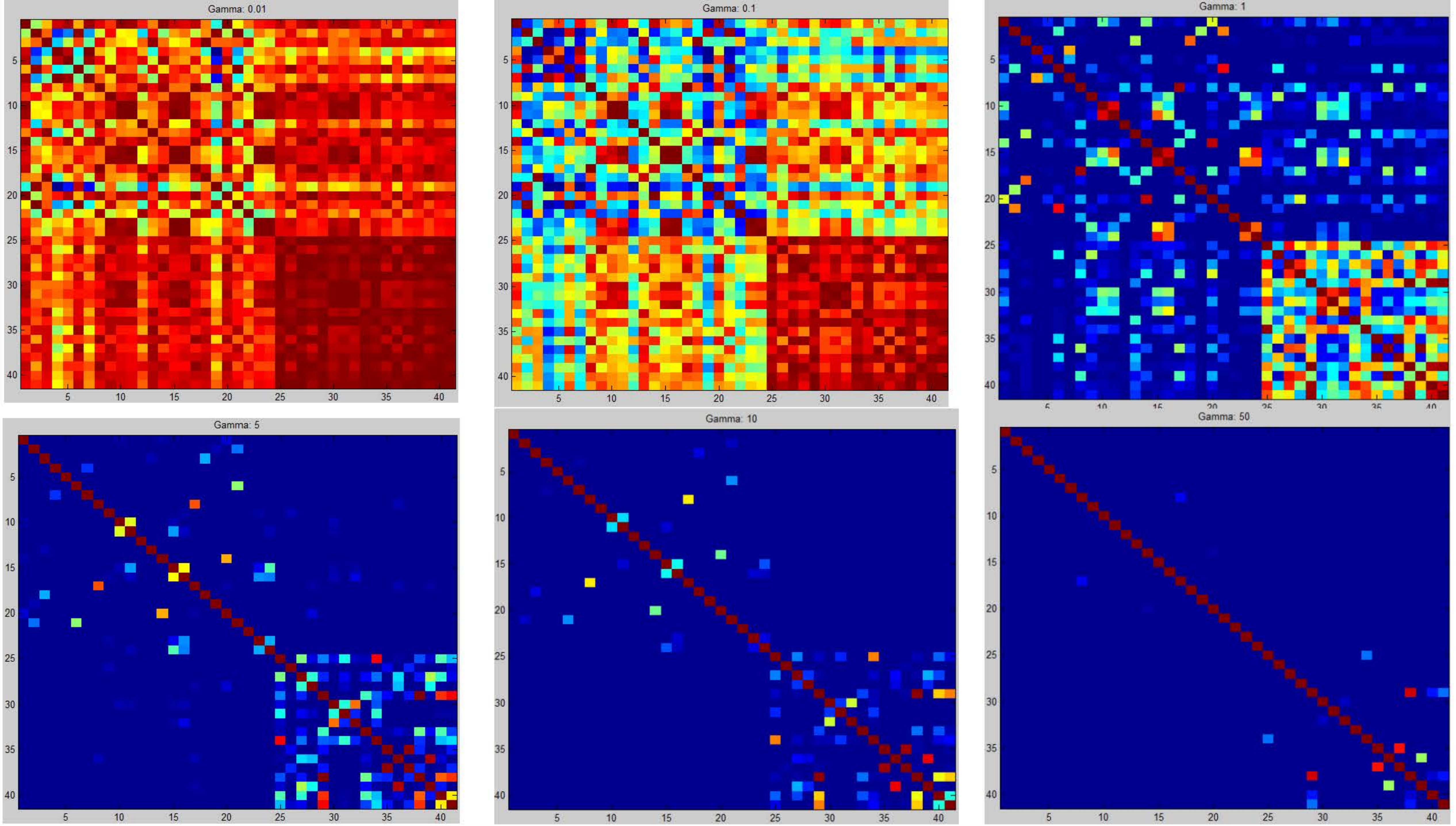
$$\sum_{i=1}^{N} \alpha_i y_i = 0, \quad C \geq \alpha_i \geq 0, \qquad i = 1, \ldots, N$$

α_i's > 0 are support vectors

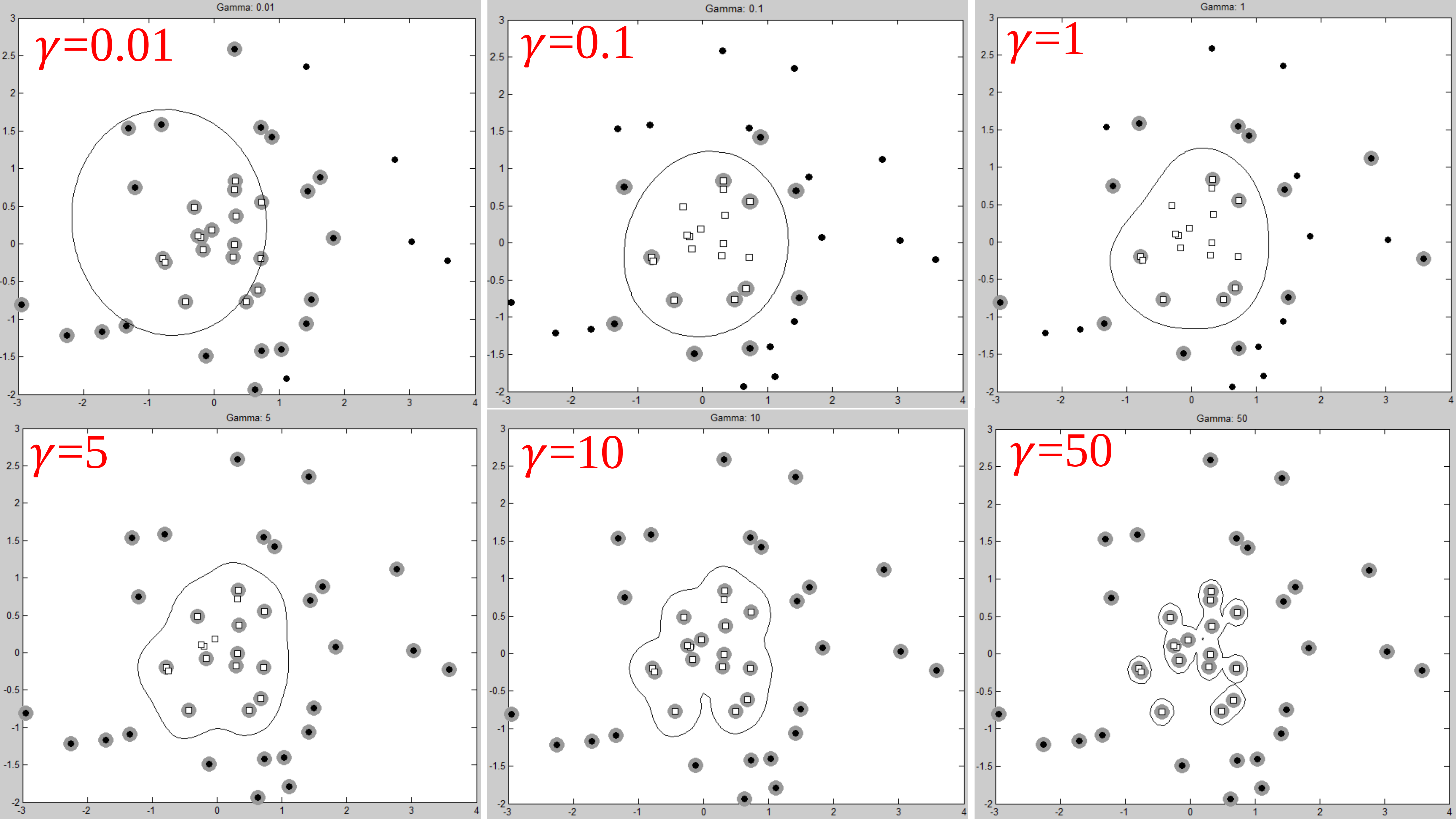
$$\sum_{x_i \in S} \alpha_i y_i \exp(-\gamma ||\boldsymbol{x}_i - \boldsymbol{x}||^2) + b$$

```matlab
% 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
5
10
15
20
25
30
35
40



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 $H(\boldsymbol{x}, \boldsymbol{x}') = \boldsymbol{g}(\boldsymbol{x})^{T} \boldsymbol{g}(\boldsymbol{x}')$

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

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

- b is obtained by

$b = y_j - \sum_{x_i \in S} \alpha_i y_i \, \boldsymbol{g}(\boldsymbol{x}_i)^{T} \boldsymbol{g}(\boldsymbol{x}_j)$, $\boldsymbol{x}_j$ is a good support vector

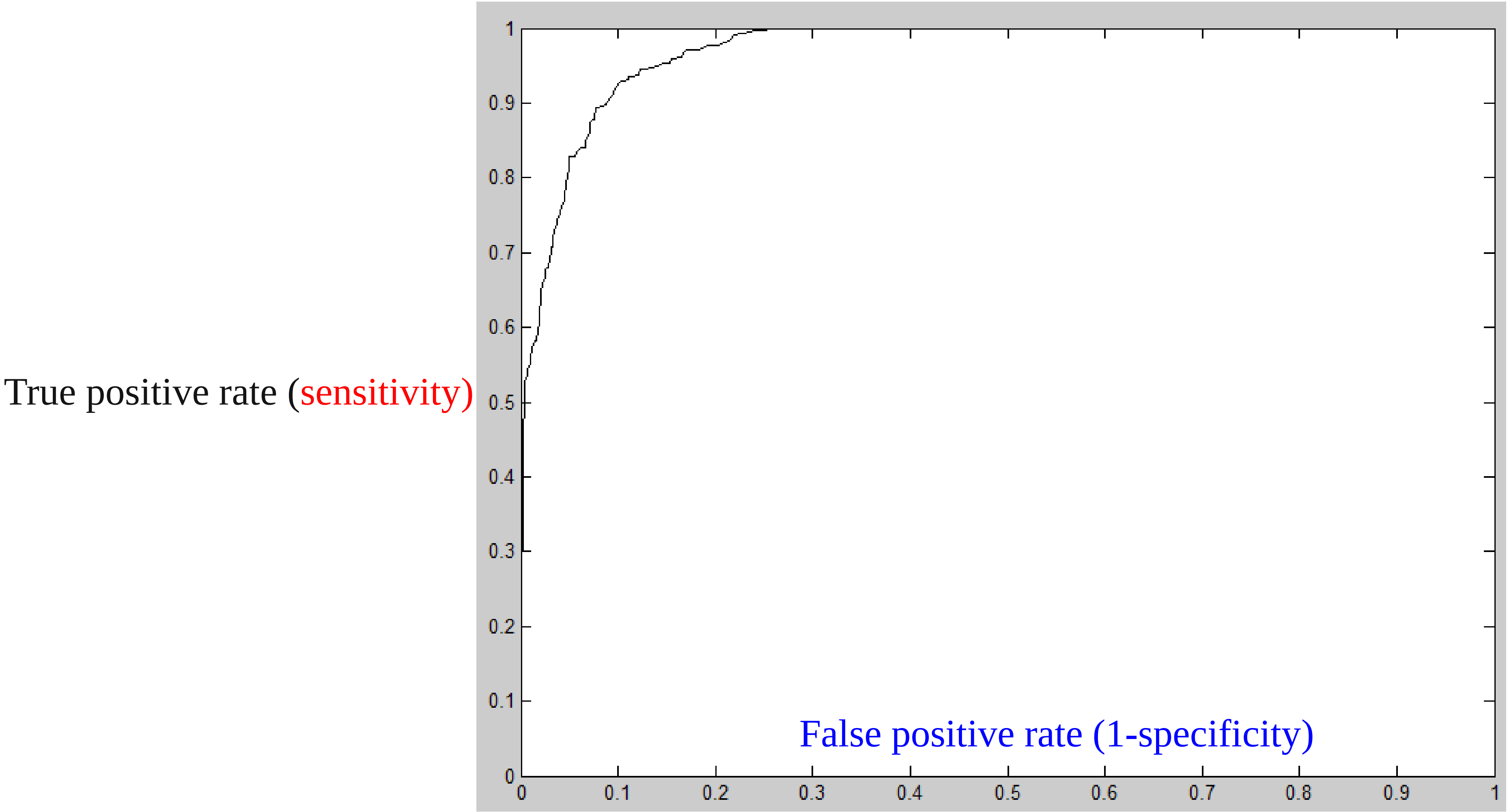
		True Status			
		Yes	No		
Predicted status	Yes	True Positive (TP)	False Positive (FP) Type I error	Positive Predictive Rate, Precision TP/(TP+FP)	False Discovery Rate FP/(TP+FP)
	No	False Negative (FN) Type II error	True Negative (TN)	False Omission Rate FN/(FN+TN)	Negative Predictive Rate TN/(FN+TN)
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
%% svmroc.m
% From A First Course in Machine Learning, Chapter 5.
% Simon Rogers, 01/11/11 [simon.rogers@glasgow.ac.uk]
% ROC analysis of SVM
clear all;close all;
%% Load the data
load t.csv
load X.csv
load testt.csv
load testX.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
```

```matlab
%% 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';
```

```matlab
%% 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)



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.