Support Vector Machines
支持向量机

Supervised learning

Supervised learning

- An agent or machine is given N sensory inputs $D = \{x_1, x_2 \ldots, x_N\}$, as well as the desired outputs $y_1, y_2, \ldots y_N$, its goal is to learn to produce the correct output given a new input.
- Given D what can we say about x_{N+1}?

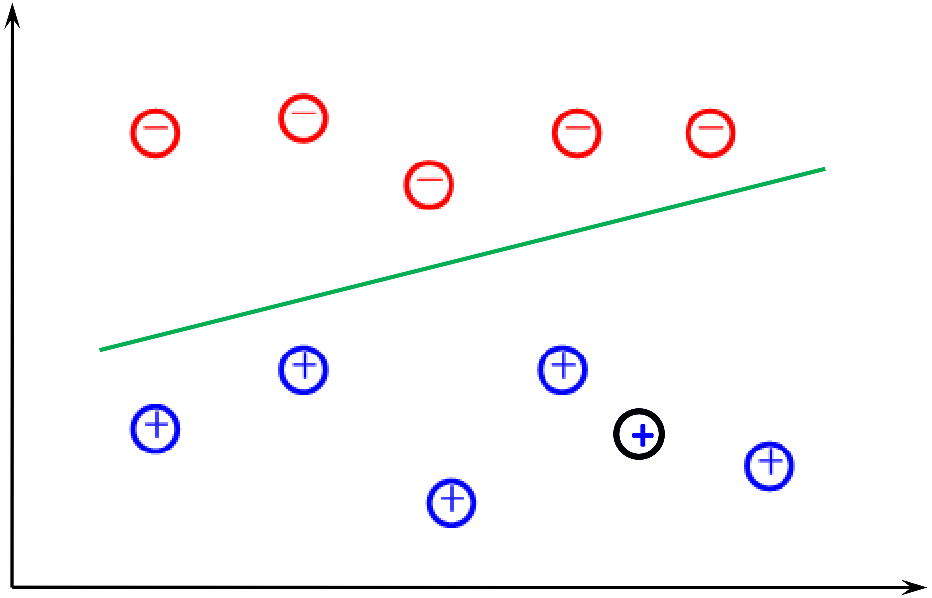
Classification: $y_1, y_2, \ldots y_N$ are discrete class labels, learn a labeling function $f(\mathbf{x}) \mapsto y$

- Naïve bayes
- Decision tree
- K nearest neighbor
- Least squares classification

Classification

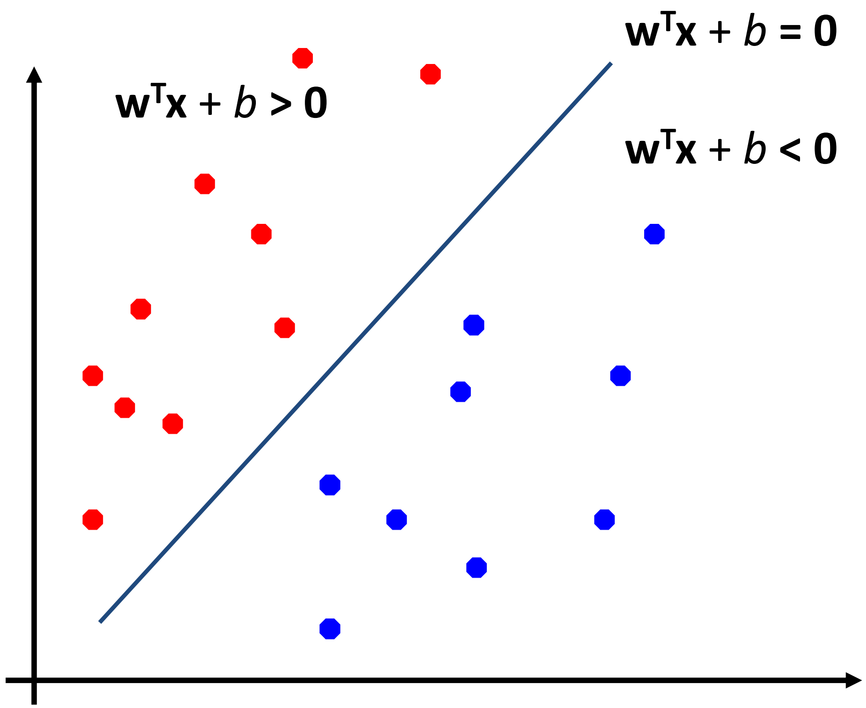
Classification

= learning from labeled data. Dominant problem in Machine Learning



Linear Classifiers

Binary classification can be viewed as the task of separating classes in feature space（特征空间）：

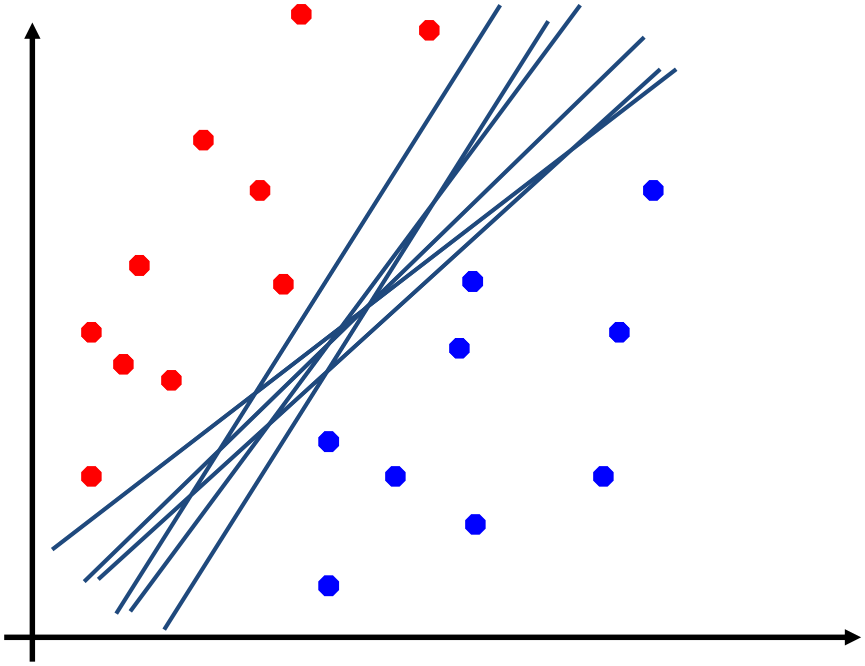


Decide $\hat{y} = 1$ if $\mathbf{w}^T\mathbf{x} + b > 0$,
otherwise $\hat{y} = -1$

$\hat{y} = h(\mathbf{x}) = \text{sign}(\mathbf{w}^T\mathbf{x} + b)$

Linear Classification

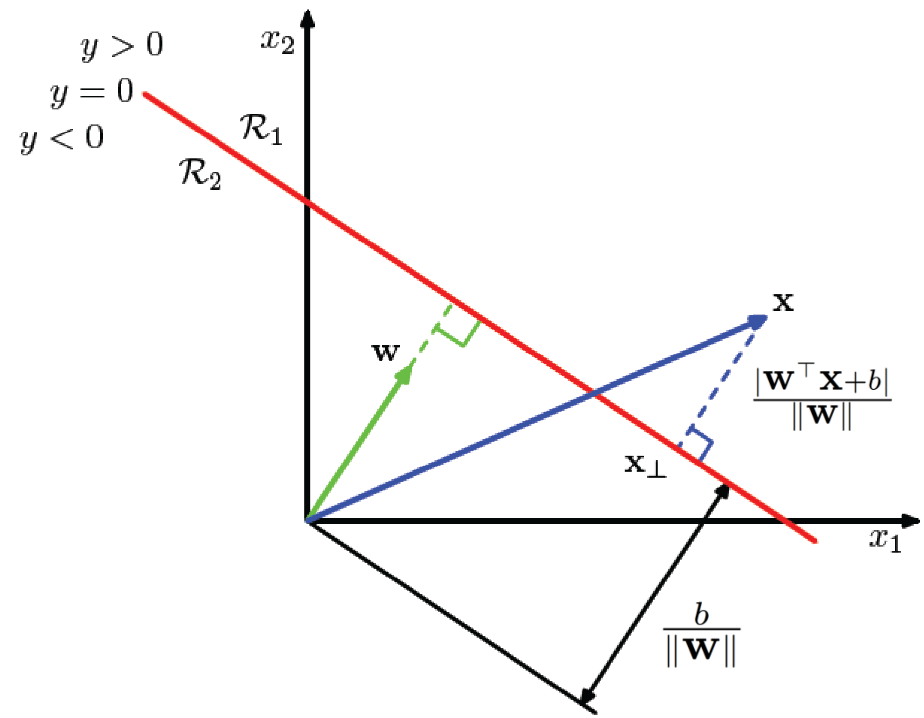
Which of the linear separators is optimal?



Classification Margin（间距）

- Geometry of linear classification
- Discriminant function

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



- Important: the distance does not change if we scale

$$\mathbf{w} \to a\mathbf{w}, b \to a \cdot b$$

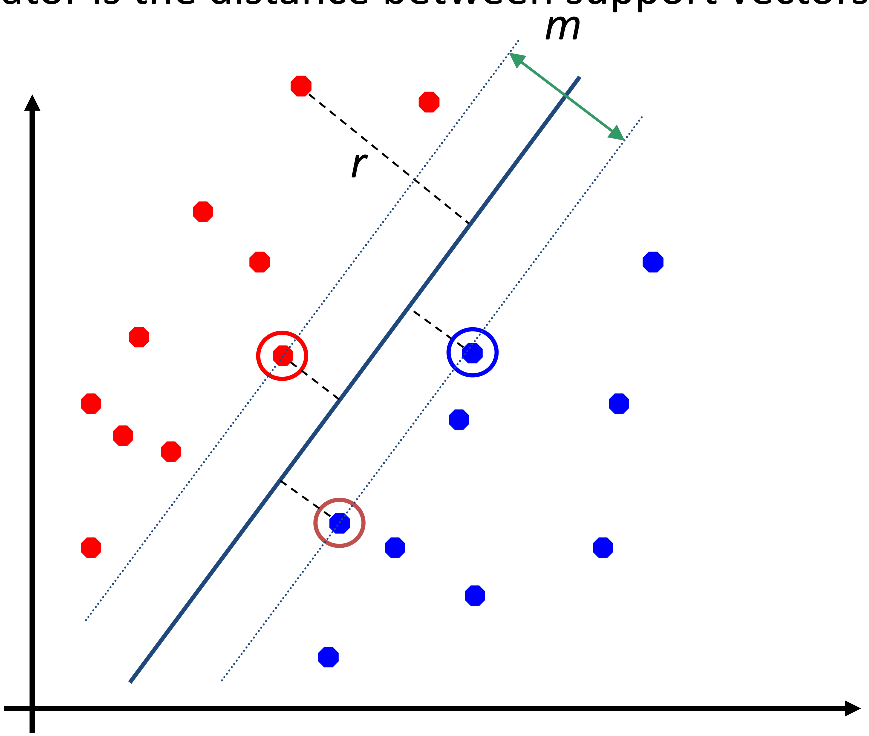
Classification Margin（间距）

Distance from example $\mathbf{x}_i$ to the separator is $r = \frac{|\mathbf{w}^T \mathbf{x}_i + b|}{\|\mathbf{w}\|}$

Define the **margin** of a linear classifier as the width that the boundary could be increased by before hitting a data point

Examples closest to the hyperplane（超平面） are ***support vectors*** （支持向量）.

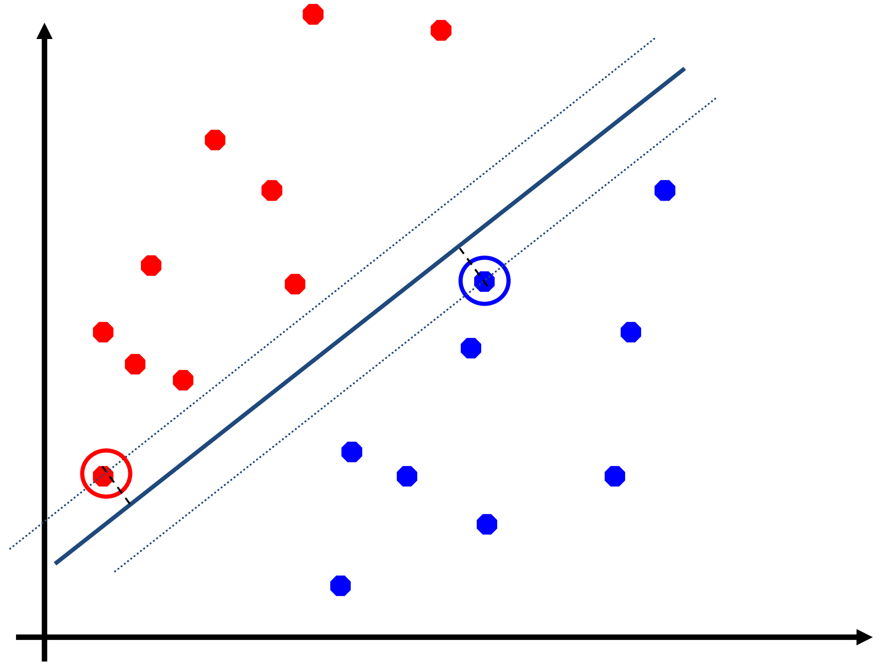
Margin *m* of the separator is the distance between support vectors.



Maximum Margin Classification
最大间距分类

Maximizing the margin is good according to intuition and PAC theory.

Implies that only support vectors matter; other training examples are ignorable.

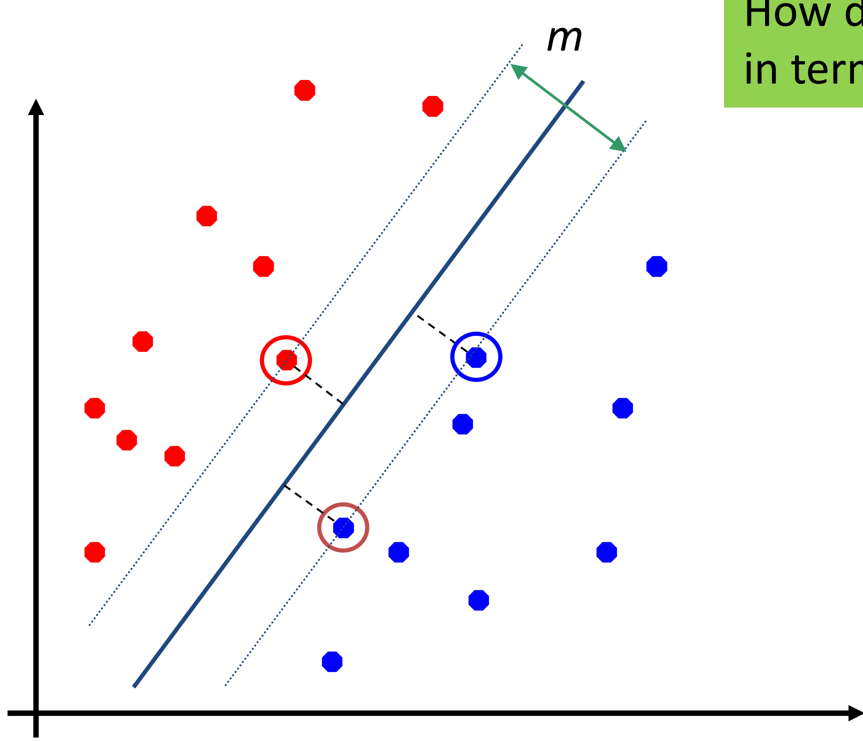


Maximum Margin Classification
最大间距分类

Maximizing the margin is good according to intuition and PAC theory.

Implies that only support vectors matter; other training examples are ignorable.

How do we compute *m* in term of ***w*** and *b*?



Maximum Margin Classification Mathematically

Let training set $\{(\mathbf{x}_i, y_i)\}_{i=1..N}$, $\mathbf{x}_i \in \mathbf{R}^d$, $y_i \in \{-1, 1\}$ be separated by a hyperplane with margin m. Then for each training example $(\mathbf{x}_i, y_i)$:

$$\begin{aligned} \mathbf{w}^T\mathbf{x}_i + b &\leq -c \quad \text{if } y_i = -1 \\ \mathbf{w}^T\mathbf{x}_i + b &\geq c \quad \text{if } y_i = 1 \end{aligned} \quad \Leftrightarrow \quad y_i(\mathbf{w}^T\mathbf{x}_i + b) \geq c$$

For every support vector $\mathbf{x}_s$

the above inequality is an equality.

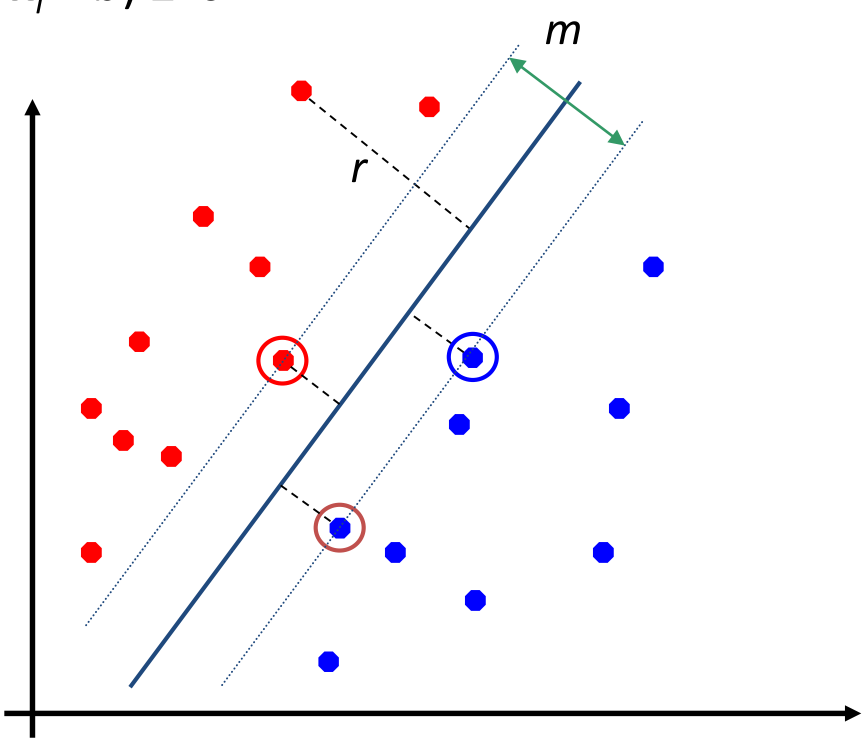
$y_s(\mathbf{w}^T\mathbf{x}_s + b) = c$

In the equality, we obtain that

distance between each $\mathbf{x}_s$ and

the hyperplane is

$$r = \frac{|\mathbf{w}^T\mathbf{x}_s + b|}{\|\mathbf{w}\|} = \frac{y_s(\mathbf{w}^T\mathbf{x}_s + b)}{\|\mathbf{w}\|} = \frac{c}{\|\mathbf{w}\|}$$



Maximum Margin Classification Mathematically

Then the margin can be expressed through **w** and b:

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

Here is our Maximum Margin Classification problem:

$$\max_{\mathbf{w},b} \frac{2c}{\|\mathbf{w}\|} \quad \text{subject to} \quad y_i(\mathbf{w}^\top \mathbf{x}_i + b) \geq c, \forall i$$

$$\max_{\mathbf{w},b} \frac{c}{\|\mathbf{w}\|} \quad \text{subject to} \quad y_i(\mathbf{w}^\top \mathbf{x}_i + b) \geq c, \forall i$$

Note that the magnitude（大小） of *c* merely scales **w** and b, and does not change the classification boundary at all!

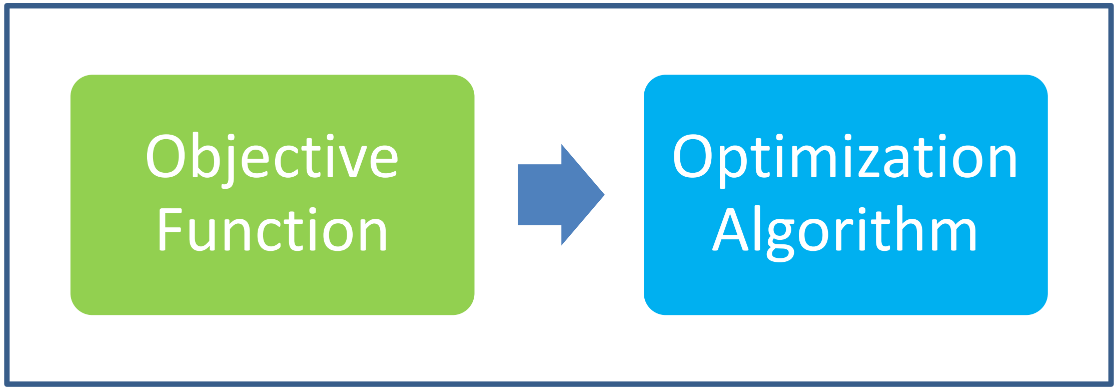
So we have a cleaner problem:

$$\max_{\mathbf{w},b} \frac{1}{\|\mathbf{w}\|} \quad \text{subject to} \quad y_i(\mathbf{w}^\top \mathbf{x}_i + b) \geq 1, \forall i$$

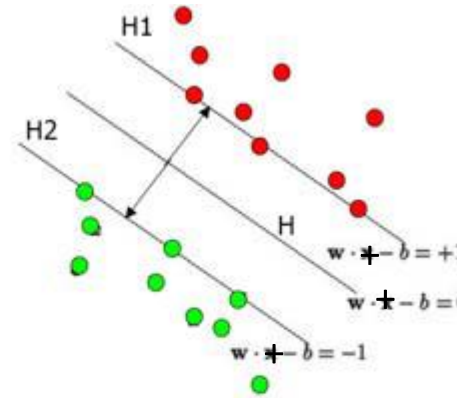
This leads to the famous Support Vector Machines 支持向量机 — believed by many to be the best "off-the-shelf" supervised learning algorithm

Learning as Optimization

Parameter Learning



Support Vector Machine



- A convex quadratic programming（凸二次规划）problem with linear constraints:

$$\max_{\mathbf{w},b} \frac{1}{\|\mathbf{w}\|} \quad \text{subject to} \quad y_i(\mathbf{w}^\top \mathbf{x}_i + b) \geq 1, \forall i$$

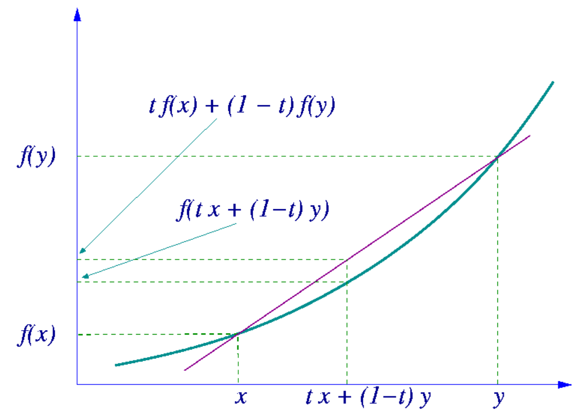
  - The attained margin is now given by $\frac{1}{\|\mathbf{w}\|}$
  - Only a few of the classification constraints are relevant → **support vectors**

- Constrained optimization（约束优化）
  - We can directly solve this using commercial quadratic programming (QP) code
  - But we want to take a more careful investigation of Lagrange duality（对偶性）, and the solution of the above in its dual form.
  - deeper insight: support vectors, kernels（核） …

Quadratic Programming

Minimize (with respect to x)

$$g(\mathbf{x}) = \frac{1}{2}\mathbf{x}^\top Q\mathbf{x} + \mathbf{c}^\top \mathbf{x}$$



Subject to one or more constraints of the form:

$$A\mathbf{x} \leq \mathbf{b} \qquad \text{(inequality constraint)}$$

$$E\mathbf{x} = \mathbf{d} \qquad \text{(equality constraint)}$$

If $Q \succeq 0$, then $g(x)$ is a convex function（凸函数）: In this case the quadratic program has a global minimizer

Quadratic program of support vector machine:

$$\min_{\mathbf{w},b} \mathbf{w}^\top \mathbf{w} \quad \text{subject to} \quad y_i(\mathbf{w}^\top \mathbf{x}_i + b) \geq 1, \forall i$$

Solving Maximum Margin Classifier

Our optimization problem:

$$\min_{\mathbf{w},b} \mathbf{w}^\top \mathbf{w} \quad \text{subject to} \quad 1 - y_i(\mathbf{w}^\top \mathbf{x}_i + b) \leq 0, \forall i \qquad (1)$$

The Lagrangian:

$$\begin{aligned} L(\mathbf{w}, b, \alpha) &= \frac{1}{2}\mathbf{w}^\top \mathbf{w} - \sum_{i=1}^{n} \alpha_i \left[y_i(\mathbf{w}^\top \mathbf{x}_i + b) - 1 \right] \\ &= \frac{1}{2}\mathbf{w}^\top \mathbf{w} + \sum_{i=1}^{n} \alpha_i \left[1 - y_i(\mathbf{w}^\top \mathbf{x}_i + b) \right] \end{aligned}$$

Consider each constraint:

$$\begin{aligned} \max_{\alpha_i \geq 0} \alpha_i \left[1 - y_i(\mathbf{w}^\top \mathbf{x}_i + b) \right] &= 0 \qquad \text{if } \mathbf{w}, \text{ b satisfies primal constraints} \\ &= \infty \qquad \text{otherwise} \end{aligned}$$

Solving Maximum Margin Classifier

Our optimization problem:

$$\min_{\mathbf{w},b} \mathbf{w}^\top \mathbf{w} \quad \text{subject to} \quad 1 - y_i(\mathbf{w}^\top \mathbf{x}_i + b) \leq 0, \forall i \tag{1}$$

The Lagrangian:

$$L(\mathbf{w}, b, \alpha) = \frac{1}{2}\mathbf{w}^\top \mathbf{w} - \sum_{i=1}^{n} \alpha_i \left[y_i(\mathbf{w}^\top \mathbf{x}_i + b) - 1 \right]$$

Lemma:

$$\begin{aligned} \max_{\alpha \geq 0} L(\mathbf{w}, b, \alpha) = & \quad \tfrac{1}{2}\mathbf{w}^\top \mathbf{w} \qquad \text{if } \mathbf{w}, \text{ b satisfies primal constraints} \\ = & \quad \infty \qquad\qquad \text{otherwise} \end{aligned}$$

(1) can be reformulated as $\min_{\mathbf{w},b} \max_{\alpha \geq 0} L(\mathbf{w}, b, \alpha)$

The dual problem（对偶问题）: $\max_{\alpha \geq 0} \min_{\mathbf{w},b} L(\mathbf{w}, b, \alpha)$

The Dual Problem（对偶问题）

$$\max_{\alpha \geq 0} \min_{\mathbf{w}, b} L(\mathbf{w}, b, \alpha)$$

We minimize L with respect to $\mathbf{w}$ and b first:

$$\frac{\partial L}{\partial \mathbf{w}} L(\mathbf{w}, b, \alpha) = \mathbf{w}^\top - \sum_{i=1}^{n} \alpha_i y_i \mathbf{x}_i^\top = 0 \qquad (2)$$

$$\frac{\partial L}{\partial \mathbf{b}} L(\mathbf{w}, b, \alpha) = -\sum_{i=1}^{n} \alpha_i y_i = 0 \qquad (3)$$

Note: $d(\mathbf{Ax}+\mathbf{b})^T(\mathbf{Ax}+\mathbf{b}) = (2(\mathbf{Ax}+\mathbf{b})^T\mathbf{A})\, d\mathbf{x}$

$d(\mathbf{x}^T\mathbf{a}) = d(\mathbf{a}^T\mathbf{x}) = \mathbf{a}^T\, d\mathbf{x}$

Note that the bias term b dropped out but had produced a "global" constraint on α

The Dual Problem（对偶问题）

$$\max_{\alpha \geq 0} \min_{\mathbf{w}, b} L(\mathbf{w}, b, \alpha)$$

We minimize L with respect to $\mathbf{w}$ and b first:

$$\frac{\partial L}{\partial \mathbf{w}} L(\mathbf{w}, b, \alpha) = \mathbf{w}^\top - \sum_{i=1}^{n} \alpha_i y_i \mathbf{x}_i^\top = 0 \tag{2}$$

$$\frac{\partial L}{\partial \mathbf{b}} L(\mathbf{w}, b, \alpha) = - \sum_{i=1}^{n} \alpha_i y_i = 0 \tag{3}$$

Note that (2) implies

$$\mathbf{w} = \sum_{i=1}^{n} \alpha_i y_i \mathbf{x}_i \tag{4}$$

Plug (4) back to L, and using (3), we have

$$L(\mathbf{w}, b, \alpha) = \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{n} \alpha_i \alpha_j y_i y_j (\mathbf{x}_i^\top \mathbf{x}_j)$$

The Dual Problem（对偶问题）

Now we have the following dual optimization problem:

$$\max_{\alpha} \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{n} \alpha_i \alpha_j y_i y_j (\mathbf{x}_i^\top \mathbf{x}_j) \quad \text{subject to} \quad \alpha_i \geq 0, \forall i$$

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

This is a quadratic programming problem again

– A global maximum can always be found

But what's the big deal?

1. w can be recovered by $\mathbf{w} = \sum_{i=1}^{n} \alpha_i y_i \mathbf{x}_i$
2. b can be recovered by $b = y_i - \mathbf{w}^\top \mathbf{x}_i$ for any i that $\alpha_i \neq 0$
3. The "kernel"—核 $\mathbf{x}_i^\top \mathbf{x}_j$ more later…

Support Vectors

If a point x_i satisfies $y_i(\mathbf{w}^\top \mathbf{x}_i + b) > 1$

Due to the fact that

$$\max_{\alpha_i \geq 0} \alpha_i \left[1 - y_i(\mathbf{w}^\top \mathbf{x}_i + b)\right] = 0 \quad y_i(\mathbf{w}^\top \mathbf{x}_i + b) \geq 1$$
$$= \infty \quad \text{otherwise}$$

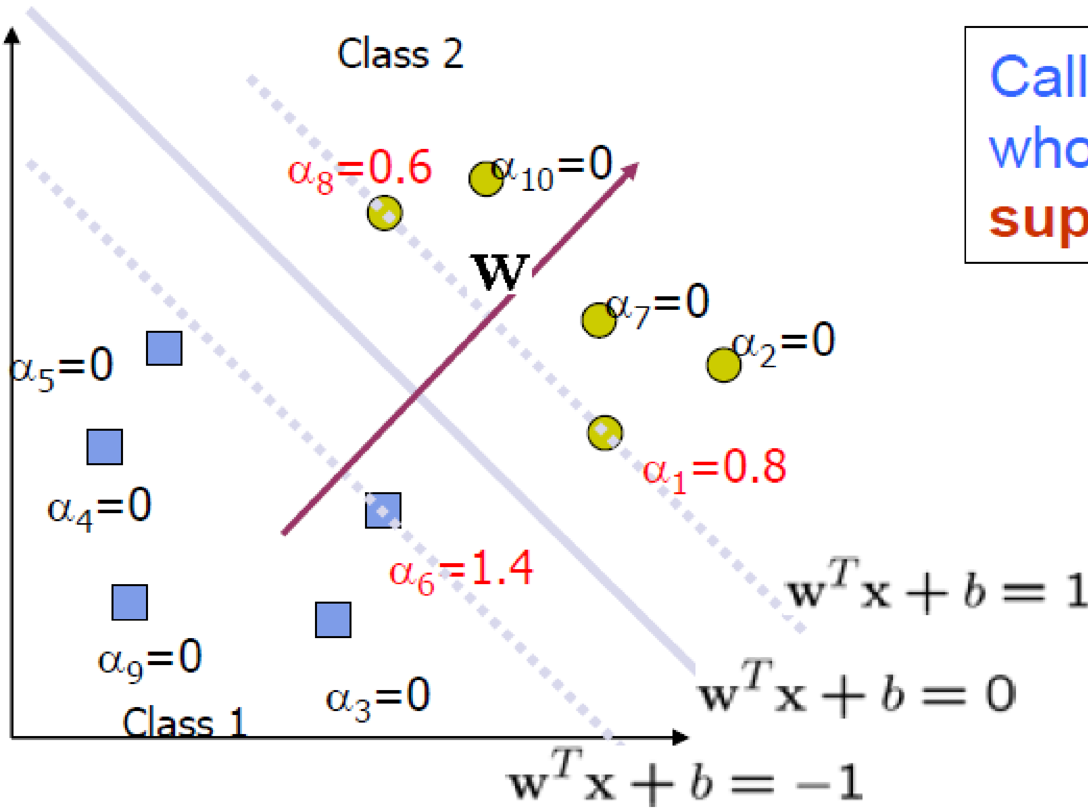
We have $\alpha^* = 0; \ \mathbf{x}_i \text{ not a support vector}$

w is decided by the points with non-zero α's

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

Support Vectors

only a few α_i's can be nonzero!!



Call the training data points whose α_i's are nonzero the **support vectors** (SV)

Support Vector Machines

Once we have the Lagrange multipliers α_i, we can reconstruct the parameter vector $\mathbf{w}$ as a weighted combination of the training examples:

$$\mathbf{w} = \sum_{i=1}^{n} \alpha_i y_i \mathbf{x}_i = \sum_{i \in SV} \alpha_i y_i \mathbf{x}_i$$

For testing with a new data $\mathbf{x}'$

Compute $\mathbf{w}^\top \mathbf{x}' + b = \sum_{i \in SV} \alpha_i y_i (\mathbf{x}_i^\top \mathbf{x}') + b$

and classify $\mathbf{x}'$ as class 1 if the sum is positive, and class 2 otherwise

Note: $\mathbf{w}$ *need not be formed explicitly*

Interpretation（解释） of support vector machines

- The optimal $\mathbf{w}$ is a linear combination of a small number of data points. This “sparse稀疏” representation can be viewed as data compression（数据压缩） as in the construction of kNN classifier

- To compute the weights α_i, and to use support vector machines we need to specify only the inner products内积 (or kernel) between the examples $\mathbf{x}_i^\top \mathbf{x}_j$

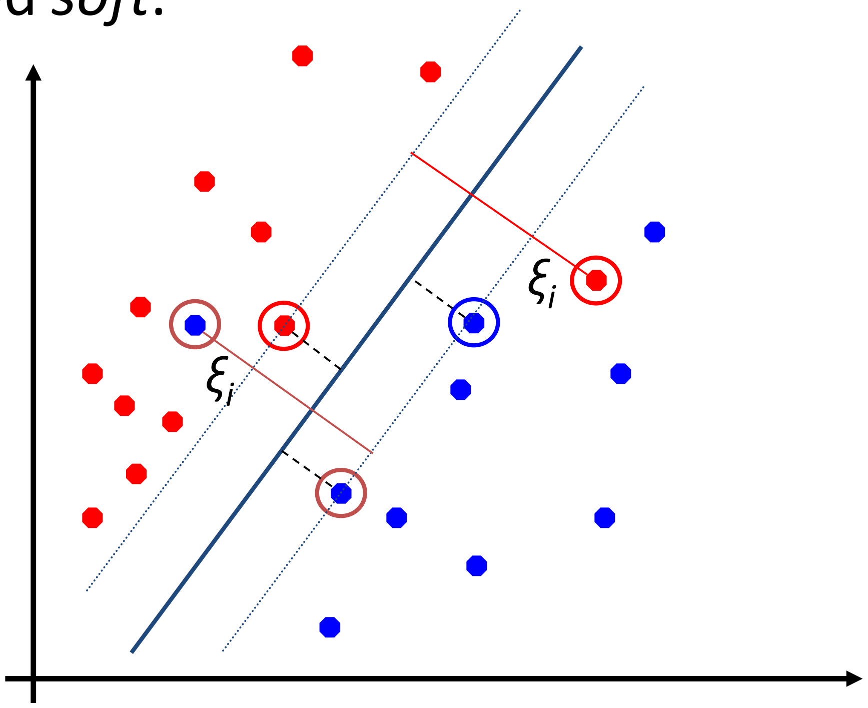
- We make decisions by comparing each new example $\mathbf{x}'$ with only the support vectors:

$$y^* = \text{sign}\left(\sum_{i \in SV} \alpha_i y_i (\mathbf{x}_i^\top \mathbf{x}') + b\right)$$

Soft Margin Classification

What if the training set is not linearly separable?

Slack variables（*松弛变量*）ξ_i can be added to allow misclassification of difficult or noisy examples, resulting margin called *soft*.



Soft Margin Classification Mathematically

"Hard" margin QP:

$$\min_{\mathbf{w},b} \mathbf{w}^\top \mathbf{w} \quad \text{subject to} \quad y_i(\mathbf{w}^\top \mathbf{x}_i + b) \geq 1, \forall i$$

Soft margin QP:

$$\min_{\mathbf{w},b} \frac{1}{2}\mathbf{w}^\top \mathbf{w} + C\sum_i \xi_i \quad \text{subject to} \quad y_i(\mathbf{w}^\top \mathbf{x}_i + b) \geq 1 - \xi_i, \forall i$$

$$\xi_i \geq 0, \forall i$$

- Note that ξ_i=0 if there is no error for $\mathbf{x_i}$
- ξ_i is an upper bound of the number of errors
- Parameter *C* can be viewed as a way to control "softness": it "trades off（折衷，权衡）" the relative importance of maximizing the margin and fitting the training data (minimizing the error).
  - Larger C → more reluctant to make mistakes

The Optimization Problem

The dual of this new constrained optimization problem is

$$\max_{\alpha} \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{n} \alpha_i \alpha_j y_i y_j (\mathbf{x}_i^\top \mathbf{x}_j) \quad \text{subject to} \quad 0 \le \alpha_i \le C, \forall i$$

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

This is very similar to the optimization problem in the linear separable case, except that there is an upper bound *C* on α_i now

Once again, a QP solver can be used to find α_i

Roadmap

SVM Prediction

Loss Minimization

Loss in SVM

$$\min_{\mathbf{w},b,\xi} \frac{1}{2}\mathbf{w}^\top\mathbf{w} + C\sum_i \xi_i \quad \text{subject to} \qquad y_i(\mathbf{w}^\top\mathbf{x}_i + b) \geq 1 - \xi_i, \forall i$$

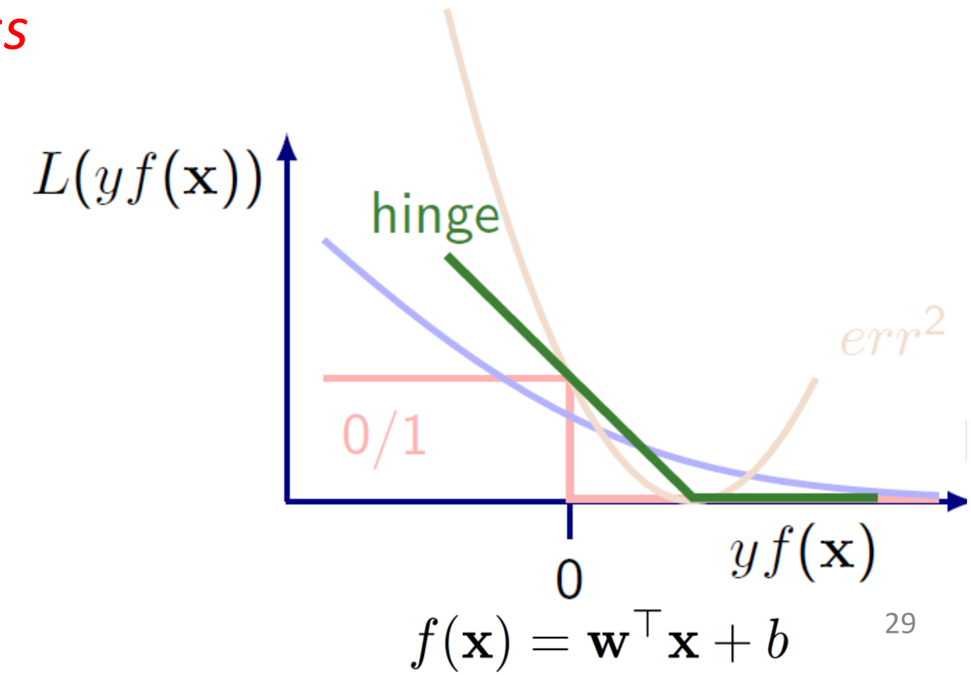
$$\xi_i \geq 0, \forall i$$

Loss is measured as

$$\xi_i = \max\left(0, 1 - y_i(\mathbf{w}^\top\mathbf{x}_i + b)\right) = [1 - y_i(\mathbf{w}^\top\mathbf{x}_i + b)]_+$$

This loss is known as *hinge loss*

$$\min_{\mathbf{w},b} \frac{1}{2C}\mathbf{w}^\top\mathbf{w} + \sum_i hingeloss_i$$



Loss functions

- Regression
  - Squared loss L_2
  - Absolute loss L_1

- Binary classification
  - Zero/one loss $L_{0/1}$ (no good algorithm)
  - Squared loss L_2
  - Absolute loss L_1
  - Hinge loss (Support vector machines)
  - Logistic loss (Logistic regression)

Linear SVMs: Overview

The classifier is a *separating hyperplane.*

Most "important" training points are support vectors; they define the hyperplane.

Quadratic optimization algorithms can identify which training points $\mathbf{x}_i$ are support vectors with non-zero Lagrangian multipliers α_i.

Both in the dual formulation of the problem and in the solution training points appear only inside inner products:

Find $\alpha_1 \ldots \alpha_N$ such that
$\mathbf{Q}(\boldsymbol{\alpha}) = \Sigma\alpha_i - \frac{1}{2}\Sigma\Sigma\alpha_i\alpha_j y_i y_j \mathbf{x}_i^T\mathbf{x}_j$ is maximized and
(1) $\Sigma\alpha_i y_i = 0$
(2) $0 \leq \alpha_i \leq C$ for all α_i

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

Non-linearity: example

- Input x:
  - Patient information and vital signs
- Output y:
  - Health (positive is good)

Features in linear space

- Philosophy: extract any features that might be relevant.
- Features for medical diagnosis: height, weight, body temperature, blood pressure, etc.
- Three problems: non-monotonicity, non-linearity, interactions between features

Non-monotonicity

- Features: $\phi(x) = (1; \text{temperature}(x))$
- Output: health y
- Problem: favor extremes; true relationship is non-monotonic

Non-monotonicity

- Solution: transform inputs

- $\phi(x) = (1; (\text{temperature}(x)-37)^2)$

- Disadvantage: requires manually-specied domain knowledge

Non-monotonicity

- $\phi(x) = (1; \text{temperature}(x); \text{temperature}(x)^2)$

- General: features should be simple building blocks to be pieced together

Interaction between features

- $\phi(x)$ = (height(x); weight(x))
- Output: health y
- Problem: can't capture relationship between height and weight

Interaction between features

- $\phi(x) = (\text{height}(x)-\text{weight}(x))^2$
- Solution: define features that combine inputs
- Disadvantage: requires manually-specified domain knowledge

Interaction between features

- $\phi(x)=[height(x)^2;weight(x)^2;\underline{height(x)weight(x)}]$

  cross term

- Solution: add features involving multiple measurements

Linear in what?

Prediction driven by score: $w \cdot \phi(x)$

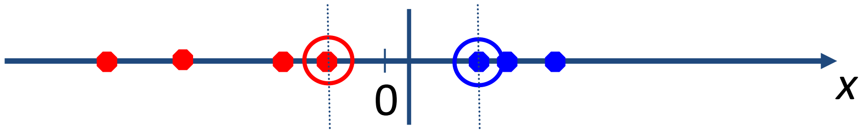
- Linear in w? Yes
- Linear in $\phi(x)$? Yes
- Linear in x? No!

Key idea: non-linearity

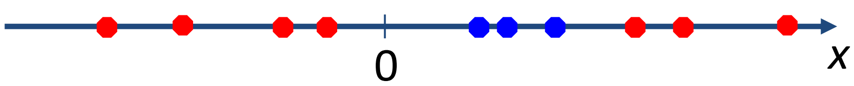
- Predictors $f_w(x)$ can be expressive non-linear functions and decision boundaries of x.
- Score $w \cdot \phi(x)$ is linear function of w and $\phi(x)$

Non-linear SVMs

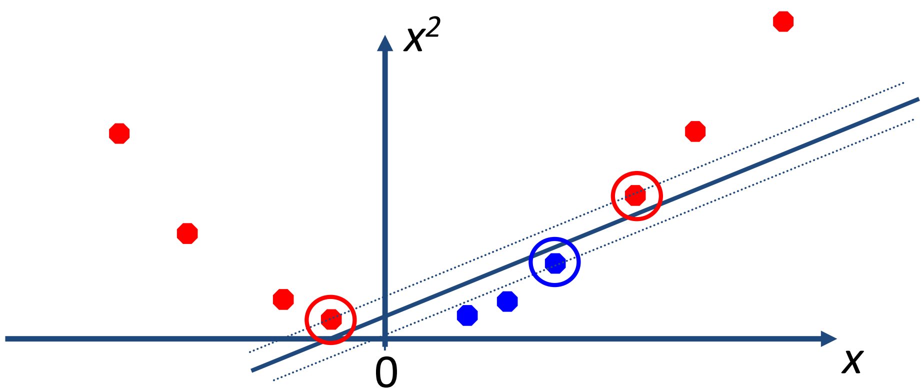
Datasets that are linearly separable with some noise work out great:



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

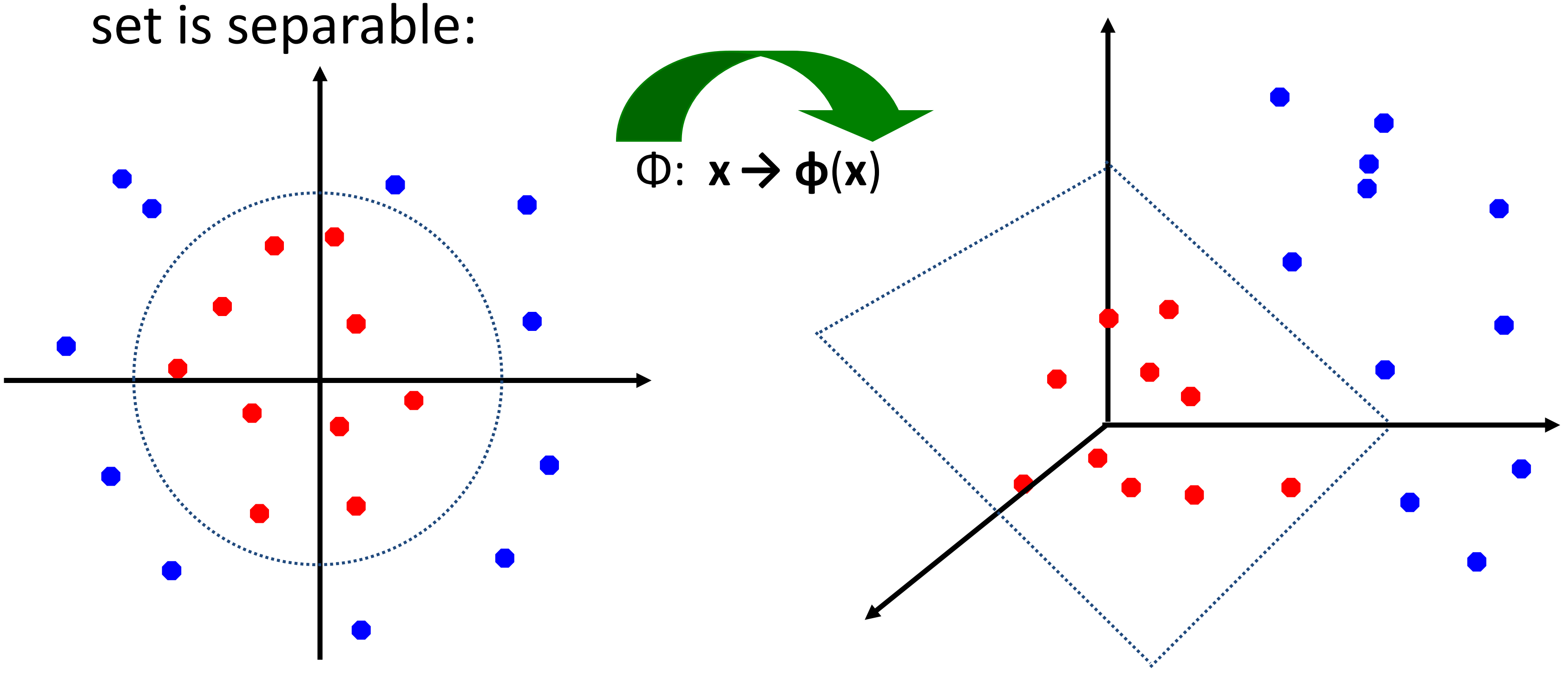


How about… mapping data to a higher-dimensional space:



Non-linear SVMs: Feature spaces

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



The "Kernel Trick"

Recall the SVM optimization problem

$$\max_{\alpha} \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{n} \alpha_i \alpha_j y_i y_j (\mathbf{x}_i^\top \mathbf{x}_j) \quad \text{subject to} \quad 0 \leq \alpha_i \leq C, \forall i$$

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

The data points only appear as inner product

- As long as we can calculate the inner product in the feature space, we do not need the mapping explicitly
- Many common geometric operations (angles, distances) can be expressed by inner products

Define the kernel function *K* by $K(\mathbf{x}_i,\mathbf{x}_j) = \phi(\mathbf{x}_i)^{\mathbf{T}} \phi(\mathbf{x}_j)$

Kernel methods

- Features viewpoint: construct and work with $\phi(\mathbf{x})$ (think in terms of **properties** of inputs)

- Kernel viewpoint: construct and work with $K(\mathbf{x}_i,\mathbf{x}_j)$ (think in terms of **similarity** between inputs)

An Example for feature mapping and kernels

- Consider an input $\mathbf{x}=[x_1,x_2]$
- Suppose $\phi(.)$ is given as follows

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

- An inner product in the feature space is

$$\left\langle \phi\left(\begin{bmatrix} x_1 \\ x_2 \end{bmatrix}\right), \phi\left(\begin{bmatrix} x_1' \\ x_2' \end{bmatrix}\right) \right\rangle =$$

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

$$K(\mathbf{x},\mathbf{x}') = \left(1 + \mathbf{x}^T\mathbf{x}'\right)^2$$

More Examples of Kernel Functions

- Linear: $K(\mathbf{x}_i,\mathbf{x}_j)= \mathbf{x}_i^{\mathbf{T}}\mathbf{x}_j$
  - Mapping Φ: $\mathbf{x} \rightarrow \boldsymbol{\varphi}(\mathbf{x})$, where $\boldsymbol{\varphi}(\mathbf{x})$ is $\mathbf{x}$ itself
- Polynomial（多项式） of power *p*: $K(\mathbf{x}_i,\mathbf{x}_j)= (1+ \mathbf{x}_i^{\mathbf{T}}\mathbf{x}_j)^p$
- Gaussian (radial-basis function径向基函数): $K(\mathbf{x}_i,\mathbf{x}_j) = e^{-\frac{\|\mathbf{x}_i-\mathbf{x}_j\|^2}{\sigma^2}}$
  - Mapping Φ: $\mathbf{x} \rightarrow \boldsymbol{\varphi}(\mathbf{x})$, where $\boldsymbol{\varphi}(\mathbf{x})$ is *infinite-dimensional*
- Higher-dimensional space still has *intrinsic* dimensionality *d*, but linear separators in it correspond to *non-linear* separators in original space.

Kernel matrix

Suppose for now that K is indeed a valid kernel corresponding to some feature mapping ϕ, then for $x_1, \ldots, x_n$, we can compute an $n \times n$ matrix $\{K_{i,j}\}$ *where* $K_{i,j} = \boldsymbol{\varphi}(\mathbf{x}_i)^{\mathbf{T}} \boldsymbol{\varphi}(\mathbf{x}_j)$

This is called a kernel matrix!

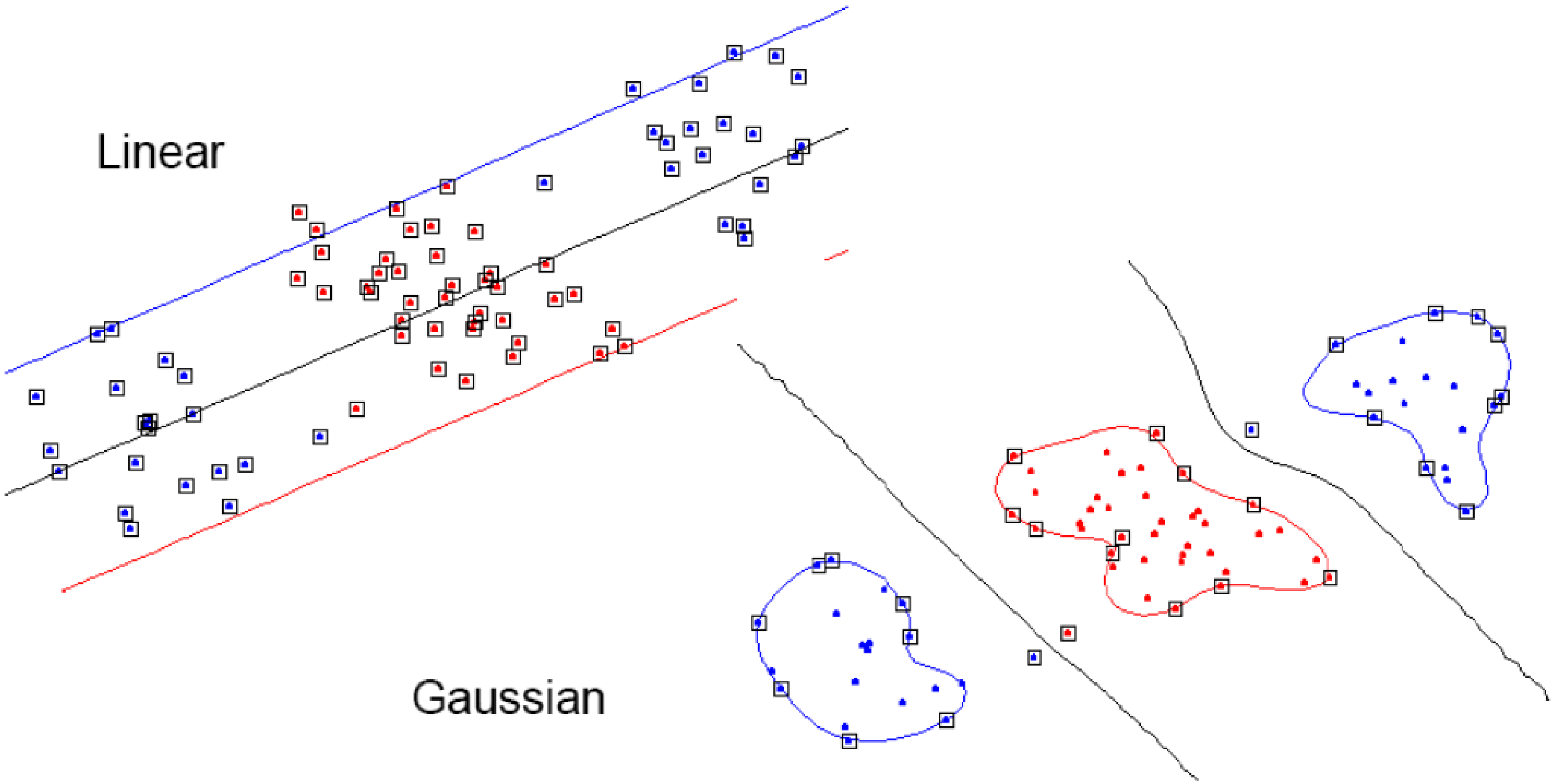
Now, if a kernel function is indeed a valid kernel, and its elements are dot-product in the transformed feature space, it must satisfy:

– Symmetry $K=K^T$
– Positive-semidefinite（半正定） $\mathbf{z}^\top K \mathbf{z} \geq 0, \quad \forall \mathbf{z} \in R^n$

Matrix formulation

$$
\begin{aligned}
& \max_\alpha \textstyle\sum_{i=1}^n \alpha_i - \frac{1}{2} \sum_{i,j=1}^n \alpha_i \alpha_j y_i y_j K(\mathbf{x}_i, \mathbf{x}_j) \\
= \quad & \max_\alpha \textstyle\sum_{i=1}^n \alpha_i - \frac{1}{2} \sum_{i,j=1}^n \alpha_i \alpha_j y_i y_j K_{i,j} \\
= \quad & \max_\alpha \alpha^\top \mathbf{e} - \tfrac{1}{2} \alpha^\top (\mathbf{y}\mathbf{y}^\top \circ K) \alpha \\
\text{subject to} \quad & 0 \le \alpha_i \le C, \forall i \\
& \textstyle\sum_{i=1}^n \alpha_i y_i = 0
\end{aligned}
$$

Nonlinear SVMs – RBF Kernel



Summary: Support Vector Machines

Linearly separable case → Hard margin SVM

- Primal quadratic programming
- Dual quadratic programming

Not linearly separable? → Soft margin SVM

Non-linear SVMs

- Kernel trick

Summary: Support Vector Machines

SVM training: build a kernel matrix K using training data

- Linear: $K(\mathbf{x}_i,\mathbf{x}_j)= \mathbf{x}_i^{\mathbf{T}}\mathbf{x}_j$
- Gaussian (radial-basis function径向基函数): $K(\mathbf{x}_i,\mathbf{x}_j) = e^{-\frac{\left\|\mathbf{x}_i-\mathbf{x}_j\right\|^2}{\sigma^2}}$

solve the following quadratic program

$$\max_\alpha \alpha^\top \mathbf{e} - \tfrac{1}{2}\alpha^\top(\mathbf{y}\mathbf{y}^\top \circ K)\alpha$$

$$\text{subject to} \quad 0 \leq \alpha_i \leq C, \forall i$$

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

SVM testing: now with α_i, recover b,

$$b = y_i - \sum_{j=1}^{n} \alpha_j y_j K(\mathbf{x}_i, \mathbf{x}_j) \quad \text{for any i that } \alpha_i \neq 0$$

we can predict new data points by:

$$y^* = \operatorname{sign}\left(\textstyle\sum_{i \in SV} \alpha_i y_i K(\mathbf{x}_i, \mathbf{x}') + b\right)$$

作业

- 已知正例点 $x_1=(1,2)^{\mathrm{T}}, x_2=(2,3)^{\mathrm{T}}, x_3=(3,3)^{\mathrm{T}}$,
  负例点 $x_4=(2,1)^{\mathrm{T}}, x_5=(3,2)^{\mathrm{T}}$，试求Hard Margin SVM的最大间隔分离超平面和分类决策函数，并在图上画出分离超平面、间隔边界及支持向量。