Chapter 14
SVM and SVM-based Object Detection

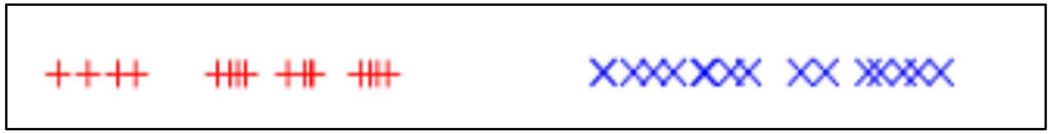
Lin ZHANG
School of Computer Science and Technology
Tongji University

Outline

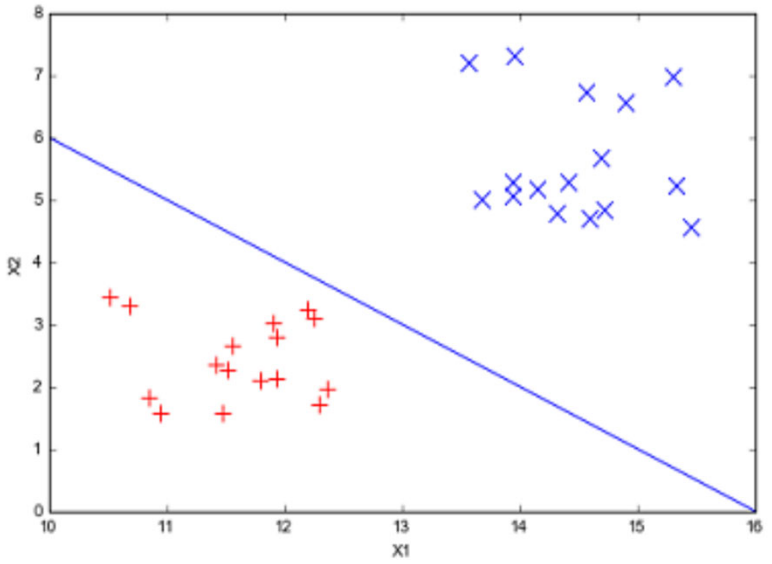
- Linear separability and hyperplanes
- The perceptron learning algorithm
- Hard-margin SVM
- Soft-margin SVM
- Kernelized SVM
- Multi-class classification with SVM
- Object detection using SVM+HOG
- Practice

Linear separability

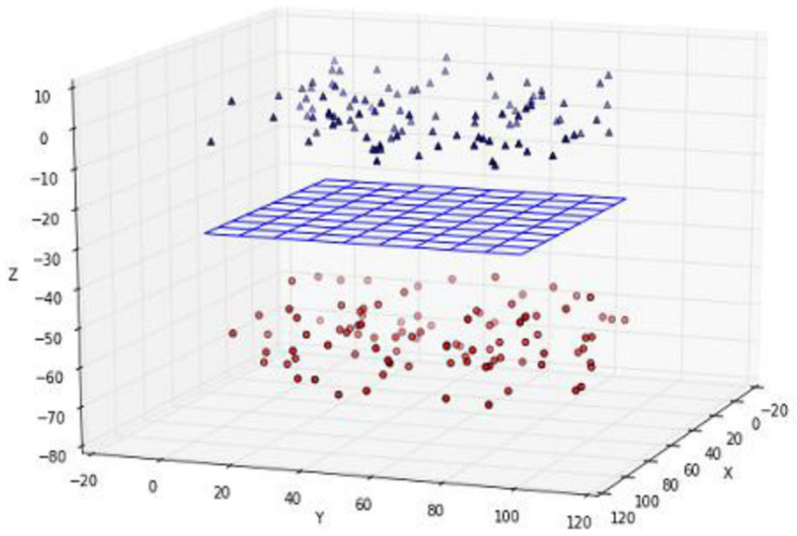
In one dimension, you can find a **point** separating the data



In two dimensions, you can find a **line** separating the data

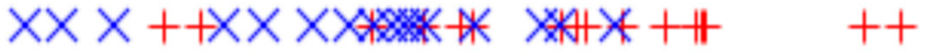


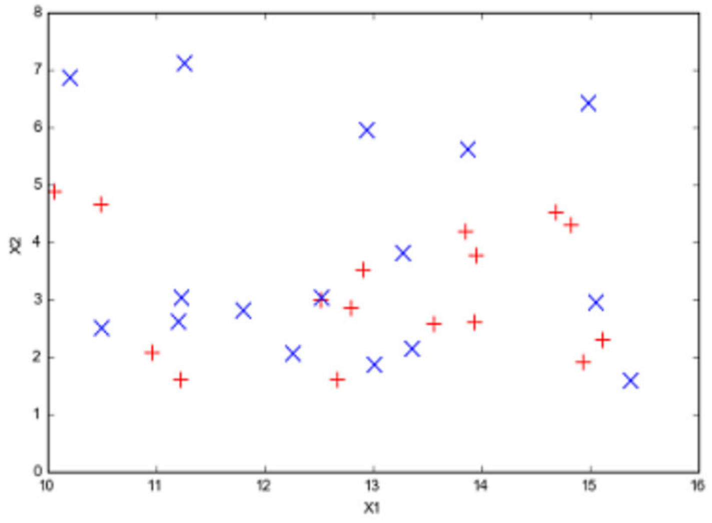
In three dimensions, you can find a **plane** separating the data



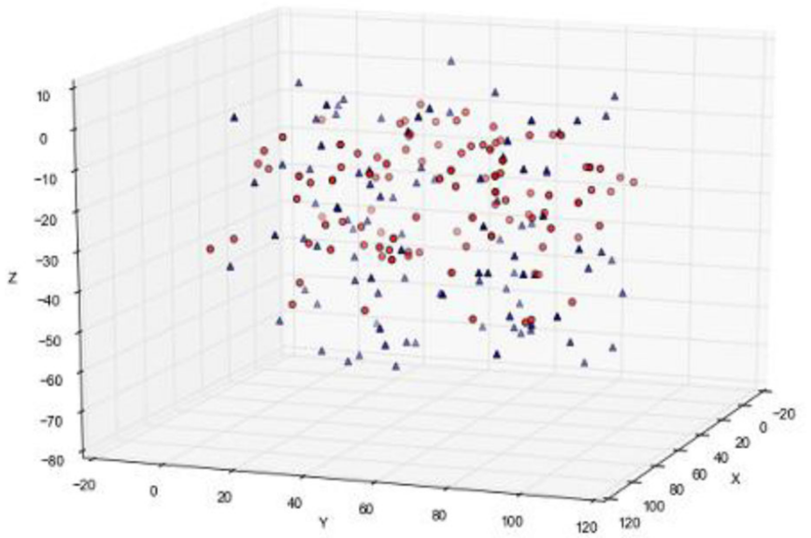
Linear separability

When data is non-linearly separable, we cannot find a separating point, line, or plane.


Non-linearly separable data in 1D



Non-linearly separable data in 2D



Non-linearly separable data in 3D

Linear separability

If data is linearly separable,

for 1D case, we use a **point (0D)** to separate the data
for 2D case, we use a **line (1D)** to separate the data
for 3D case, we use a **plane (2D)** to separate the data

What do we use to separate the data when there are more than three dimensions?

⬇

Hyperplane!

Hyperplane

Definition 1: Hyperplane

In geometry, a hyperplane is a **subspace** of one dimension less than its **ambient** space

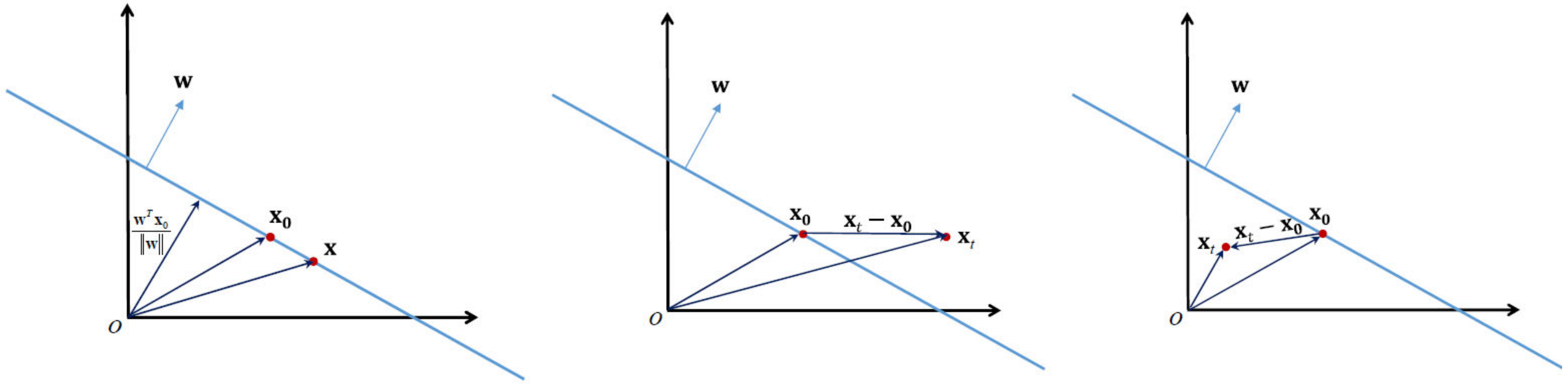
In *d*-D Euclidean space, a hyperplane is the set of points satisfying,

$$\mathbf{w}\cdot\mathbf{x}+b=0$$

where $\mathbf{x}\in\mathbb{R}^d$ is the point locating on that hyperplane, $\mathbf{w}\in\mathbb{R}^d$ is the hyperplane's normal vector, and b is a constant

If not so straightforward, make an analogy with the cases of the line in 2D space and the plane in the 3D Euclidean space

Hyperplane



（a） （b） （c）

（a）超平面方程的几何解释：$\mathbf{w}^T\mathbf{x}+b=0$ 定义了一个超平面，其法向量为 $\mathbf{w}$，若该平面过一已知点 $\mathbf{x}_0$，则 $b=-\mathbf{w}^T\mathbf{x}_0$；以 $\mathbf{w}$ 为正向, 从原点出发到超平面的有向距离为 $\frac{-b}{\|\mathbf{w}\|}$；（b）位于超平面正侧（$\mathbf{w}$ 指向的一侧）的点 $\mathbf{x}_t$ 满足 $\mathbf{w}^T\mathbf{x}_t+b>0$；（c）位于超平面负侧的点 $\mathbf{x}_t$ 满足 $\mathbf{w}^T\mathbf{x}_t+b<0$。

Classifying data with a hyperplane

Given a set of data points $\{\mathbf{x}_i\}$ shown as the right figure and a hyperplane defined by $\mathbf{w}=(0.4, 1.0)^T$ and $b=-9$

Suppose the classification model is $h_{\mathbf{w},b}$:

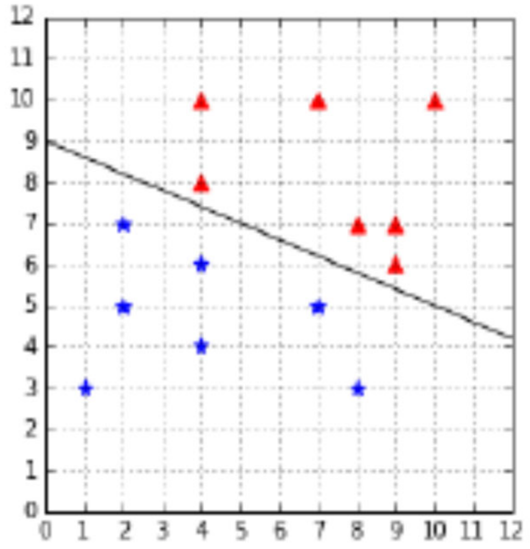
$$h_{\mathbf{w},b}(\mathbf{x}) = \begin{cases} +1, \textit{ if } \mathbf{w}\cdot\mathbf{x}+b \geq 0 \\ -1, \textit{ if } \mathbf{w}\cdot\mathbf{x}+b < 0 \end{cases}$$

(a linear classifier)

which is equivalent to :

$$h_{\mathbf{w},b}(\mathbf{x}) = \operatorname{sign}(\mathbf{w}\cdot\mathbf{x}+b)$$

We can associate each $\mathbf{x}_i$ with a label +1 or -1



Eg., $\mathbf{x}=(8,7)$

$\mathbf{w}\cdot\mathbf{x}+b = 0.4*8+1.0*7-9=1.2>0$, which is positive, so $h_{\mathbf{w},b}(\mathbf{x})=+1$

It means $\mathbf{x}$ is above the hyperplane.

It uses the position of $\mathbf{x}$ with respect to the hyperplane to predict $\mathbf{x}$'s label

Classifying data with a hyperplane

The main question is, given a set of linearly separable data, how to find such a hyperplane that separates the data points?

Next, we will at first introduce a simple algorithm to perform this task, i.e., perceptron

Outline

- Linear separability and hyperplanes
- The perceptron learning algorithm
- Hard-margin SVM
- Soft-margin SVM
- Kernelized SVM
- Multi-class classification with SVM
- Object detection using SVM+HOG
- Practice

The perceptron learning algorithm

- The goal of the perceptron algorithm is to find a hyperplane that can separate a linearly separable data set; once the hyperplane is found, it is used to perform binary classification; thus, it is a linear binary classifier (by extension, such an architecture can also perform multiclass classification)
- It was invented by Frank Rosenblatt in 1957[1]

（a）

（b）

弗兰克·罗森布拉特（Frank Rosenblatt，1928年7月11日-1971年7月11日）是美国心理学家，在人工智能领域享有盛誉。1971年，43岁生日那天，他在切萨皮克湾（Chesapeake Bay）驾驶一艘名为Clearwater的单桅帆船时溺水身亡。由于他最早提出了感知器学习算法，而感知器模型现在被认为是神经网络的基础构件，因此也有文献认为他是“深度学习之父”；（b）为了纪念Frank Rosenblatt，IEEE（电气电子工程师学会）从2004年开始设立了IEEE Frank Rosenblatt Award奖，用以表彰对生物和语言驱动的计算范式和系统做出杰出贡献的学者。

[1] Rosenblatt, Frank (1957). "The Perceptron—a perceiving and recognizing automaton". Report 85-460-1. Cornell Aeronautical Laboratory.

The perceptron learning algorithm

The problem the perceptron algorithm wants to solve is,

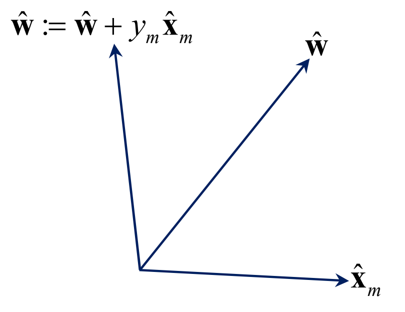
Given a linearly separable training dataset $\mathcal{D} = \left\{ (\mathbf{x}_i, y_i) :|\ \mathbf{x}_i \in \mathbb{R}^d, y_i \in \{+1, -1\} \right\}_{i=1}^{n}$, determine the hyperplane $(\mathbf{w}, b)$, $\forall (\mathbf{x}_i, y_i) \in \mathcal{D}$, $h_{\mathbf{w},b}(\mathbf{x}_i) = \text{sign}(\mathbf{w} \cdot \mathbf{x}_i + b) = y_i$

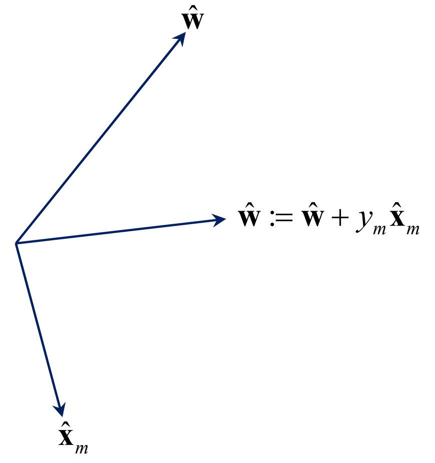
Denote $\hat{\mathbf{x}}_i = \begin{pmatrix} \mathbf{x}_i \\ 1 \end{pmatrix}$ and $\hat{\mathbf{w}} = \begin{pmatrix} \mathbf{w} \\ b \end{pmatrix}$, then $\mathbf{w}^T \mathbf{x}_i + b = \hat{\mathbf{w}}^T \hat{\mathbf{x}}_i$

The hyperplane can then be represented by $\hat{\mathbf{w}}$

The perceptron learning algorithm

```
感知器算法
输入：
    训练集 $\mathcal{D}=\{(\mathbf{x}_i,y_i):|\mathbf{x}_i\in\mathbb{R}^d, y_i\in\{+1,-1\}\}_{i=1}^n$
输出：
    超平面参数 $\hat{\mathbf{w}}$
随机初始化 $\hat{\mathbf{w}}$
misclassified_examples $:=\{(\mathbf{x}_i,y_i)\in\mathcal{D}:|h_{\hat{\mathbf{w}}}(\mathbf{x}_i)\neq y_i\}$
while misclassified_examples 非空
    从 misclassified_examples 中随机选取一个样本$(\mathbf{x}_m, y_m)$
    //注意：$y_m$ 是这个错分样本的真实类标
    $\hat{\mathbf{w}}:=\hat{\mathbf{w}}+\hat{\mathbf{x}}_m y_m$
    misclassified_examples $:=\{(\mathbf{x}_i,y_i)\in\mathcal{D}:|h_{\hat{\mathbf{w}}}(\mathbf{x}_i)\neq y_i\}$
end
返回最终得到的 $\hat{\mathbf{w}}$
```



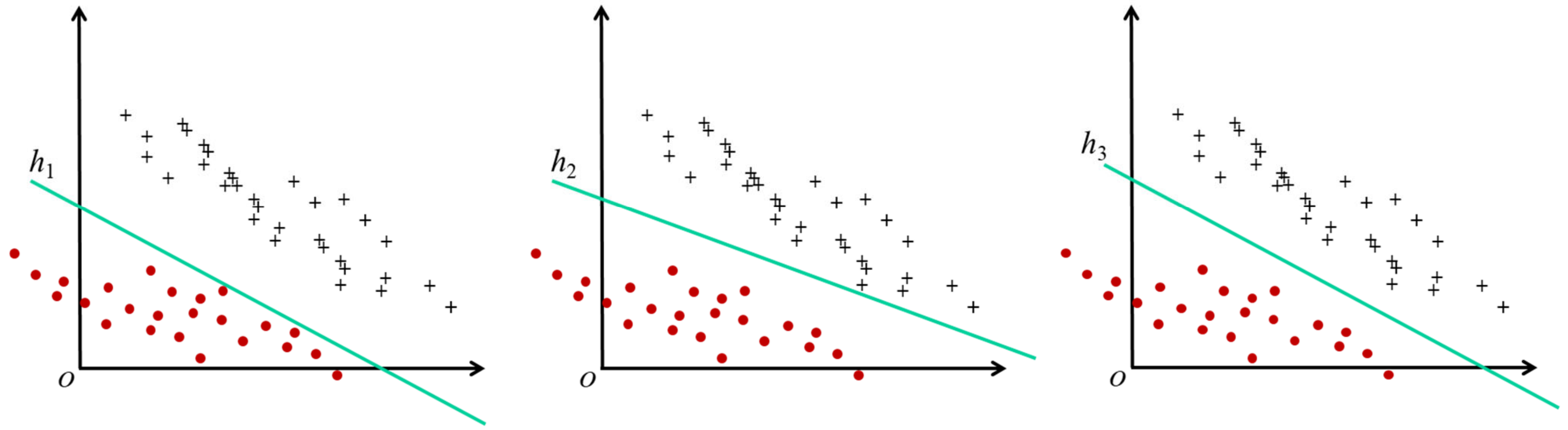


The updating rules for $\hat{\mathbf{w}}$

The perceptron learning algorithm

- The potential drawbacks of the perceptron learning algorithm
  - Since the hyperplane is randomly initialized and the misclassified example is randomly selected when updating the parameters, after running this algorithm several times, you may get several different separating hyperplanes
  - All of these different separating hyperplanes can correctly classify the training data, however, they are not equally good!

Example:



Which hyperplane is the best?

The perceptron learning algorithm

Given a linearly separable dataset, can we get the unique optimal separating hyperplane?

that comes $\Big\downarrow$

Hard-margin SVM

弗拉基米尔·瓦普尼克(Vladimir N. Vapnik，1936年12月6日-)，统计学家，因提出了支持向量机、VC理论等而著名。他出生于前苏联。1958年，他在撒马尔罕（现属乌兹别克斯坦）的乌兹别克国立大学完成了硕士学业。1964年，他于莫斯科控制科学学院获得博士学位。毕业后，他一直在该校工作直到1990年，在此期间，他成为了该校计算机科学与研究系的系主任。1990 年底，弗拉基米尔·瓦普尼克移居美国，加入了位于新泽西州霍姆德尔的At&T贝尔实验室的自适应系统研究部门。1995年，他被伦敦大学聘为计算机与统计科学专业的教授。现在，他工作于新泽西州普林斯顿的NEC实验室。他同时是哥伦比亚大学的特聘教授。2006年，他成为美国国家工程院院士。

Outline

Redefine the separating hyperplane

For a linearly separable training dataset $\mathcal{D}=\left\{\left(\mathbf{x}_i, y_i\right):|\ \mathbf{x}_i \in \mathbb{R}^d, y_i \in \{+1,-1\}\right\}_{i=1}^{n}$, we have already know what is the separating hyperplane. If $\mathbf{w}^T\mathbf{x}+b=0$ is the separating hyperplane, it must satisfy,

$$\forall\left(\mathbf{x}_i, y_i\right)\in\mathcal{D}, y_i\left(\mathbf{w}^T\mathbf{x}_i+b\right)>0$$

It needs to be noted that, the absolute magnitude of $y_i\left(\mathbf{w}^T\mathbf{x}_i+b\right)$ is not important for classification, since,

$\boxed{(\mathbf{w}, b)}$ $\boxed{\forall\rho>0,\left(\dfrac{\mathbf{w}}{\rho},\dfrac{b}{\rho}\right)}$ represent the same hyperplane

Redefine the separating hyperplane

For a separating hyperplane $(\mathbf{w}, b)$ of $\mathcal{D}$, we re-parameterize it as

$$(\mathbf{w}/\rho, b/\rho)$$

where $\rho = \min_{i=1,\dots,n} y_i\left(\mathbf{w}^T\mathbf{x}_i + b\right)$. Then, for the separating hyperplane $(\mathbf{w}/\rho, b/\rho)$, we have,

$$\min_{i=1,\dots,n} y_i\left(\left(\frac{\mathbf{w}}{\rho}\right)^T \mathbf{x}_i + \frac{b}{\rho}\right) = \frac{1}{\rho}\min_{i=1,\dots,n} y_i\left(\mathbf{w}^T\mathbf{x}_i + b\right) = \frac{\rho}{\rho} = 1$$

That is, for any separating hyperplane of $\mathcal{D}$, it can be parameterized as $(\mathbf{w}, b)$ satisfying,

$$\forall(\mathbf{x}_i, y_i) \in \mathcal{D}, y_i\left(\mathbf{w}^T\mathbf{x}_i + b\right) \geq 1$$

Definition 2: Separating hyperplane. For a training dataset $\mathcal{D} = \left\{(\mathbf{x}_i, y_i) :| \mathbf{x}_i \in \mathbb{R}^d, y_i \in \{+1, -1\}\right\}_{i=1}^n$ a hyperplane h is a separating hyperplane, if and only if it can be represented by $(\mathbf{w}, b)$ satisfying,

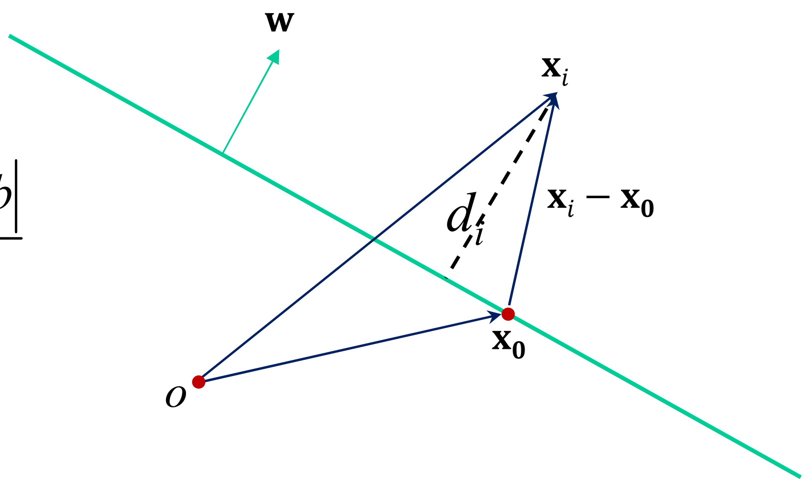
$$\min_{i=1,\dots,n} y_i\left(\mathbf{w}^T\mathbf{x}_i + b\right) = 1$$

Geometric margin

The **geometric margin** of a data point $(\mathbf{x}_i, y_i)$ with respect to the hyperplane (**w**, *b*) can tell us its distance to the hyperplane and can also **indicate whether it is correctly classified**

The Euclidean distance from $\mathbf{x}_i$ to the hyperplane,

$$d_i = \left|\frac{\mathbf{w}}{\|\mathbf{w}\|}\cdot(\mathbf{x}_i - \mathbf{x}_0)\right| = \frac{|\mathbf{w}\cdot(\mathbf{x}_i - \mathbf{x}_0)|}{\|\mathbf{w}\|} = \frac{|\mathbf{w}^T\mathbf{x}_i - \mathbf{w}^T\mathbf{x}_0|}{\|\mathbf{w}\|} = \frac{|\mathbf{w}^T\mathbf{x}_i + b|}{\|\mathbf{w}\|}$$



Based on d_i, we can derive the formula of the geometric margin of $(\mathbf{x}_i, y_i)$

$$\gamma_i = \frac{y_i\left(\mathbf{w}^T\mathbf{x}_i + b\right)}{\|\mathbf{w}\|}$$

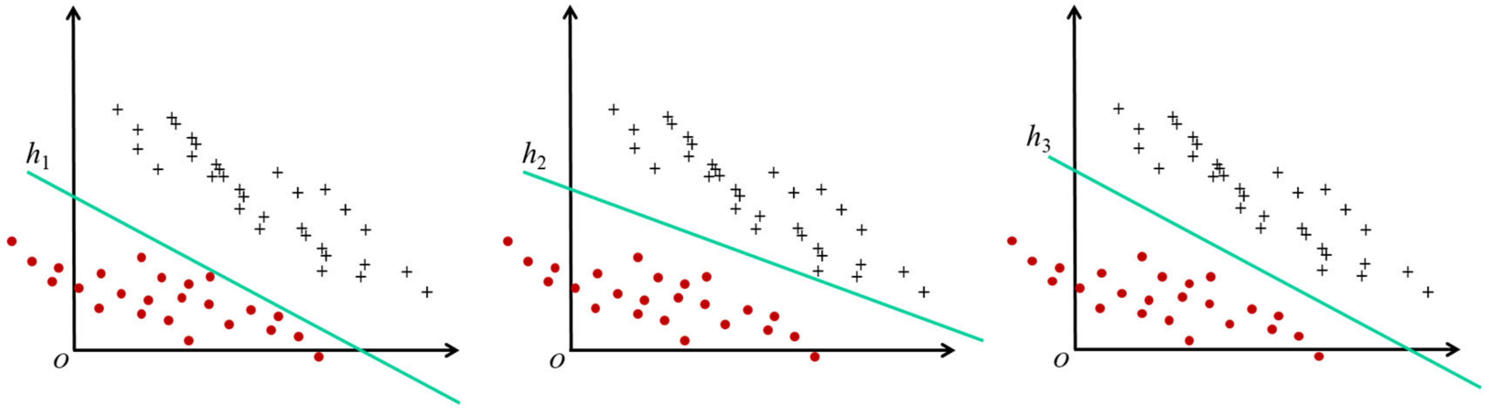
Do you notice the difference between the Euclidean distance and the geometric margin (from the data point to the hyperplane)?

Geometric margin

Based on the geometric margins of the points, we can define the geometric margin of a hyperplane

Definition 3: Geometric margin of a hyperplane. For a training dataset $\mathcal{D}$, the geometric margin of a hyperplane $\mathbf{w}^T\mathbf{x}+b=0$ is,

$$\gamma = \min_{i=1,2,..,n} \gamma_i = \min_{i=1,2,..,n} \frac{y_i\left(\mathbf{w}^T\mathbf{x}_i+b\right)}{\|\mathbf{w}\|} = \frac{1}{\|\mathbf{w}\|} \min_{i=1,2,..,n} y_i\left(\mathbf{w}^T\mathbf{x}_i+b\right)$$



Why do we think h_3 is the best? It has the largest geometric margin!

Hard-margin Support Vector Machine

For a linearly separable dataset $\mathcal{D}$, its optimal separating hyperplane $(\mathbf{w}^*, b^*)$ is the one having the largest geometric margin, i.e.,

$$\mathbf{w}^*, b^* = \arg\max_{\mathbf{w},b} \gamma = \arg\max_{\mathbf{w},b} \left(\frac{1}{\|\mathbf{w}\|} \min_{i=1,2,..,n} y_i \left(\mathbf{w}^T \mathbf{x}_i + b \right) \right) = \arg\max_{\mathbf{w},b} \frac{1}{\|\mathbf{w}\|}$$

$$\text{subject to } \min_{i=1,2,..,n} y_i \left(\mathbf{w}^T \mathbf{x}_i + b \right) = 1$$

where the constraint indicates that the desired hyperplane at first should be a separating hyperplane

The above problem can be reformulated as,

$$\mathbf{w}^*, b^* = \arg\min_{\mathbf{w},b} \frac{1}{2} \mathbf{w}^T \mathbf{w}$$

$$\text{subject to } y_i \left(\mathbf{w}^T \mathbf{x}_i + b \right) \geq 1, i = 1, \ldots, n$$

Hard-margin Support Vector Machine

Definition 4: Hard-margin SVM. Suppose that the training dataset $\mathcal{D}$ is linearly separable. The classification approach identifying the optimal separating hyperplane by solving the following problem is called the **hard-margin SVM**,

$$\mathbf{w}^*, b^* = \arg\min_{\mathbf{w},b} \frac{1}{2}\mathbf{w}^T\mathbf{w} \qquad \textbf{(Eq. 1)}$$

$$\text{subject to } y_i\left(\mathbf{w}^T\mathbf{x}_i + b\right) \geq 1, i = 1, \ldots, n$$

"Hard-margin" means that for every training sample, we require that its geometric margin should be $\geq 1/\|\mathbf{w}\|$

When we get the optimal separating hyperplane ($\mathbf{w}^*$, b^*), if the sample ($\mathbf{x}_k$, y_k) satisfies,

$$y_k\left(\mathbf{w}^{*T}\mathbf{x}_k + b^*\right) = 1$$

it is the **supporting vector** for the hyperplane ($\mathbf{w}^*$, b^*)

Hard-margin Support Vector Machine

Proposition 1: The problem for solving the hard-margin SVM (Eq. 1) is a convex quadratic program problem.

$$\mathbf{w}^*, b^* = \arg\min_{\mathbf{w},b} \frac{1}{2}\mathbf{w}^T\mathbf{w}$$

$$\text{subject to } y_i\left(\mathbf{w}^T\mathbf{x}_i + b\right) \geq 1, i = 1, \ldots, n$$

(Eq. 1)

Denote by $\mathbf{u} = \begin{pmatrix} b \\ \mathbf{w} \end{pmatrix}$. Then, $\mathbf{w}^T\mathbf{w} = \mathbf{u}^T \begin{bmatrix} 0 & \mathbf{0}_{1\times d} \\ \mathbf{0}_{d\times 1} & \mathbf{I}_{d\times d} \end{bmatrix} \mathbf{u}$, $-y_i\left(\mathbf{w}^T\mathbf{x}_i + b\right) + 1 = \left(-y_i \quad -y_i\mathbf{x}_i^T\right)\mathbf{u} + 1$

Eq. 1 can be reformulated as,

$$\mathbf{u}^* = \arg\min_{\mathbf{u}} \frac{1}{2}\mathbf{u}^T \begin{bmatrix} 0 & \mathbf{0}_{1\times d} \\ \mathbf{0}_{d\times 1} & \mathbf{I}_{d\times d} \end{bmatrix} \mathbf{u}$$

$$\text{subject to } \left(-y_i \quad -y_i\mathbf{x}_i^T\right)\mathbf{u} + 1 \leq 0, i = 1, \ldots, n$$

(Eq. 2)

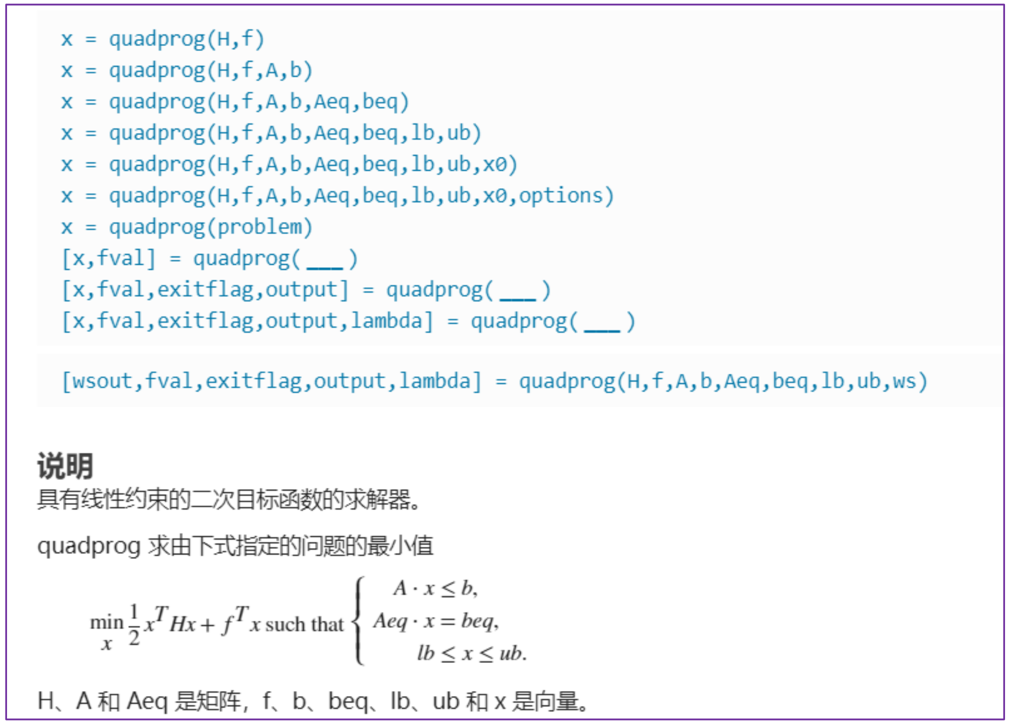
According to Def. 14 in Chapter 13, Eq. 2 is a convex quadratic program problem

Hard-margin Support Vector Machine

Since the problem of hard-margin SVM is a convex quadratic program problem, it can be solved by using standard packages for solving convex quadratic program problems

Example:

The 'quadprog' routine in matlab



```
x = quadprog(H,f)
x = quadprog(H,f,A,b)
x = quadprog(H,f,A,b,Aeq,beq)
x = quadprog(H,f,A,b,Aeq,beq,lb,ub)
x = quadprog(H,f,A,b,Aeq,beq,lb,ub,x0)
x = quadprog(H,f,A,b,Aeq,beq,lb,ub,x0,options)
x = quadprog(problem)
[x,fval] = quadprog(___)
[x,fval,exitflag,output] = quadprog(___)
[x,fval,exitflag,output,lambda] = quadprog(___)
```

```
[wsout,fval,exitflag,output,lambda] = quadprog(H,f,A,b,Aeq,beq,lb,ub,ws)
```

说明

具有线性约束的二次目标函数的求解器。

quadprog 求由下式指定的问题的最小值

$$\min_x \frac{1}{2}x^T Hx + f^T x \text{ such that } \begin{cases} A \cdot x \leq b, \\ Aeq \cdot x = beq, \\ lb \leq x \leq ub. \end{cases}$$

H、A 和 Aeq 是矩阵，f、b、beq、lb、ub 和 x 是向量。

By exploring the special characteristics of SVM, more efficient algorithms, based on duality, have been designed

Hard-margin SVM solver based on duality

The primal problem,

$$\mathbf{w}^*, b^* = \arg\min_{\mathbf{w},b} \frac{1}{2}\mathbf{w}^T\mathbf{w}$$

$$\text{subject to } -y_i\left(\mathbf{w}^T\mathbf{x}_i + b\right) + 1 \le 0, i = 1, \ldots, n$$

(Eq. 1)

Its Lagrangian (Def. 15 of Chapter 13) is,

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

where $\boldsymbol{\alpha} = (\alpha_1, \alpha_2, \ldots, \alpha_n)^T$

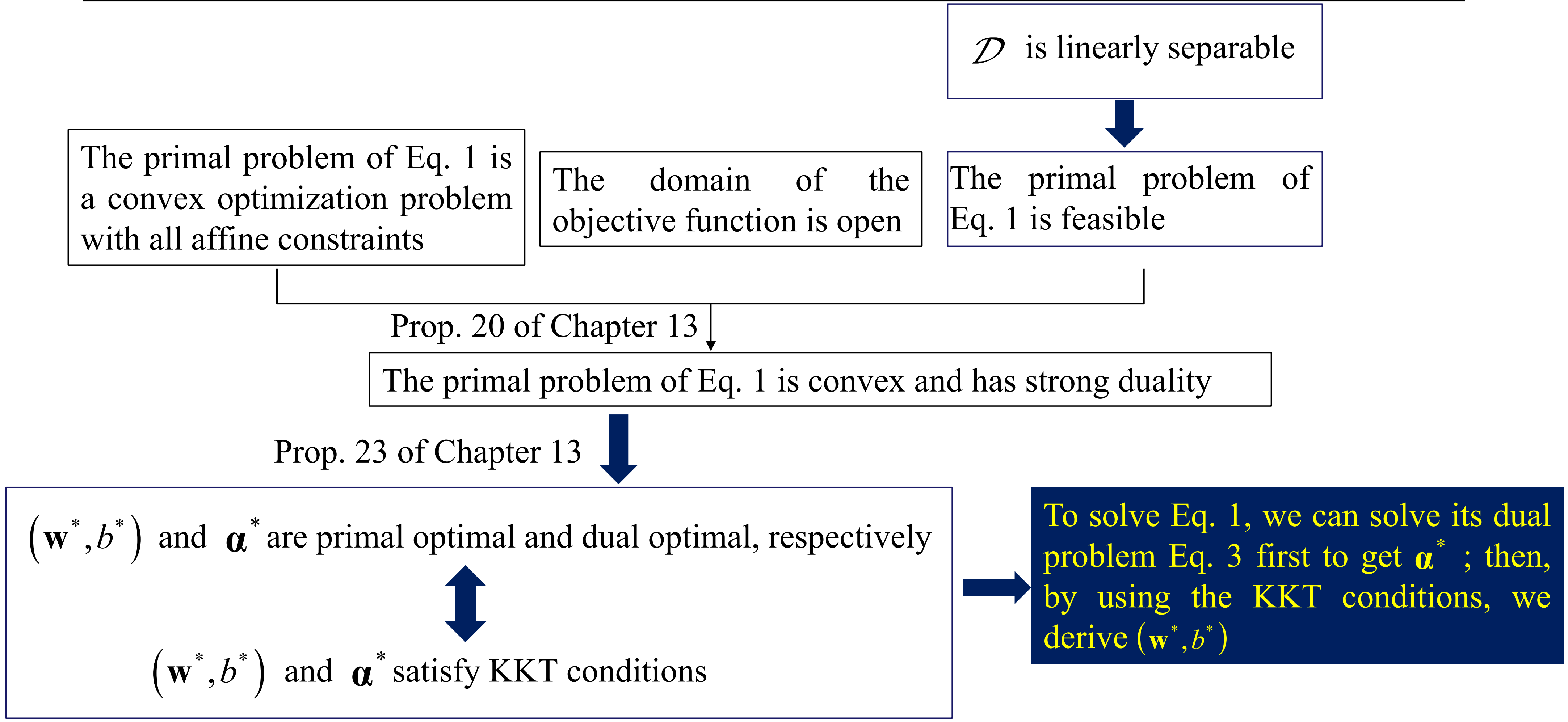
The dual problem (Def. 17 of Chapter 13) of Eq. 1 is,

$$\boldsymbol{\alpha}^* = \arg\max_{\boldsymbol{\alpha}} \left\{\min_{\mathbf{w},b} l(\mathbf{w}, b, \boldsymbol{\alpha})\right\}$$

$$\text{subject to } \boldsymbol{\alpha} \ge \mathbf{0}$$

(Eq. 3)

Hard-margin SVM solver based on duality



$\mathcal{D}$ is linearly separable

The primal problem of Eq. 1 is a convex optimization problem with all affine constraints

The domain of the objective function is open

The primal problem of Eq. 1 is feasible

Prop. 20 of Chapter 13

The primal problem of Eq. 1 is convex and has strong duality

Prop. 23 of Chapter 13

$(\mathbf{w}^*, b^*)$ and $\boldsymbol{\alpha}^*$ are primal optimal and dual optimal, respectively

$(\mathbf{w}^*, b^*)$ and $\boldsymbol{\alpha}^*$ satisfy KKT conditions

To solve Eq. 1, we can solve its dual problem Eq. 3 first to get $\boldsymbol{\alpha}^*$; then, by using the KKT conditions, we derive $(\mathbf{w}^*, b^*)$

Hard-margin SVM solver based on duality

First, solve the dual problem, Eq. 3, to get the dual optimal solution $\boldsymbol{\alpha}^*$

(1) Compute $\min_{\mathbf{w},b} l(\mathbf{w},b,\boldsymbol{\alpha})$

$$l_1(\mathbf{w},b)=\frac{1}{2}\mathbf{w}^T\mathbf{w}-\sum_{i=1}^{n}\alpha_i y_i\left(\mathbf{w}^T\mathbf{x}_i+b\right)+\sum_{i=1}^{n}\alpha_i$$

(about ($\mathbf{w}$, b), it is a convex function)

Get its stationary point by solving,

$$\begin{cases}\dfrac{\partial l_1}{\partial \mathbf{w}}=\mathbf{0}\\[2ex]\dfrac{\partial l_1}{\partial b}=0\end{cases}\Rightarrow\begin{cases}\mathbf{w}-\sum_{i=1}^{n}\alpha_i y_i\mathbf{x}_i=\mathbf{0}\\[2ex]-\sum_{i=1}^{n}\alpha_i y_i=0\end{cases}\Rightarrow\begin{cases}\mathbf{w}=\sum_{i=1}^{n}\alpha_i y_i\mathbf{x}_i\\[2ex]\sum_{i=1}^{n}\alpha_i y_i=0\end{cases}$$

$$\min_{\mathbf{w},b} l(\mathbf{w},b,\boldsymbol{\alpha})=\frac{1}{2}\left(\sum_{i=1}^{n}\alpha_i y_i\mathbf{x}_i\right)\cdot\left(\sum_{j=1}^{n}\alpha_j y_j\mathbf{x}_j\right)-\sum_{i=1}^{n}\alpha_i y_i\left(\left(\sum_{j=1}^{n}\alpha_j y_j\mathbf{x}_j\right)\cdot\mathbf{x}_i+b\right)+\sum_{i=1}^{n}\alpha_i$$

$$=\frac{1}{2}\left(\sum_{i=1}^{n}\alpha_i y_i\mathbf{x}_i\right)\cdot\left(\sum_{j=1}^{n}\alpha_j y_j\mathbf{x}_j\right)-\sum_{i=1}^{n}\alpha_i y_i\mathbf{x}_i\cdot\left(\sum_{j=1}^{n}\alpha_j y_j\mathbf{x}_j\right)-b\sum_{i=1}^{n}\alpha_i y_i+\sum_{i=1}^{n}\alpha_i=-\frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n}\alpha_i\alpha_j y_i y_j\mathbf{x}_i\cdot\mathbf{x}_j+\sum_{i=1}^{n}\alpha_i$$

Hard-margin SVM solver based on duality

First, solve the dual problem, Eq. 3, to get the dual optimal solution $\boldsymbol{\alpha}^*$

(2) Solve

$$\boldsymbol{\alpha}^* = \arg\max_{\boldsymbol{\alpha}} \left\{ -\frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n}\alpha_i\alpha_j y_i y_j \mathbf{x}_i \cdot \mathbf{x}_j + \sum_{i=1}^{n}\alpha_i \right\},$$

subject to $\boldsymbol{\alpha} \geq \mathbf{0}$

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

$\Rightarrow$

$$\boldsymbol{\alpha}^* = \arg\min_{\boldsymbol{\alpha}} \left\{ \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n}\alpha_i\alpha_j y_i y_j \mathbf{x}_i \cdot \mathbf{x}_j - \sum_{i=1}^{n}\alpha_i \right\},$$

subject to $-\boldsymbol{\alpha} \leq \mathbf{0}$

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

(Eq. 4)

Denote by $\mathbf{X} = \begin{bmatrix} y_1\mathbf{x}_1^T \\ y_2\mathbf{x}_2^T \\ \vdots \\ y_n\mathbf{x}_n^T \end{bmatrix}$, $\mathbf{Q} = \mathbf{X}\mathbf{X}^T = \begin{bmatrix} y_1y_1\mathbf{x}_1^T\mathbf{x}_1 & y_1y_2\mathbf{x}_1^T\mathbf{x}_2 \cdots y_1y_n\mathbf{x}_1^T\mathbf{x}_n \\ y_2y_1\mathbf{x}_2^T\mathbf{x}_1 & y_2y_2\mathbf{x}_2^T\mathbf{x}_2 \cdots y_ny_n\mathbf{x}_2^T\mathbf{x}_n \\ \vdots & \\ y_ny_1\mathbf{x}_n^T\mathbf{x}_1 & y_ny_2\mathbf{x}_n^T\mathbf{x}_2 \cdots y_ny_n\mathbf{x}_n^T\mathbf{x}_n \end{bmatrix}$

(positive semi-definite)

$$\boldsymbol{\alpha}^* = \arg\min_{\boldsymbol{\alpha}} \left\{ \frac{1}{2}\boldsymbol{\alpha}^T\mathbf{Q}\boldsymbol{\alpha} - \mathbf{1}_{n\times 1}^T\boldsymbol{\alpha} \right\},$$

subject to $\begin{bmatrix} \mathbf{y}^T \\ -\mathbf{y}^T \\ -\mathbf{I}_{n\times n} \end{bmatrix} \boldsymbol{\alpha} \leq \mathbf{0}_{(n+2)\times 1}$

(Eq. 5)

It is also a convex quadratic program problem!

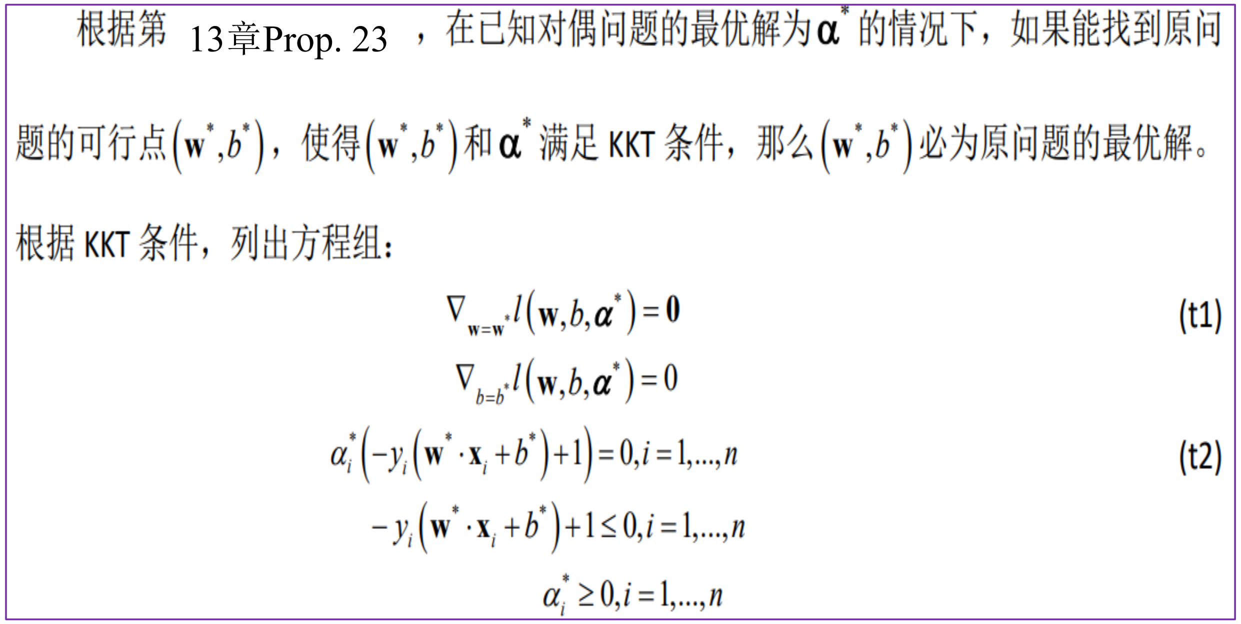
For solving the specific optimization problem of Eq. 5, algorithms more efficient than standard packages for solving the general convex quadratic program problems exist, such as the sequential minimal optimization, SMO

Hard-margin SVM solver based on duality

Second, with $\boldsymbol{\alpha}^*$, based on KKT conditions, derive the primal optimal solution $(\mathbf{w}^*, b^*)$

Proposition 2: Suppose that $\boldsymbol{\alpha}^* = (\alpha_1^*, \alpha_2^*, \ldots, \alpha_n^*)$ is the optimal solution for Eq. 3. There exists an j, making $\alpha_j^* > 0$, and $(\mathbf{w}^*, b^*)$ can be computed as,

$$\mathbf{w}^* = \sum_{i=1}^{n} \alpha_i^* y_i \mathbf{x}_i \qquad \textbf{(Eq. 6)} \qquad\qquad b^* = y_j - \sum_{i=1}^{n} \alpha_i^* y_i (\mathbf{x}_i \cdot \mathbf{x}_j) \qquad \textbf{(Eq. 7)}$$



根据第 13章Prop. 23 ，在已知对偶问题的最优解为$\boldsymbol{\alpha}^*$的情况下，如果能找到原问题的可行点$(\mathbf{w}^*, b^*)$，使得$(\mathbf{w}^*, b^*)$和$\boldsymbol{\alpha}^*$满足 KKT 条件，那么$(\mathbf{w}^*, b^*)$必为原问题的最优解。

根据 KKT 条件，列出方程组：

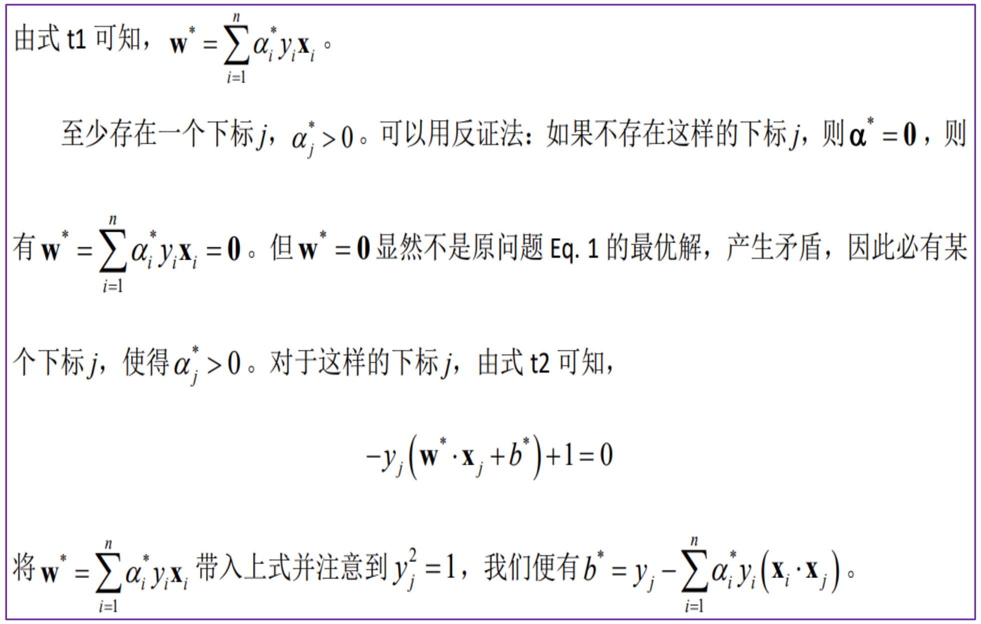
$$\nabla_{\mathbf{w}=\mathbf{w}^*} l(\mathbf{w}, b, \boldsymbol{\alpha}^*) = \mathbf{0} \qquad \text{(t1)}$$

$$\nabla_{b=b^*} l(\mathbf{w}, b, \boldsymbol{\alpha}^*) = 0$$

$$\alpha_i^*\left(-y_i\left(\mathbf{w}^* \cdot \mathbf{x}_i + b^*\right) + 1\right) = 0, i = 1, \ldots, n \qquad \text{(t2)}$$

$$-y_i\left(\mathbf{w}^* \cdot \mathbf{x}_i + b^*\right) + 1 \le 0, i = 1, \ldots, n$$

$$\alpha_i^* \ge 0, i = 1, \ldots, n$$



由式 t1 可知，$\mathbf{w}^* = \sum_{i=1}^{n} \alpha_i^* y_i \mathbf{x}_i$。

至少存在一个下标j，$\alpha_j^* > 0$。可以用反证法：如果不存在这样的下标j，则$\boldsymbol{\alpha}^* = \mathbf{0}$，则有$\mathbf{w}^* = \sum_{i=1}^{n} \alpha_i^* y_i \mathbf{x}_i = \mathbf{0}$。但$\mathbf{w}^* = \mathbf{0}$显然不是原问题 Eq. 1 的最优解，产生矛盾，因此必有某个下标j，使得$\alpha_j^* > 0$。对于这样的下标j，由式 t2 可知，

$$-y_j\left(\mathbf{w}^* \cdot \mathbf{x}_j + b^*\right) + 1 = 0$$

将$\mathbf{w}^* = \sum_{i=1}^{n} \alpha_i^* y_i \mathbf{x}_i$带入上式并注意到$y_j^2 = 1$，我们便有$b^* = y_j - \sum_{i=1}^{n} \alpha_i^* y_i (\mathbf{x}_i \cdot \mathbf{x}_j)$。

Hard-margin SVM solver based on duality

Second, with $\boldsymbol{\alpha}^*$, based on KKT conditions, derive the primal optimal solution ($\mathbf{w}^*$, b^*)

Proposition 2: Suppose that $\boldsymbol{\alpha}^* = \left(\alpha_1^*, \alpha_2^*, ..., \alpha_n^*\right)$ is the optimal solution for Eq. 3. There exists an j, making $\alpha_j^* > 0$, and ($\mathbf{w}^*$, b^*) can be computed as,

$$\mathbf{w}^* = \sum_{i=1}^{n} \alpha_i^* y_i \mathbf{x}_i \qquad \textbf{(Eq. 6)} \qquad\qquad b^* = y_j - \sum_{i=1}^{n} \alpha_i^* y_i \left(\mathbf{x}_i \cdot \mathbf{x}_j\right) \qquad \textbf{(Eq. 7)}$$

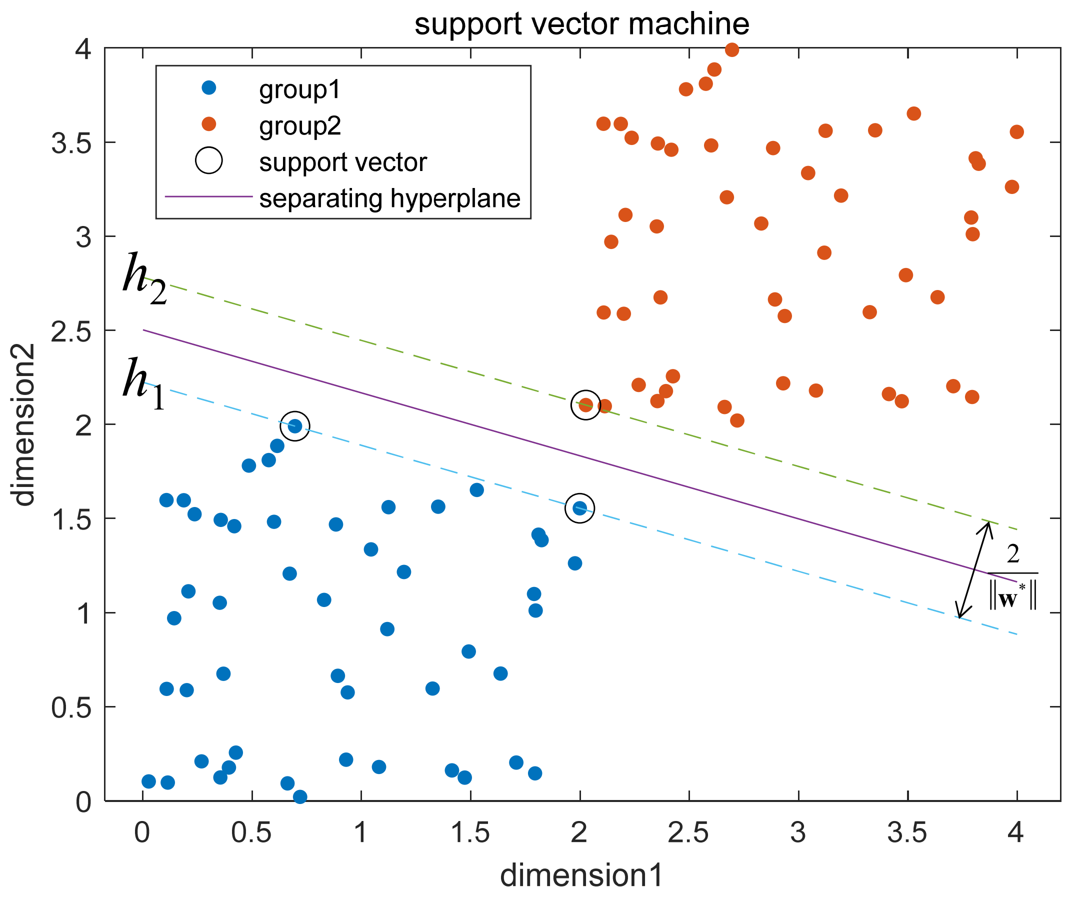
With the optimal separating hyperplane ($\mathbf{w}^*$, b^*), the classification decision function can be,

$$h_{\mathbf{w}^*, b^*}\left(\mathbf{x}\right) = \text{sign}\left(\sum_{i=1}^{n} \alpha_i^* y_i \left(\mathbf{x} \cdot \mathbf{x}_i\right) + b^*\right)$$

Actually, only a few elements in $\boldsymbol{\alpha}^*$ are greater than 0. If $\alpha_i^* > 0$, the associated feature vector of the training sample ($\mathbf{x}_i$, y_i) is a support vector to the hyperplane ($\mathbf{w}^*$, b^*) since $y_i\left(\mathbf{w}^{*T}\mathbf{x}_i + b^*\right) = 1$ holds

Hard-margin SVM

Example:



Outline

From hard-margin SVM to soft-margin SVM

- When the training dataset is linearly separable, the hard-margin SVM can work perfect
- However, when the training dataset is not linearly separable, the hard-margin SVM does not work since the separating hyperplane does not exist, i.e., the following optimization problem is not feasible,

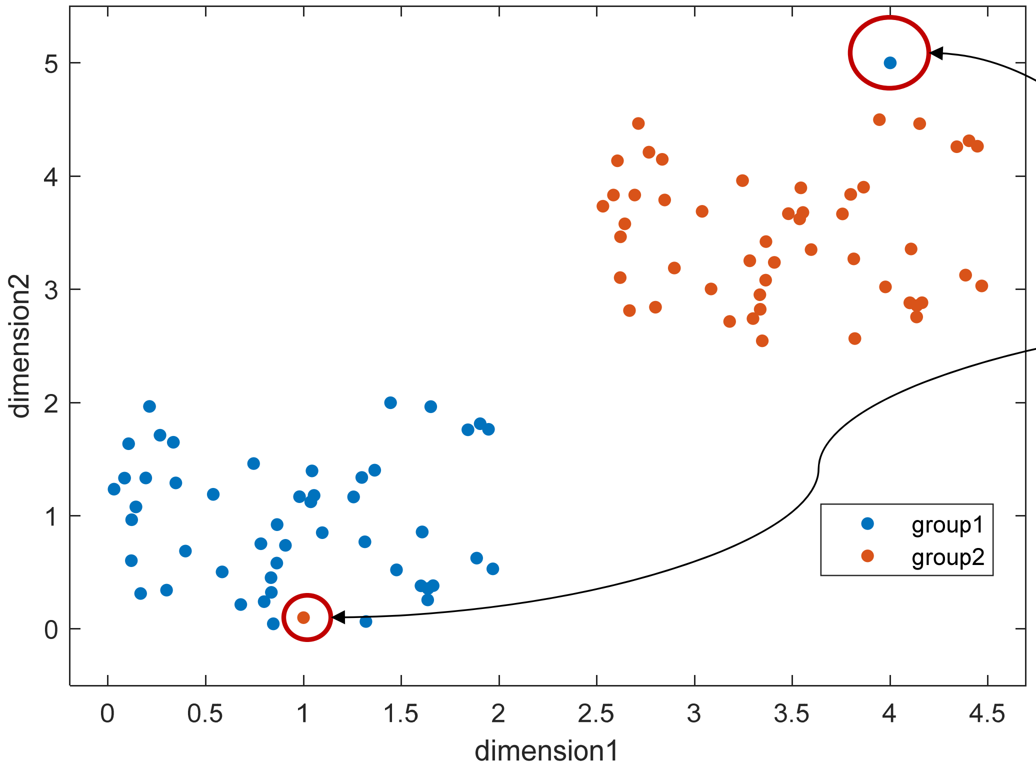
$$\mathbf{w}^*, b^* = \arg\min_{\mathbf{w},b} \frac{1}{2}\mathbf{w}^T\mathbf{w}$$
$$\text{subject to } y_i\left(\mathbf{w}^T\mathbf{x}_i + b\right) \geq 1, i = 1, \ldots, n \qquad \textbf{(Eq. 1)}$$

How to deal with the dataset that is not linearly separable?

Let's first consider a relatively simple case ➡

From hard-margin SVM to soft-margin SVM

The training dataset $\mathcal{D} = \left\{ \left(\mathbf{x}_i, y_i \right) :| \mathbf{x}_i \in \mathbb{R}^d, y_i \in \{+1,-1\} \right\}_{i=1}^{n}$ is not linearly separable; however, after removing the outliers, the remaining points are linearly separable



These two points are outliers; by removing them, the dataset would be linearly separable

Can we deal with this case by extending the hard-margin SVM?

From hard-margin SVM to soft-margin SVM

The training dataset $\mathcal{D}=\left\{\left(\mathbf{x}_i, y_i\right) :| \mathbf{x}_i \in \mathbb{R}^d, y_i \in \{+1,-1\}\right\}_{i=1}^{n}$ is not linearly separable; however, after removing the outliers, the remaining points are linearly separable

$$\mathbf{w}^*, b^* = \arg\min_{\mathbf{w},b} \frac{1}{2}\mathbf{w}^T\mathbf{w}$$

$$\boxed{\text{subject to } y_i\left(\mathbf{w}^T\mathbf{x}_i + b\right) \geq 1, i = 1, \ldots, n}$$

(Eq. 1)

Relax it to,

$$\boxed{\begin{array}{c}\text{subject to } y_i\left(\mathbf{w}^T\mathbf{x}_i + b\right) \geq 1 - \xi_i, i = 1, \ldots, n, \\ \xi_i \geq 0, i = 1, \ldots, n,\end{array}}$$

For every hyperplane (**w**, b), for each training sample ($\mathbf{x}_i$, y_i), $\exists \xi_i \geq 0$, making

$$y_i\left(\mathbf{w}^T\mathbf{x}_i + b\right) \geq 1 - \xi_i$$

satisfied. Of course, smaller $\{\xi_i\}$ are preferred, i.e., large $\{\xi_i\}$ need to be penalized

➡ These motivations suggest the following modified SVM model

Soft-margin SVM

Definition 5: Soft-margin SVM. Suppose that the training dataset $\mathcal{D}$ is **nearly linearly separable**. The classification approach identifying the optimal separating hyperplane by solving the following problem is called the **soft-margin SVM**,

$$\mathbf{w}^*, b^* = \arg\min_{\mathbf{w}, b, \xi} \frac{1}{2}\mathbf{w}^T\mathbf{w} + C\sum_{i=1}^{n} \xi_i \qquad \textbf{(Eq. 8)}$$

$$\text{subject to } y_i\left(\mathbf{w}^T\mathbf{x}_i + b\right) \geq 1 - \xi_i, i = 1, \ldots, n$$

$$\xi_i \geq 0, i = 1, \ldots, n$$

- By letting $C = 0$, $\xi_i = 0, i = 1, \ldots, n$, the soft-margin SVM model degenerates to the hard-margin one; the soft-margin SVM of course can deal with the case when the training dataset is linearly separable; thus, the soft-margin SVM "includes" the hard-margin SVM, i.e., the hard-margin SVM is a special case of the soft-margin one
- The soft-margin SVM is a linear model since its decision boundary is a hyperplane; so, it is also called **linear SVM**; it can deal with case when the training dataset is nearly linear separable

Soft-margin SVM solver based on duality

Proposition 3: The problem for solving the soft-margin SVM (Eq. 8) is a convex quadratic program problem

For proof, refer to the textbook

Similar as the case of the hard-margin SVM, we can solve the soft-margin SVM using the duality theory

Soft-margin SVM solver based on duality

The primal problem,

$$\mathbf{w}^*, b^* = \arg\min_{\mathbf{w},b} \frac{1}{2}\mathbf{w}^T\mathbf{w} + C\sum_{i=1}^{n}\xi_i$$

$$\text{subject to } y_i\left(\mathbf{w}^T\mathbf{x}_i + b\right) \geq 1 - \xi_i, i = 1,\ldots,n \qquad \textbf{(Eq. 8)}$$

$$\xi_i \geq 0, i = 1,\ldots,n$$

Its Lagrangian (Def. 15 of Chapter 13) is,

$$l\left(\left(\mathbf{w}, b, \boldsymbol{\xi}\right), \boldsymbol{\alpha}, \boldsymbol{\mu}\right) = \frac{1}{2}\mathbf{w}^T\mathbf{w} + C\sum_{i=1}^{n}\xi_i + \sum_{i=1}^{n}\alpha_i\left(-y_i\left(\mathbf{w}^T\mathbf{x}_i + b\right) - \xi_i + 1\right) + \sum_{i=1}^{n}\mu_i\left(-\xi_i\right)$$

The dual problem (Def. 17 of Chapter 13) of Eq. 8 is,

$$\boldsymbol{\alpha}^*, \boldsymbol{\mu}^* = \arg\max_{\boldsymbol{\alpha},\boldsymbol{\mu}}\left\{\min_{(\mathbf{w},b,\boldsymbol{\xi})} l\left(\left(\mathbf{w}, b, \boldsymbol{\xi}\right), \boldsymbol{\alpha}, \boldsymbol{\mu}\right)\right\} \qquad \textbf{(Eq. 9)}$$

$$\text{subject to } \boldsymbol{\alpha} \geq \mathbf{0}$$

$$\boldsymbol{\mu} \geq \mathbf{0}$$

Soft-margin SVM solver based on duality

To solve Eq. 8, we can solve its dual problem Eq. 9 first to get $\left(\boldsymbol{\alpha}^*, \boldsymbol{\mu}^*\right)$; then, by using the KKT equations, we derive $\left(\mathbf{w}^*, b^*, \boldsymbol{\xi}^*\right)$

Soft-margin SVM solver based on duality

First, solve the dual problem, Eq. 9, to get the dual optimal solution $\left(\boldsymbol{\alpha}^*,\boldsymbol{\mu}^*\right)$

(1) Compute $\min_{(\mathbf{w},b,\boldsymbol{\xi})} l\left((\mathbf{w},b,\boldsymbol{\xi}),\boldsymbol{\alpha},\boldsymbol{\mu}\right)$

$$l_1(\mathbf{w},b,\boldsymbol{\xi})=\frac{1}{2}\mathbf{w}^T\mathbf{w}+C\sum_{i=1}^n\xi_i+\sum_{i=1}^n\alpha_i\left(-y_i\left(\mathbf{w}^T\mathbf{x}_i+b\right)-\xi_i+1\right)+\sum_{i=1}^n\mu_i\left(-\xi_i\right)\qquad \text{(about } (\mathbf{w},b,\boldsymbol{\xi}) \text{, it is a convex function)}$$

Get its stationary point by solving, $\begin{cases}\dfrac{\partial l_1}{\partial \mathbf{w}}=\mathbf{0}\\ \dfrac{\partial l_1}{\partial b}=0\\ \dfrac{\partial l_1}{\partial \xi_i}=0\end{cases}\Rightarrow\begin{cases}\mathbf{w}-\sum_{i=1}^n\alpha_i y_i\mathbf{x}_i=\mathbf{0}\\ -\sum_{i=1}^n\alpha_i y_i=0\\ C-\alpha_i-\mu_i=0\end{cases}\Rightarrow\begin{cases}\mathbf{w}=\sum_{i=1}^n\alpha_i y_i\mathbf{x}_i\\ \sum_{i=1}^n\alpha_i y_i=0\\ C-\alpha_i-\mu_i=0\end{cases}$

$$\min_{(\mathbf{w},b,\boldsymbol{\xi})} l\left((\mathbf{w},b,\boldsymbol{\xi}),\boldsymbol{\alpha},\boldsymbol{\mu}\right)=$$

$$\frac{1}{2}\left(\sum_{i=1}^n\alpha_i y_i\mathbf{x}_i\right)\cdot\left(\sum_{j=1}^n\alpha_j y_j\mathbf{x}_j\right)+C\sum_{i=1}^n\xi_i-\left(\sum_{i=1}^n\alpha_i y_i\mathbf{x}_i\right)\cdot\left(\sum_{j=1}^n\alpha_j y_j\mathbf{x}_j\right)-b\sum_{j=1}^n\alpha_j y_j-\sum_{i=1}^n\alpha_i\xi_i+\sum_{i=1}^n\alpha_i-\sum_{i=1}^n\mu_i\xi_i$$

$$=-\frac{1}{2}\left(\sum_{i=1}^n\alpha_i y_i\mathbf{x}_i\right)\cdot\left(\sum_{j=1}^n\alpha_j y_j\mathbf{x}_j\right)+\sum_{i=1}^n\alpha_i+\sum_{i=1}^n(C-\alpha_i-\mu_i)\xi_i=-\frac{1}{2}\sum_{i=1}^n\sum_{j=1}^n\alpha_i\alpha_j y_i y_j\mathbf{x}_i\cdot\mathbf{x}_j+\sum_{i=1}^n\alpha_i$$

Soft-margin SVM solver based on duality

First, solve the dual problem, Eq. 9, to get the dual optimal solution $\left(\boldsymbol{\alpha}^*, \boldsymbol{\mu}^*\right)$

(2) Solve

$$\boldsymbol{\alpha}^*, \boldsymbol{\mu}^* = \arg\max_{\boldsymbol{\alpha},\boldsymbol{\mu}} \left\{ -\frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n}\alpha_i\alpha_j y_i y_j \mathbf{x}_i \cdot \mathbf{x}_j + \sum_{i=1}^{n}\alpha_i \right\}$$

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

$\Rightarrow$

$$\boxed{\begin{aligned} &\boldsymbol{\alpha}^* = \arg\min_{\boldsymbol{\alpha}} \left\{ \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n}\alpha_i\alpha_j y_i y_j \mathbf{x}_i \cdot \mathbf{x}_j - \sum_{i=1}^{n}\alpha_i \right\} \\ &\text{subject to } C \geq \alpha_i \geq 0, i = 1,\ldots,n \qquad \textbf{(Eq. 10)} \\ &\sum_{i=1}^{n}\alpha_i y_i = 0 \end{aligned}}$$

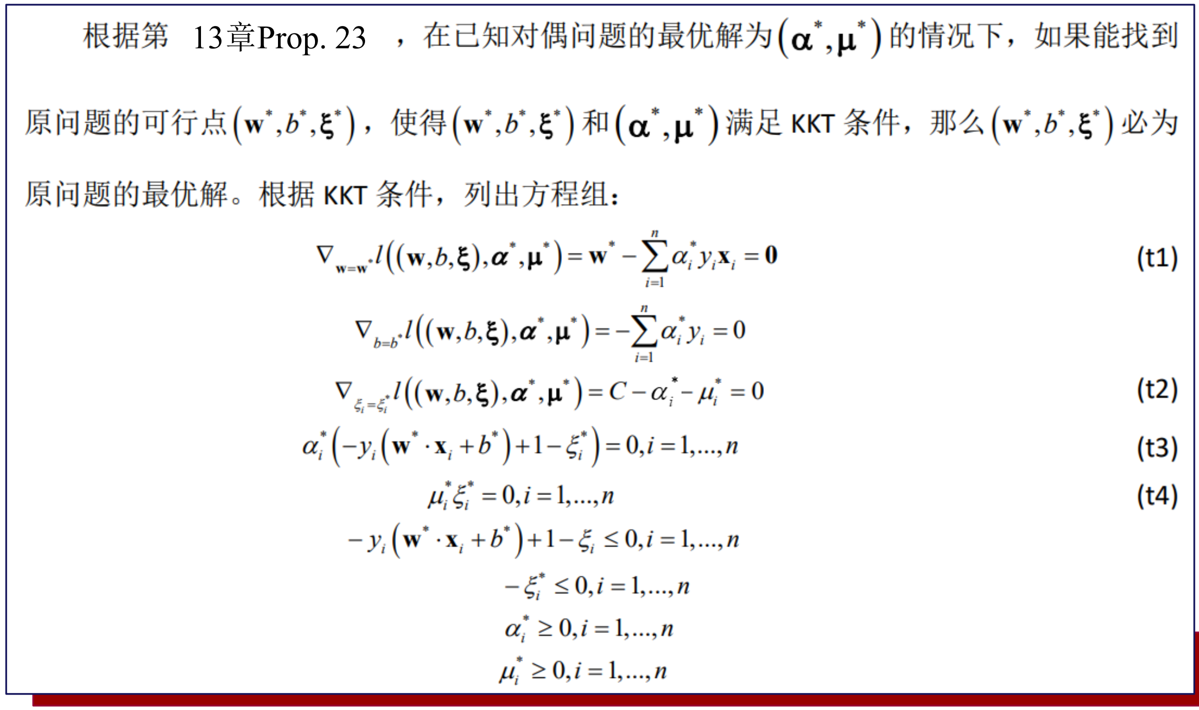
Make a comparison between Eq. 10 and Eq. 4, they are quite similar; Eq. 10 can also be efficiently solved by SMO

Soft-margin SVM solver based on duality

Second, with $(\boldsymbol{\alpha}^*, \boldsymbol{\mu}^*)$, based on KKT conditions, derive the primal optimal solution $(\mathbf{w}^*, b^*, \boldsymbol{\xi}^*)$

Proposition 4: Suppose that $\boldsymbol{\alpha}^* = (\alpha_1^*, \alpha_2^*, \ldots, \alpha_n^*)$ is the optimal solution for Eq. 9. If there exists an j, making $0 < \alpha_j^* < C$, then, $(\mathbf{w}^*, b^*)$ can be computed as,

$$\mathbf{w}^* = \sum_{i=1}^{n} \alpha_i^* y_i \mathbf{x}_i \qquad \textbf{(Eq. 11)} \qquad\qquad b^* = y_j - \sum_{i=1}^{n} \alpha_i^* y_i (\mathbf{x}_i \cdot \mathbf{x}_j) \qquad \textbf{(Eq. 12)}$$



根据第 13章Prop. 23 ，在已知对偶问题的最优解为$(\boldsymbol{\alpha}^*, \boldsymbol{\mu}^*)$的情况下，如果能找到原问题的可行点$(\mathbf{w}^*, b^*, \boldsymbol{\xi}^*)$，使得$(\mathbf{w}^*, b^*, \boldsymbol{\xi}^*)$和$(\boldsymbol{\alpha}^*, \boldsymbol{\mu}^*)$满足 KKT 条件，那么$(\mathbf{w}^*, b^*, \boldsymbol{\xi}^*)$必为原问题的最优解。根据 KKT 条件，列出方程组：

$$\nabla_{\mathbf{w}=\mathbf{w}^*} l((\mathbf{w}, b, \boldsymbol{\xi}), \boldsymbol{\alpha}^*, \boldsymbol{\mu}^*) = \mathbf{w}^* - \sum_{i=1}^{n} \alpha_i^* y_i \mathbf{x}_i = \mathbf{0} \qquad \text{(t1)}$$

$$\nabla_{b=b^*} l((\mathbf{w}, b, \boldsymbol{\xi}), \boldsymbol{\alpha}^*, \boldsymbol{\mu}^*) = -\sum_{i=1}^{n} \alpha_i^* y_i = 0$$

$$\nabla_{\xi_i=\xi_i^*} l((\mathbf{w}, b, \boldsymbol{\xi}), \boldsymbol{\alpha}^*, \boldsymbol{\mu}^*) = C - \alpha_i^* - \mu_i^* = 0 \qquad \text{(t2)}$$

$$\alpha_i^* \left(-y_i \left(\mathbf{w}^* \cdot \mathbf{x}_i + b^*\right) + 1 - \xi_i^*\right) = 0, i = 1, \ldots, n \qquad \text{(t3)}$$

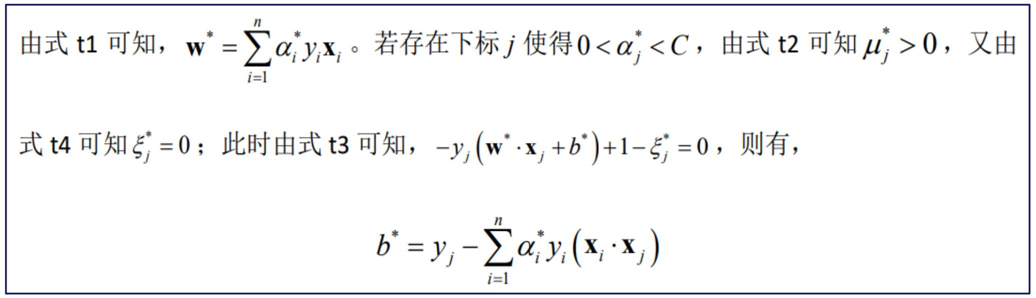
$$\mu_i^* \xi_i^* = 0, i = 1, \ldots, n \qquad \text{(t4)}$$

$$-y_i \left(\mathbf{w}^* \cdot \mathbf{x}_i + b^*\right) + 1 - \xi_i \le 0, i = 1, \ldots, n$$

$$-\xi_i^* \le 0, i = 1, \ldots, n$$

$$\alpha_i^* \ge 0, i = 1, \ldots, n$$

$$\mu_i^* \ge 0, i = 1, \ldots, n$$



由式 t1 可知，$\mathbf{w}^* = \sum_{i=1}^{n} \alpha_i^* y_i \mathbf{x}_i$。若存在下标 j 使得$0 < \alpha_j^* < C$，由式 t2 可知$\mu_j^* > 0$，又由式 t4 可知$\xi_j^* = 0$；此时由式 t3 可知，$-y_j\left(\mathbf{w}^* \cdot \mathbf{x}_j + b^*\right) + 1 - \xi_j^* = 0$，则有，

$$b^* = y_j - \sum_{i=1}^{n} \alpha_i^* y_i \left(\mathbf{x}_i \cdot \mathbf{x}_j\right)$$

Soft-margin SVM solver based on duality

Second, with $\left(\boldsymbol{\alpha}^*, \boldsymbol{\mu}^*\right)$, based on KKT conditions, derive the primal optimal solution $\left(\mathbf{w}^*, b^*, \boldsymbol{\xi}^*\right)$

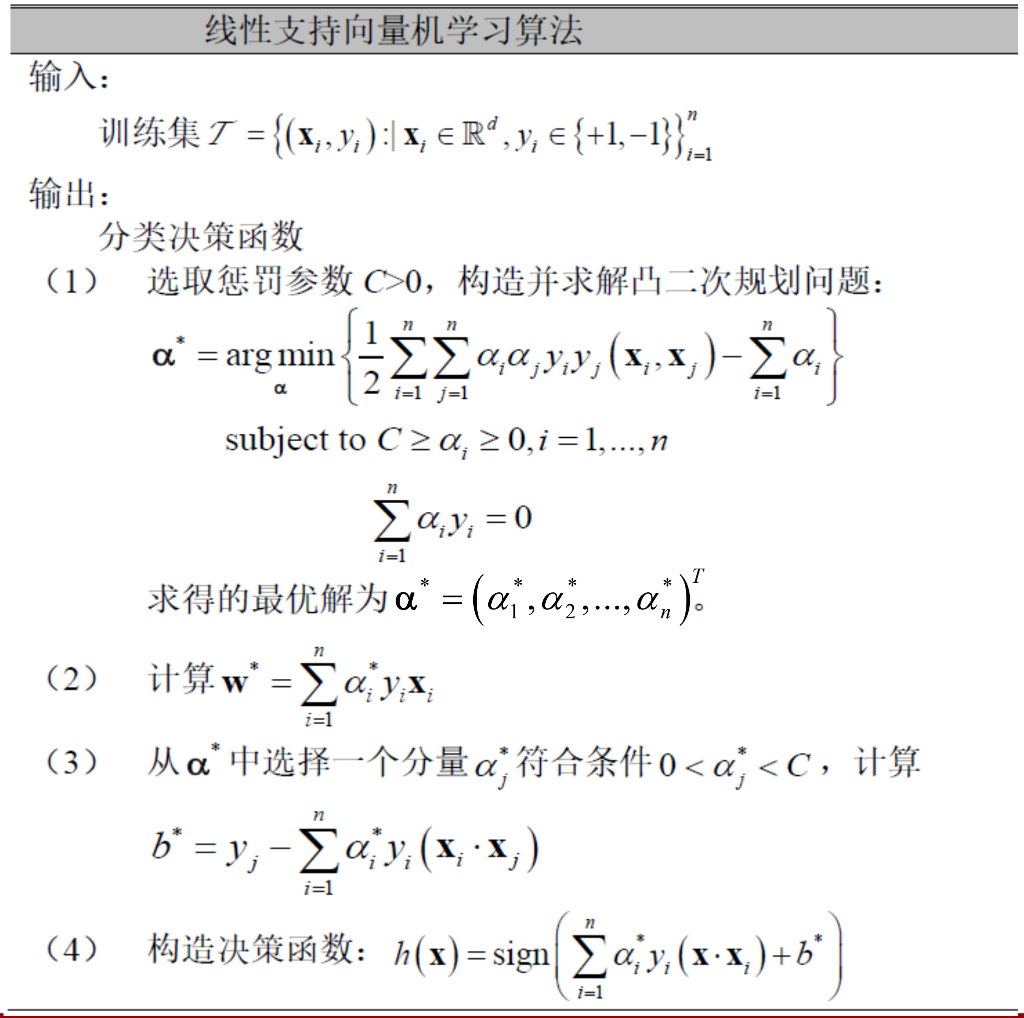
Proposition 4: Suppose that $\boldsymbol{\alpha}^* = \left(\alpha_1^*, \alpha_2^*, \dots, \alpha_n^*\right)$ is the optimal solution for Eq. 9. If there exists an *j*, making $0 < \alpha_j^* < C$, then, $(\mathbf{w}^*, b^*)$ can be computed as,

$$\mathbf{w}^* = \sum_{i=1}^{n} \alpha_i^* y_i \mathbf{x}_i \qquad \textbf{(Eq. 11)} \qquad\qquad b^* = y_j - \sum_{i=1}^{n} \alpha_i^* y_i \left(\mathbf{x}_i \cdot \mathbf{x}_j\right) \qquad \textbf{(Eq. 12)}$$

With the optimal separating hyperplane $(\mathbf{w}^*, b^*)$, the classification decision function can be,

$$h_{\mathbf{w}^*, b^*}(\mathbf{x}) = \operatorname{sign}\left(\sum_{i=1}^{n} \alpha_i^* y_i \left(\mathbf{x} \cdot \mathbf{x}_i\right) + b^*\right)$$

Soft-margin SVM



线性支持向量机学习算法

输入：

训练集 $T=\{(\mathbf{x}_i,y_i):|\,\mathbf{x}_i\in\mathbb{R}^d,y_i\in\{+1,-1\}\}_{i=1}^{n}$

输出：

分类决策函数

（1） 选取惩罚参数 $C>0$，构造并求解凸二次规划问题：

$$\boldsymbol{\alpha}^*=\arg\min_{\boldsymbol{\alpha}}\left\{\frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n}\alpha_i\alpha_j y_i y_j(\mathbf{x}_i,\mathbf{x}_j)-\sum_{i=1}^{n}\alpha_i\right\}$$

$$\text{subject to } C\ge\alpha_i\ge 0, i=1,\dots,n$$

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

求得的最优解为 $\boldsymbol{\alpha}^*=(\alpha_1^*,\alpha_2^*,\dots,\alpha_n^*)^T$。

（2） 计算 $\mathbf{w}^*=\sum_{i=1}^{n}\alpha_i^* y_i\mathbf{x}_i$

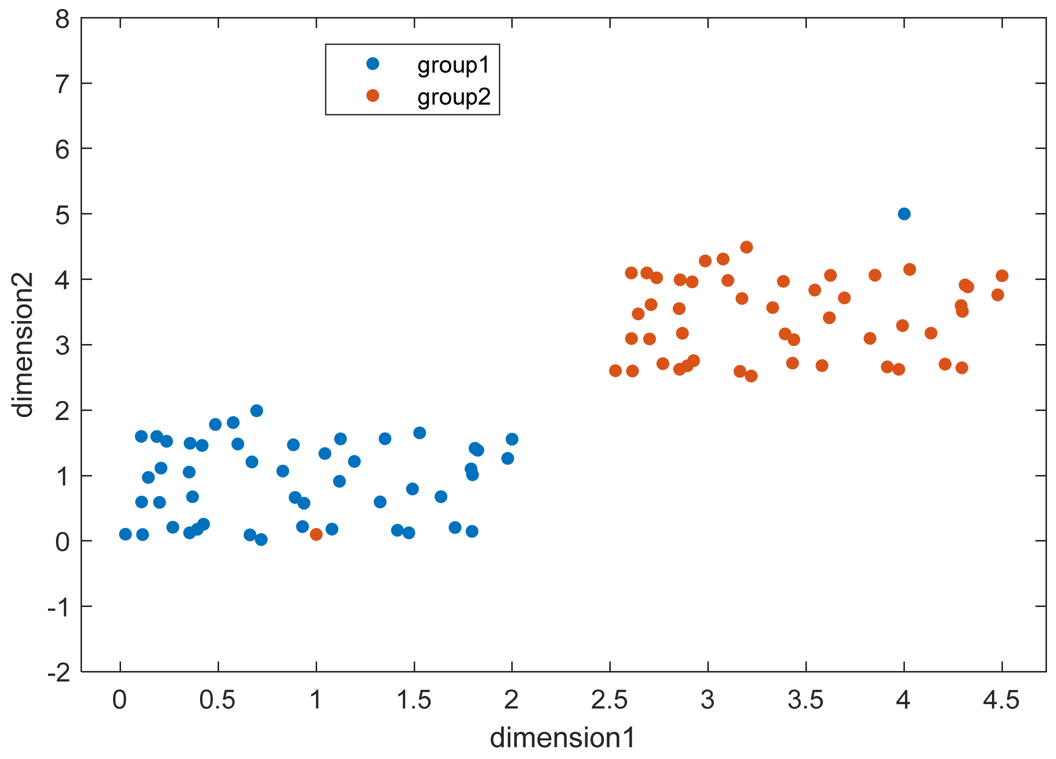
（3） 从 $\boldsymbol{\alpha}^*$ 中选择一个分量 α_j^* 符合条件 $0<\alpha_j^*<C$，计算

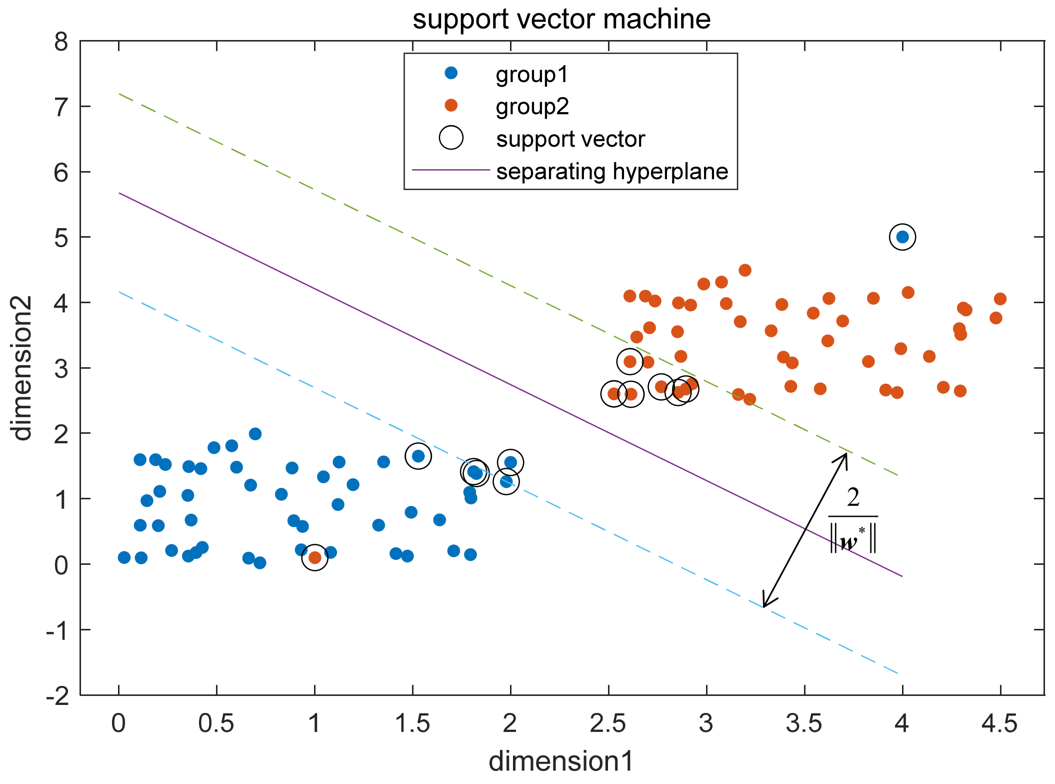
$$b^*=y_j-\sum_{i=1}^{n}\alpha_i^* y_i(\mathbf{x}_i\cdot\mathbf{x}_j)$$

（4） 构造决策函数：$h(\mathbf{x})=\text{sign}\left(\sum_{i=1}^{n}\alpha_i^* y_i(\mathbf{x}\cdot\mathbf{x}_i)+b^*\right)$

Soft-margin SVM

Example:





The training sample $(\mathbf{x}_i, y_i)$ corresponding to $\alpha_i^* > 0$ is the support vector and it must satisfy $y_i\left(\mathbf{w}^{*T}\mathbf{x}_i + b^*\right) = 1 - \xi_i$

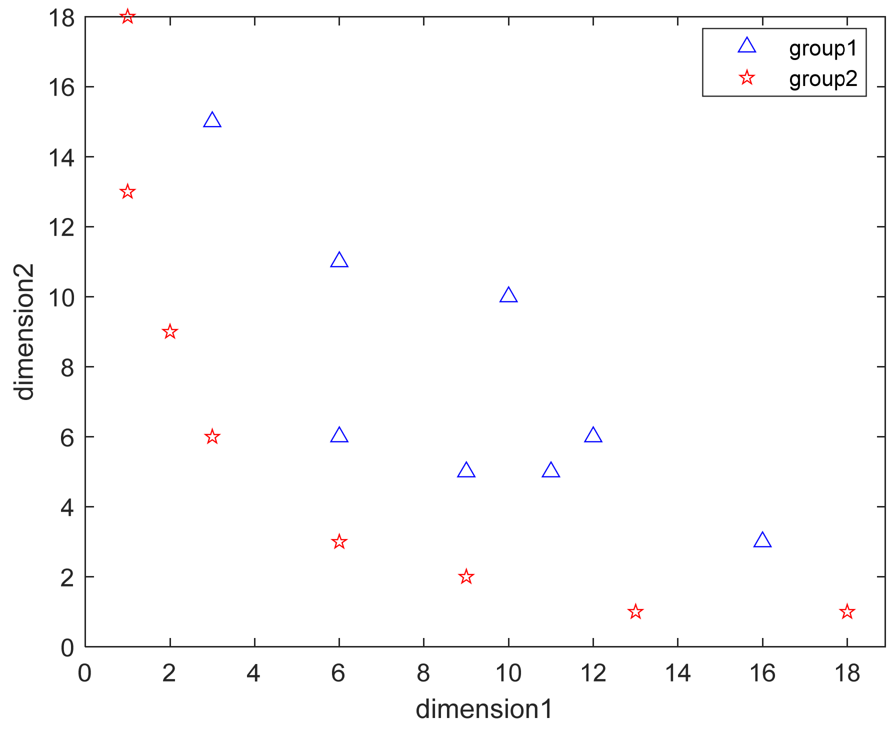
Outline

For nonlinear classification: Mapping from low dimensional space to a high dimensional one

- When the dataset is linearly separable or nearly linearly separable, linear SVM (soft-margin SVM) can work well
- However, when the dataset is not nearly linearly separable, the performance of linear SVM will be poor

Consider a dataset of 2D points as the right figure. You cannot find a proper "separating line" that can separate the two classes of data

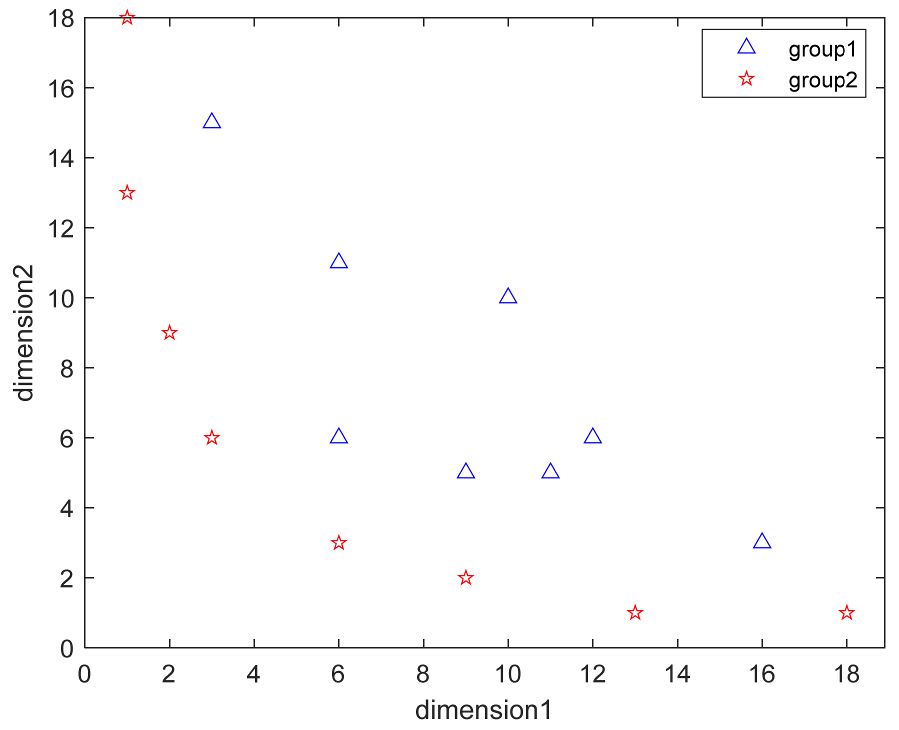
However, a dataset being not linearly separable in its **original space** does not necessarily mean that it is not linearly separable when being mapped to another **high-dimensional space**!

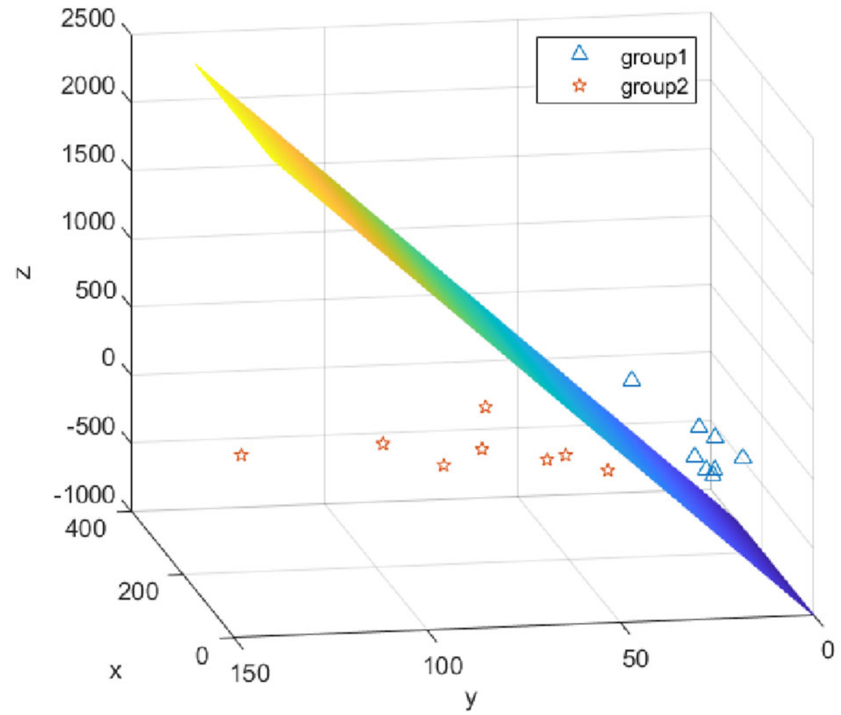


For nonlinear classification: Mapping from low dimensional space to a high dimensional one

For example, using the following mapping $\phi : \mathbb{R}^2 \rightarrow \mathbb{R}^3$,

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





The mapped data in the high-dimensional space is now linearly separable!

For nonlinear classification: Mapping from low dimensional space to a high dimensional one

From this example, we can formulate a general idea to solve non-linear classification problems with SVM:

Training stage

Given a training dataset $\mathcal{D} = \left\{ (\mathbf{x}_i, y_i) : | \mathbf{x}_i \in \mathbb{R}^d, y_i \in \{+1, -1\} \right\}_{i=1}^n$ which is not linearly separable

Determine a mapping function ϕ

For each $\mathbf{x}_i$, map it as $\phi(\mathbf{x}_i)$ into a high-dimensional feature space $\mathcal{H}$

In $\mathcal{H}$, train a linear SVM model $\mathcal{M}$ using $\left\{ \phi(\mathbf{x}_i), y_i \right\}_{i=1}^n$ as the training data

Testing stage

Given a test sample $\mathbf{t}$, which is expressed in the original low-dimensional space

Map $\mathbf{t}$ into $\mathcal{H}$ as $\phi(\mathbf{t})$

Classify $\phi(\mathbf{t})$ using $\mathcal{M}$

How to determine ϕ for a given dataset does not have a fixed correct answer, and it may depend on experience!

For nonlinear classification: Mapping from low dimensional space to a high dimensional one

Linear SVM:

$$\boldsymbol{\alpha}^* = \arg\min_{\boldsymbol{\alpha}} \left\{ \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n}\alpha_i\alpha_j y_i y_j \mathbf{x}_i\cdot\mathbf{x}_j - \sum_{i=1}^{n}\alpha_i \right\}$$

subject to $C \ge \alpha_i \ge 0, i = 1,\dots,n$

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

The classification decision function,

$$h_{\mathbf{w}^*,b^*}(\mathbf{x}) = \text{sign}\left(\sum_{i=1}^{n}\alpha_i^* y_i (\mathbf{x}\cdot\mathbf{x}_i) + b^*\right)$$

where

$$b^* = y_j - \sum_{i=1}^{n}\alpha_i^* y_i (\mathbf{x}_i\cdot\mathbf{x}_j)$$

When data is not linearly separable, mapping data into $\mathcal{H}$ by using ϕ

Linear SVM in high-dimensional space $\mathcal{H}$:

$$\boldsymbol{\alpha}^* = \arg\min_{\boldsymbol{\alpha}} \left\{ \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n}\alpha_i\alpha_j y_i y_j \phi(\mathbf{x}_i)\cdot\phi(\mathbf{x}_j) - \sum_{i=1}^{n}\alpha_i \right\}$$

subject to $C \ge \alpha_i \ge 0, i = 1,\dots,n$

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

The classification decision function,

$$h_{\mathbf{w}^*,b^*}(\mathbf{x}) = \text{sign}\left(\sum_{i=1}^{n}\alpha_i^* y_i \phi(\mathbf{x})\cdot\phi(\mathbf{x}_i) + b^*\right)$$

where

$$b^* = y_j - \sum_{i=1}^{n}\alpha_i^* y_i \left(\phi(\mathbf{x}_i)\cdot\phi(\mathbf{x}_j)\right)$$

① Using ϕ, mapping data points to $\mathcal{H}$
② Computing the dot product between two mapped points

Is there a function that can implicitly return the dot product of two points in the high-dimensional space?

That is the kernel function!

Kernel function

Definition 6: Kernel function. Suppose that χ is the original low-dimensional feature space and $\mathcal{H}$ is the high-dimensional feature space. If there exists a mapping,

$$\phi(\mathbf{x}) : \chi \to \mathcal{H}$$

$\forall \mathbf{x}, \mathbf{z} \in \chi$, the function $K(\cdot,\cdot) : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$ satisfies that,

$$K(\mathbf{x}, \mathbf{z}) = \phi(\mathbf{x}) \cdot \phi(\mathbf{z})$$

Then, the function $K(\cdot,\cdot)$ is called the **kernel function**

With the kernel function, we do not need to explicitly define the mapping function and the high-dimensional space!

Kernel function

- It needs to be noted that for a given kernel function K, the associated high-dimensional space $\mathcal{H}$ and the mapping function ϕ are not unique

Example.

Suppose the original feature space is $\mathbb{R}^2$. The kernel function is

$$K(\mathbf{x},\mathbf{z}) = (\mathbf{x}\cdot\mathbf{z})^2 = x_1^2 z_1^2 + 2x_1 x_2 z_1 z_2 + x_2^2 z_2^2$$

$\mathcal{H}$ can be $\mathbb{R}^3$ and ϕ can be $\phi(\mathbf{x}) = \left(x_1^2, \sqrt{2}x_1x_2, x_2^2\right)$

$\mathcal{H}$ can be $\mathbb{R}^3$ and ϕ can be $\phi(\mathbf{x}) = \frac{1}{\sqrt{2}}\left(x_1^2 - x_2^2, 2x_1x_2, x_1^2 + x_2^2\right)$

$\mathcal{H}$ can be $\mathbb{R}^4$ and ϕ can be $\phi(\mathbf{x}) = \left(x_1^2, x_1x_2, x_1x_2, x_2^2\right)$

Kernel function

- Commonly used kernel functions in the field of SVM

Polynomial kernel function:

$$K(\mathbf{x},\mathbf{z}) = (\mathbf{x}\cdot\mathbf{z}+1)^p$$

Gaussian kernel function (radial basis function, RBF):

$$K(\mathbf{x},\mathbf{z}) = \exp\left(-\frac{\|\mathbf{x}-\mathbf{z}\|_2^2}{2\sigma^2}\right)$$

Kernelized SVM

Linear SVM in high-dimensional space $\mathcal{H}$:

$$\boldsymbol{\alpha}^* = \arg\min_{\boldsymbol{\alpha}} \left\{ \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j \phi(\mathbf{x}_i) \cdot \phi(\mathbf{x}_j) - \sum_{i=1}^{n} \alpha_i \right\}$$

subject to $C \geq \alpha_i \geq 0, i = 1, \ldots, n$

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

The classification decision function,

$$h_{\mathbf{w}^*, b^*}(\mathbf{x}) = \text{sign}\left(\sum_{i=1}^{n} \alpha_i^* y_i \phi(\mathbf{x}) \cdot \phi(\mathbf{x}_i) + b^* \right)$$

where

$$b^* = y_j - \sum_{i=1}^{n} \alpha_i^* y_i \left(\phi(\mathbf{x}_i) \cdot \phi(\mathbf{x}_j) \right)$$

with the kernel function K $\Longrightarrow$

Kernelized SVM:

$$\boldsymbol{\alpha}^* = \arg\min_{\boldsymbol{\alpha}} \left\{ \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j K(\mathbf{x}_i, \mathbf{x}_j) - \sum_{i=1}^{n} \alpha_i \right\}$$

subject to $C \geq \alpha_i \geq 0, i = 1, \ldots, n$

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

The classification decision function,

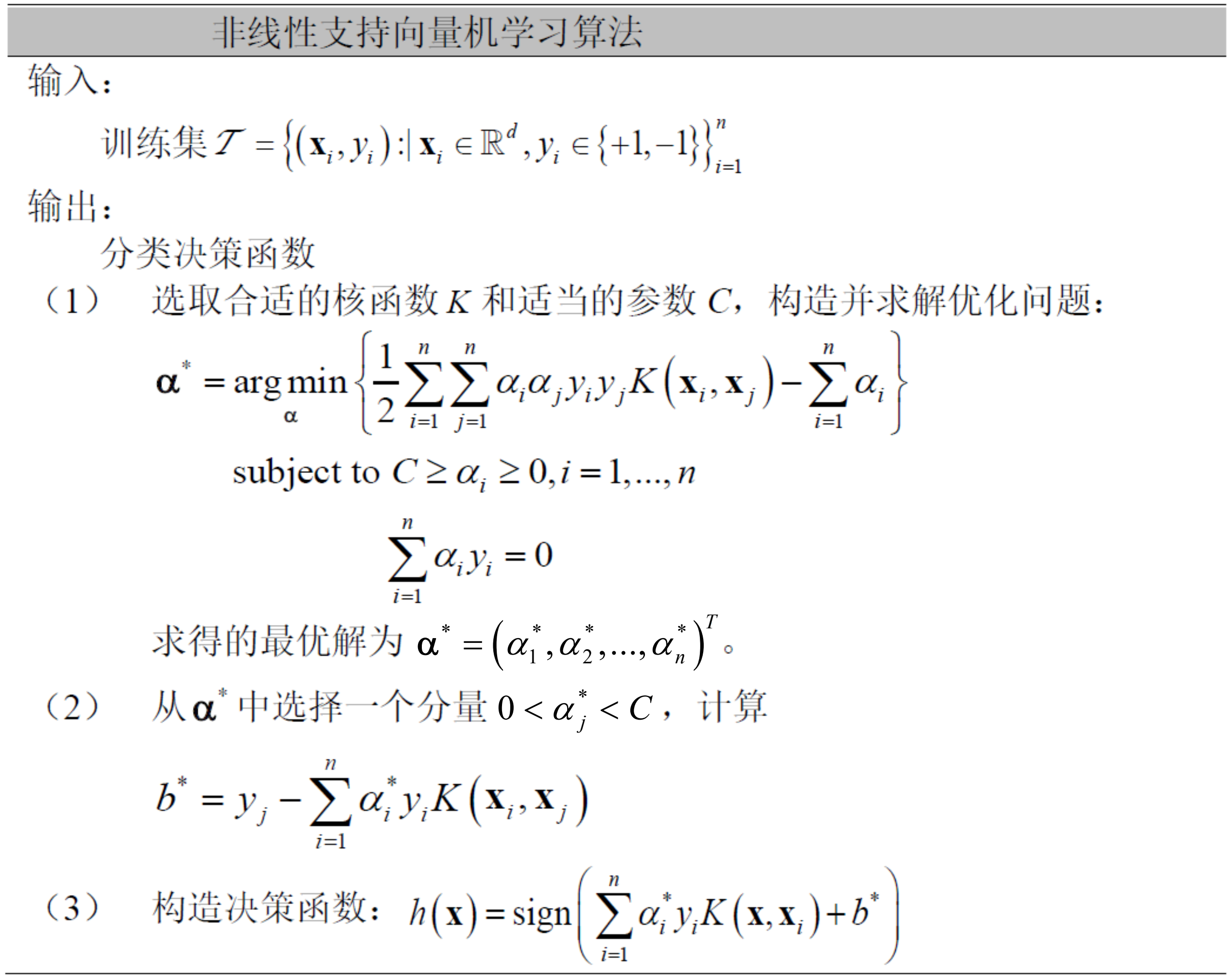
$$h_{\mathbf{w}^*, b^*}(\mathbf{x}) = \text{sign}\left(\sum_{i=1}^{n} \alpha_i^* y_i K(\mathbf{x}, \mathbf{x}_i) + b^* \right)$$

where

$$b^* = y_j - \sum_{i=1}^{n} \alpha_i^* y_i K(\mathbf{x}_i, \mathbf{x}_j)$$

When the mapping function $\phi : \chi \to \mathcal{H}$ implicitly represented by the kernel function $K(\cdot,\cdot) : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$ is nonlinear, the kernelized SVM is a nonlinear model and is called **nonlinear SVM**

Nonlinear SVM



非线性支持向量机学习算法

输入：

训练集 $\mathcal{T} = \{(\mathbf{x}_i, y_i) : | \mathbf{x}_i \in \mathbb{R}^d, y_i \in \{+1, -1\}\}_{i=1}^{n}$

输出：

分类决策函数

（1） 选取合适的核函数 K 和适当的参数 C，构造并求解优化问题：

$$\boldsymbol{\alpha}^* = \arg\min_{\boldsymbol{\alpha}} \left\{ \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j K(\mathbf{x}_i, \mathbf{x}_j) - \sum_{i=1}^{n} \alpha_i \right\}$$

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

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

求得的最优解为 $\boldsymbol{\alpha}^* = (\alpha_1^*, \alpha_2^*, \ldots, \alpha_n^*)^T$。

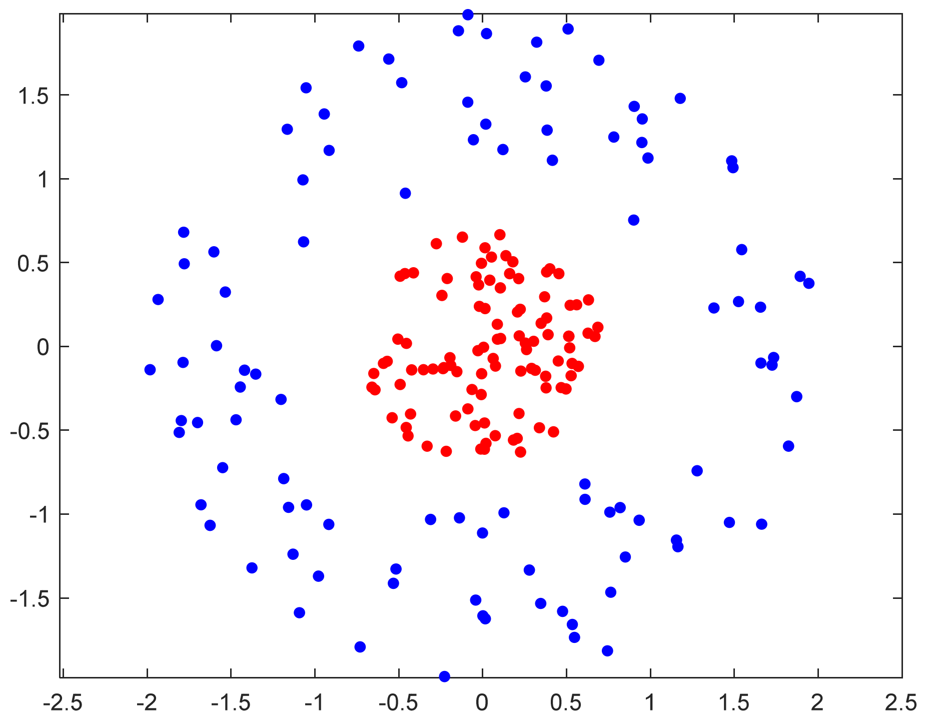
（2） 从 $\boldsymbol{\alpha}^*$ 中选择一个分量 $0 < \alpha_j^* < C$，计算

$$b^* = y_j - \sum_{i=1}^{n} \alpha_i^* y_i K(\mathbf{x}_i, \mathbf{x}_j)$$

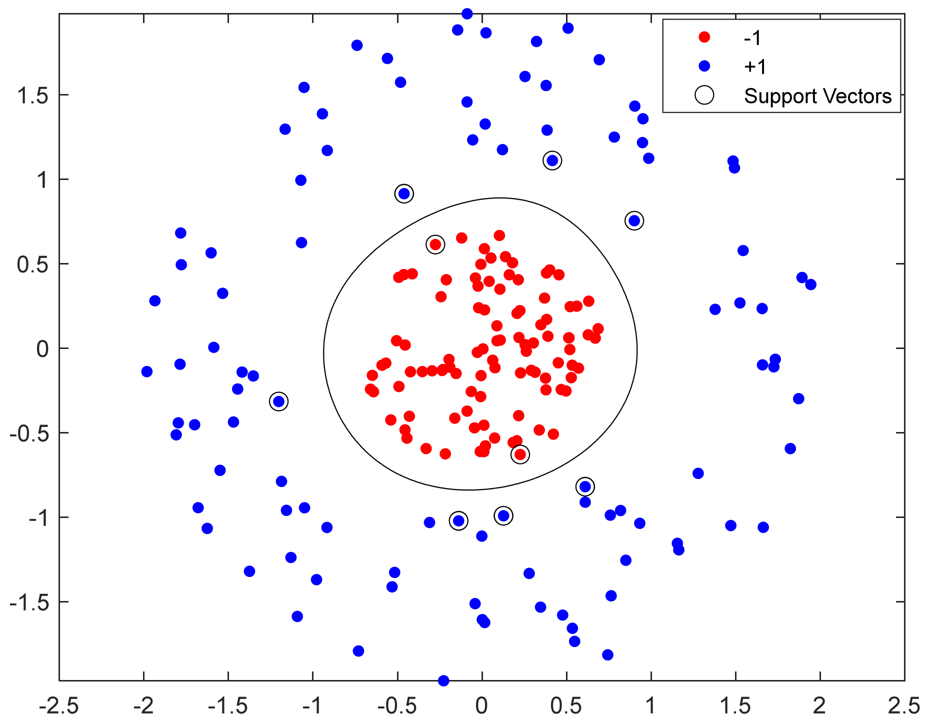
（3） 构造决策函数： $h(\mathbf{x}) = \text{sign}\left(\sum_{i=1}^{n} \alpha_i^* y_i K(\mathbf{x}, \mathbf{x}_i) + b^* \right)$

Nonlinear SVM

Example:



A nonlinear classification problem



The curve is the decision boundary obtained by using nonlinear SVM

Outline

- Linear separability and hyperplanes
- The perceptron learning algorithm
- Hard-margin SVM
- Soft-margin SVM
- Kernelized SVM
- Multi-class classification with SVM
- Object detection using SVM+HOG
- Practice

From binary classification to multi-class classification

- The previously mentioned SVM models are all designed for binary classification
- With proper extensions, binary classification models (not limited to SVM) can be adapted to solve the multi-class classification problems
- Two commonly used strategies to extend a binary classification model to a multi-class classification model, the **one-versus-all** approach and the **one-versus-one** approach

One-versus-all

Suppose that we are solving a *K*-class classification task

Training stage

We need to train *K* binary classifiers $h_1, h_2, ..., h_K$

When training h_k (represented by the hyperplane ($\mathbf{w}_k$, b_k)), taking training samples belonging to *k*th-class as the positive training samples, and all the other ones as negative training samples

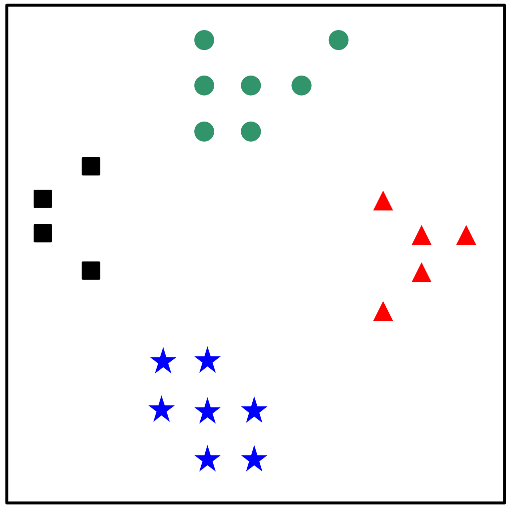
Testing stage

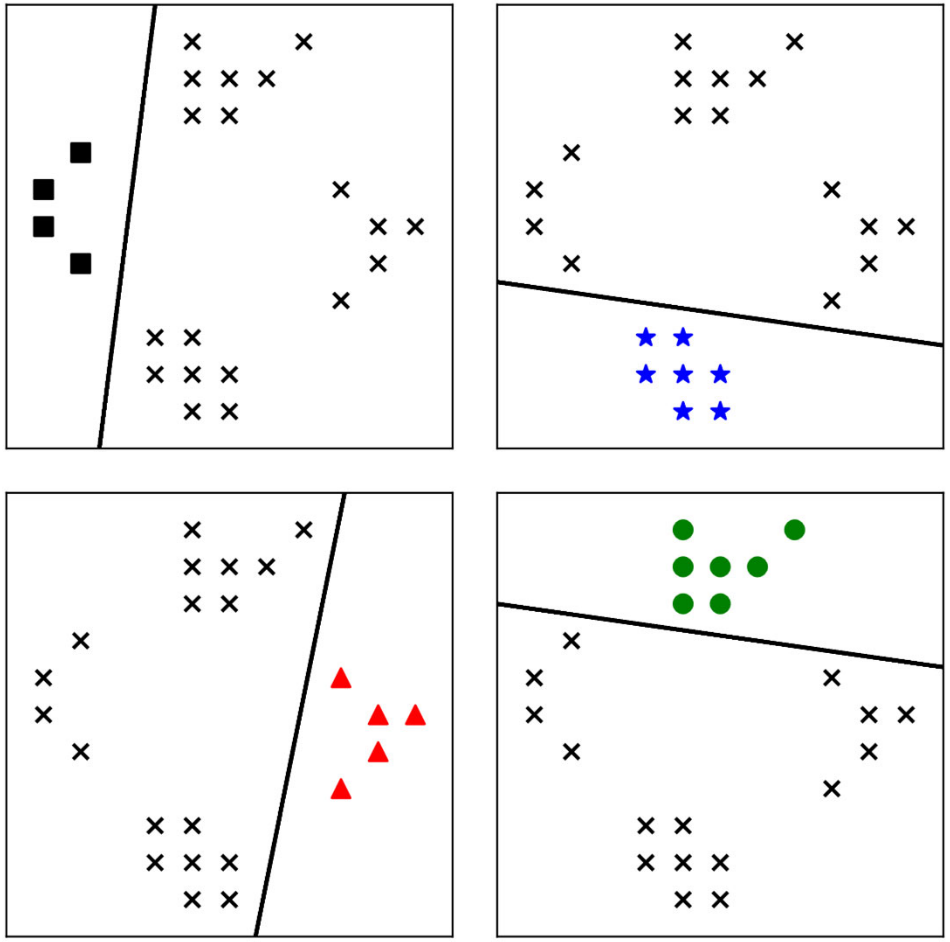
Given a test sample **t**

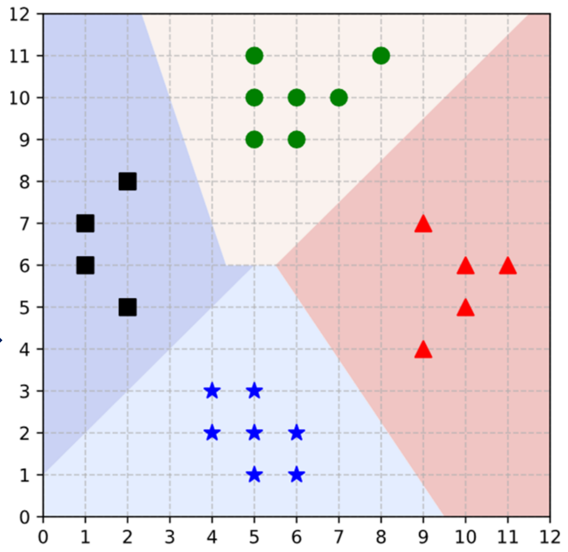
Compute **t**'s classification response with respect to all the *K* classifiers $\left\{\mathbf{w}_i^T\mathbf{t}+b_i\right\}_{i=1}^K$

t's label = $\arg\max_j \mathbf{w}_j^T\mathbf{t}+b_j$

One-versus-all: A toy example


A 4-class classification problem


train 4 binary classifiers



Decision boundaries for the 4 classes

One-versus-one

Suppose that we are solving a *K*-class classification task

Training stage

For each pair of two classes, *i* and *j*, $i \neq j$, train a binary classifier h_{ij}

Altogether, we can have $K(K-1)/2$ binary classifiers $\{h_{ij}\}_{i,j=1,\ldots,K,i\neq j}$

Testing stage

Given a test sample **t**

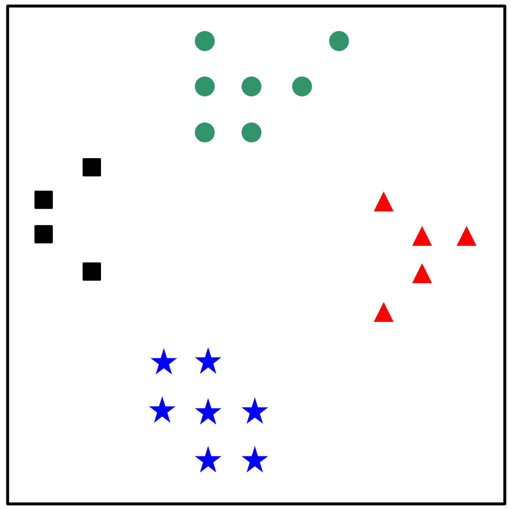
Classify **t** using all the $K(K-1)/2$ binary classifiers $\{h_{ij}\}_{i,j=1,\ldots,K,i\neq j}$ and we can get $K(K-1)/2$ results,

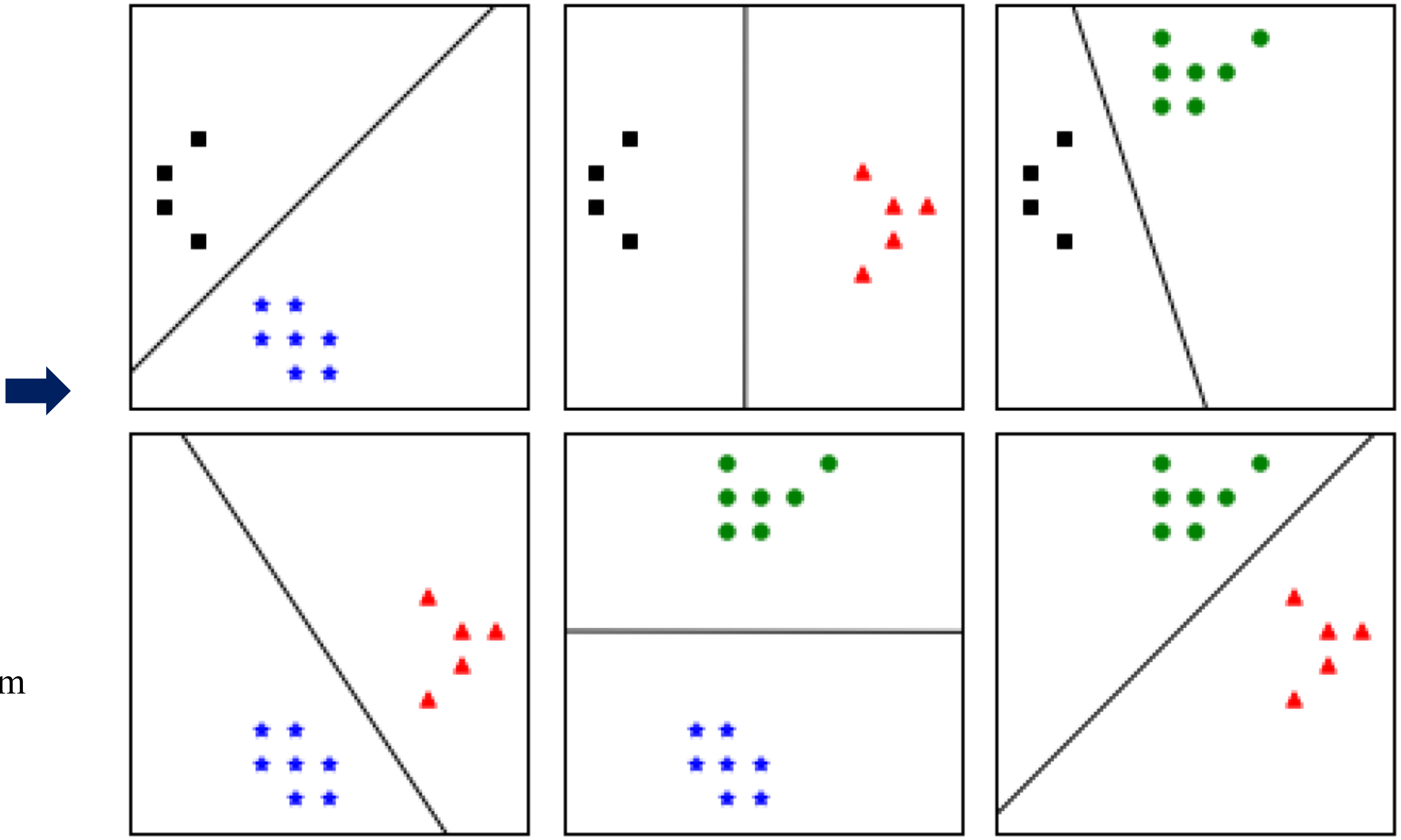
$$\mathcal{C} = \{r_1, r_2, \ldots, r_{K(K-1)/2}\}, r_i \in \{1, 2, \ldots, K\}$$

Using the "voting" strategy to get the final label of **t**,

t's label = The element with the most occurrences in $\mathcal{C}$

One-versus-one: A toy example


A 4-class classification problem


train 6 binary classifiers

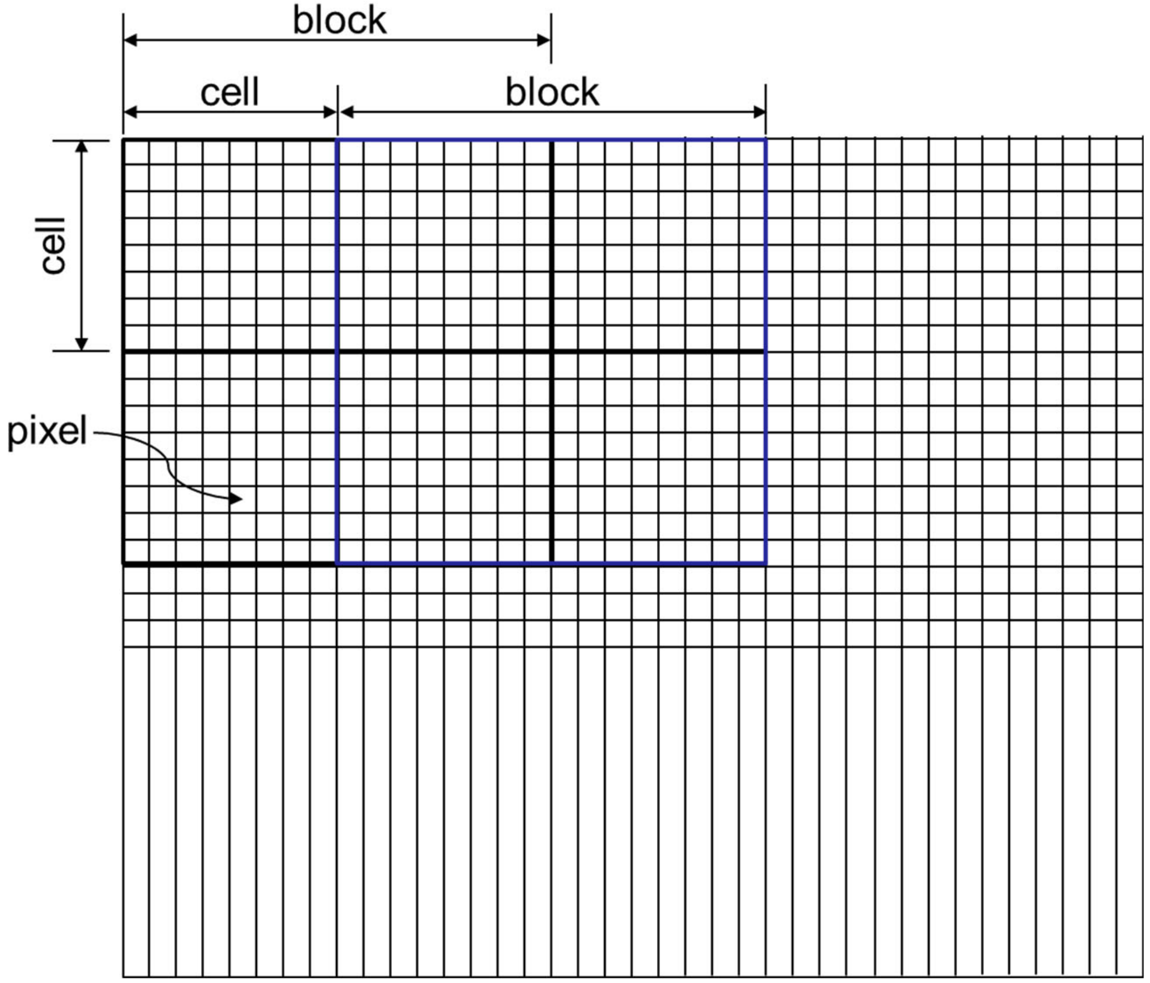
Outline

- Linear separability and hyperplanes
- The perceptron learning algorithm
- Hard-margin SVM
- Soft-margin SVM
- Kernelized SVM
- Multi-class classification with SVM
- Object detection using SVM+HOG
- Practice

Histogram of Oriented Gradients (HOG)

- Histogram of Oriented Gradients (HOG) is a feature descriptor widely used in computer vision for object detection and recognition tasks; It captures the distribution of gradient orientations in an image
  - Gradient Computation: The first step is to compute the gradients of the image, typically using filters like Sobel or Prewitt
  - Cell Division: Image is divided into small spatial regions called cells (e.g., 8×8 pixels)
  - Orientation Binning: Within each cell, gradient orientations are quantized into a fixed number of bins (e.g., 9 bins covering 0° to 180°). The gradient magnitudes are accumulated into these bins to form a histogram, representing the dominant edge directions in the cell
  - Block Normalization: Cells are grouped into larger regions called blocks (e.g., 2×2 cells). To improve robustness to lighting variations, the histograms within a block are normalized using methods like l_2-Norm or l_1-Norm
  - Feature Vector Formation: The normalized histograms from all blocks are concatenated into a single feature vector, which represents the HOG descriptor for the image or region

Histogram of Oriented Gradients (HOG)



Object detection using SVM + HOG

Suppose we want to train a detector for three classes of objects, cats, dogs, and bicycles

Training Phase

1) Collect image samples of three categories (cat, dog, bicycle) and a set of negative sample images that do not contain these three types of targets
2) Normalize the size of all sample images to $w \times h$; this size will be the size of the sliding window used in sliding window detection during the testing phase
3) Extract the HOG feature vector representing each sample from every sample image
4) Using the set of HOG feature vectors of all image samples as the training sample set, train a 4-category (cat, dog, bicycle, non-interest target) SVM classifier $\mathcal{M}$. During the testing phase, we will use $\mathcal{M}$ to classify the image content within the sliding window

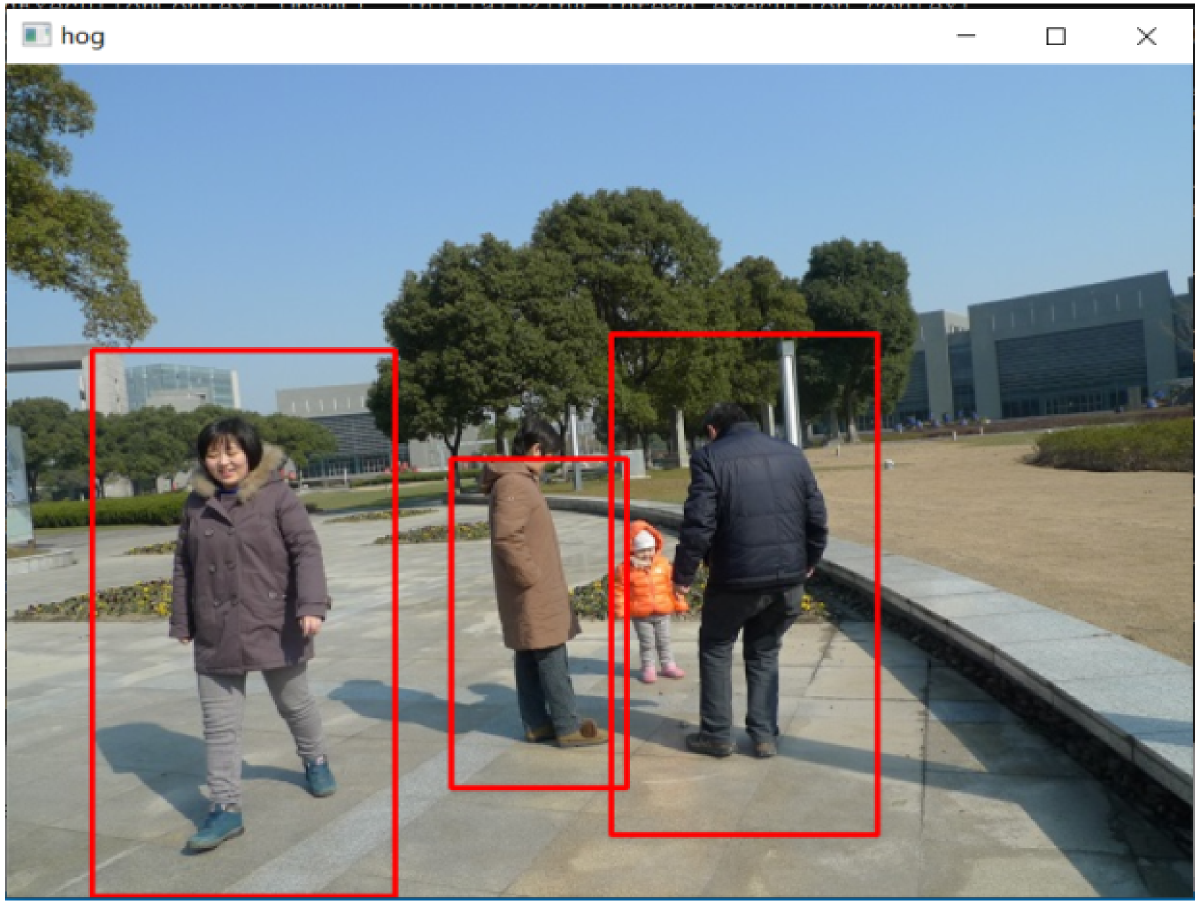
Object detection using SVM + HOG

Suppose we want to train a detector for three classes of objects, cats, dogs, and bicycles

Testing Phase (Given an input image ***I***, detect objects in ***I***)

1) To detect targets of different scales, it is necessary to construct an image pyramid for ***I***. Let *scale_factor* be the scale factor of the pyramid (e.g., 1.2). Then, the scale parameter of the lth layer in the pyramid is $s_l = scale\_factor^{l-1}$ (where l=1, 2, …, L), where L is the total number of layers; The resolution of the lth layer is $1/s_l$ of the resolution of ***I***
2) On the lth layer, take a sliding window of size $w\times h$. At the current sliding window position, extract the HOG feature vector of the image covered by the window and feed it into $\mathcal{M}$ for classification. If the classification result of the current window belongs to one of the interest targets (cat, dog, or bicycle), record its position, category, and classification confidence
3) After completing the above detection tasks on all pyramid layers, convert all detection results (positions and sizes) back to the original resolution of ***I***. For example, suppose a target is detected at position (x_o, y_o) on layer l=3. The size of this target under the original resolution should be $[w\times scale\_factor^2, h\times scale\_factor^2]$, and its position should be $(x_o\times scale\_factor^2, y_o\times scale\_factor^2)$
4) Finally, to eliminate overlapping detection boxes, a non-maximum suppression strategy is used to retain only the detection result with the highest classification confidence within a local region

Object detection using SVM + HOG

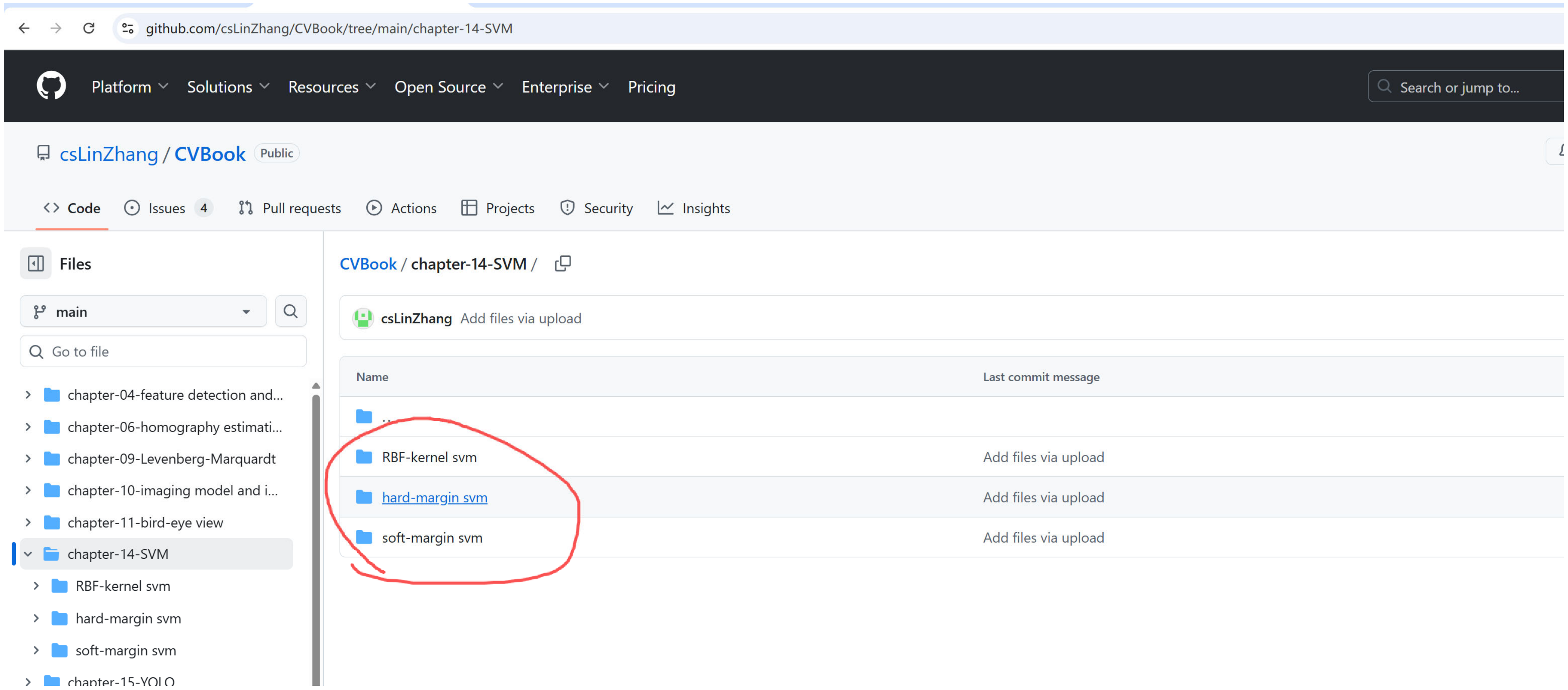


SVM + HOG pedestrian detection result using OpenCV’s implementation

Outline

- Linear separability and hyperplanes
- The perceptron learning algorithm
- Hard-margin SVM
- Soft-margin SVM
- Kernelized SVM
- Multi-class classification with SVM
- Object detection using SVM+HOG
- Practice

Practice


github.com/csLinZhang/CVBook/tree/main/chapter-14-SVM

chapter-04-feature detection and...
chapter-06-homography estimati...
chapter-09-Levenberg-Marquardt
chapter-10-imaging model and i...
chapter-11-bird-eye view
chapter-14-SVM
RBF-kernel svm
hard-margin svm
soft-margin svm
chapter-15-YOLO

CVBook / chapter-14-SVM /

csLinZhang Add files via upload

Name	Last commit message
..	
RBF-kernel svm	Add files via upload
hard-margin svm	Add files via upload
soft-margin svm	Add files via upload

Q+A