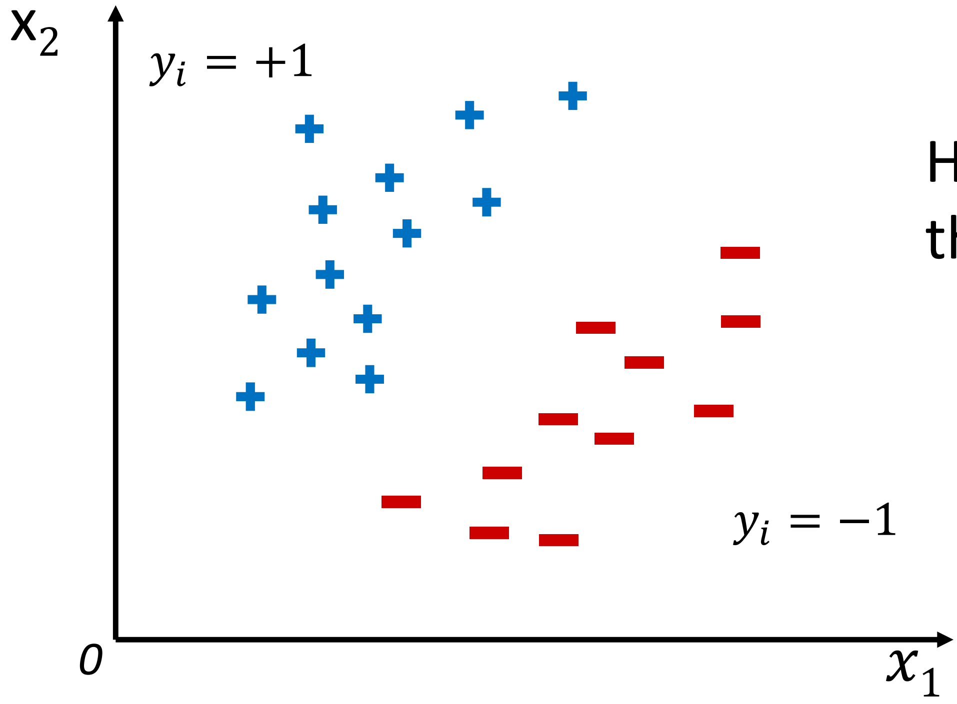
Support vector machine

Outline

- Margin and support vector
- Dual problem
- Kernel function
- Soft margin
- Support Vector Regression
- Usage of LIBSVM with matlab

Linear Classifier

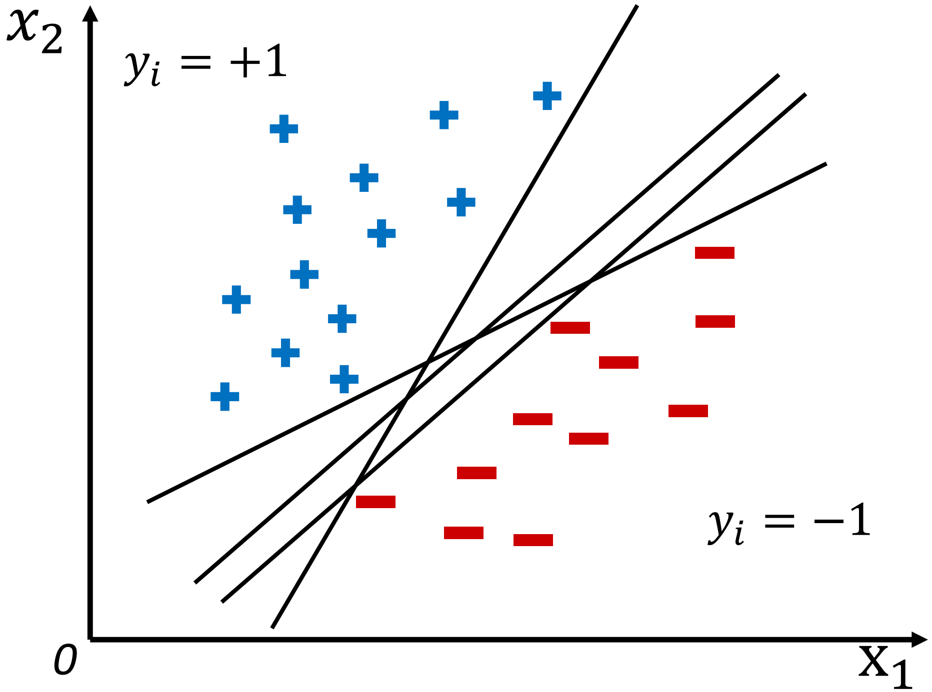
Train data set D = $\{(\boldsymbol{x}_1, y_1),(\boldsymbol{x}_2,y_2),\ldots,(\boldsymbol{x}_m,y_m)\}$, $y_i \in \{-1, +1\}$



How would you classify this data correctly ?

Linear Classifier

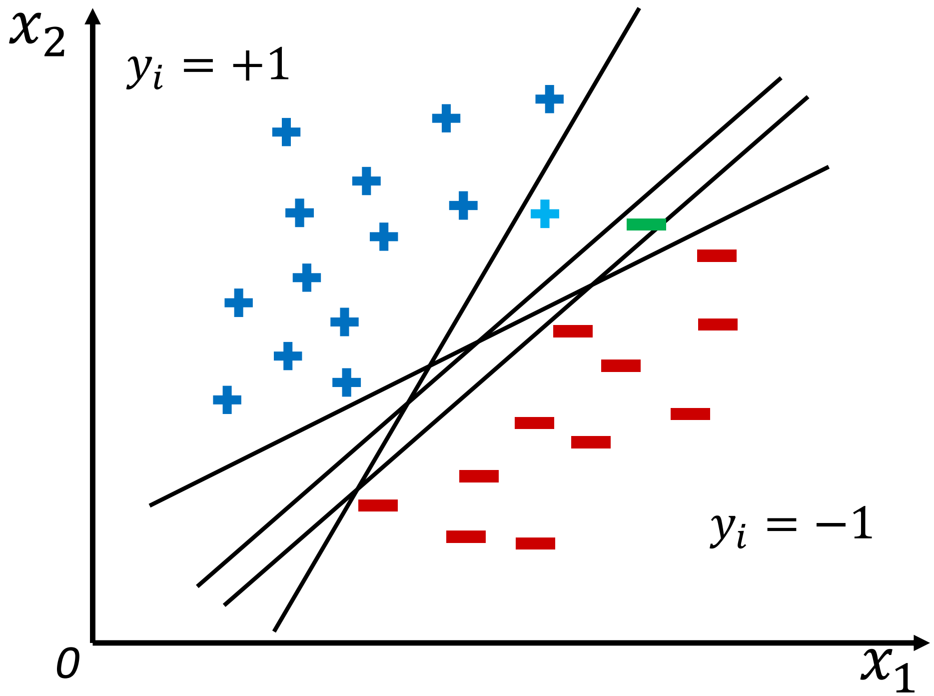
Train data set D = $\{(\boldsymbol{x}_1, y_1),(\boldsymbol{x}_2,y_2),\ldots,(\boldsymbol{x}_m,y_m)\}$, $y_i \in \{-1, +1\}$



Any of these would be fine…

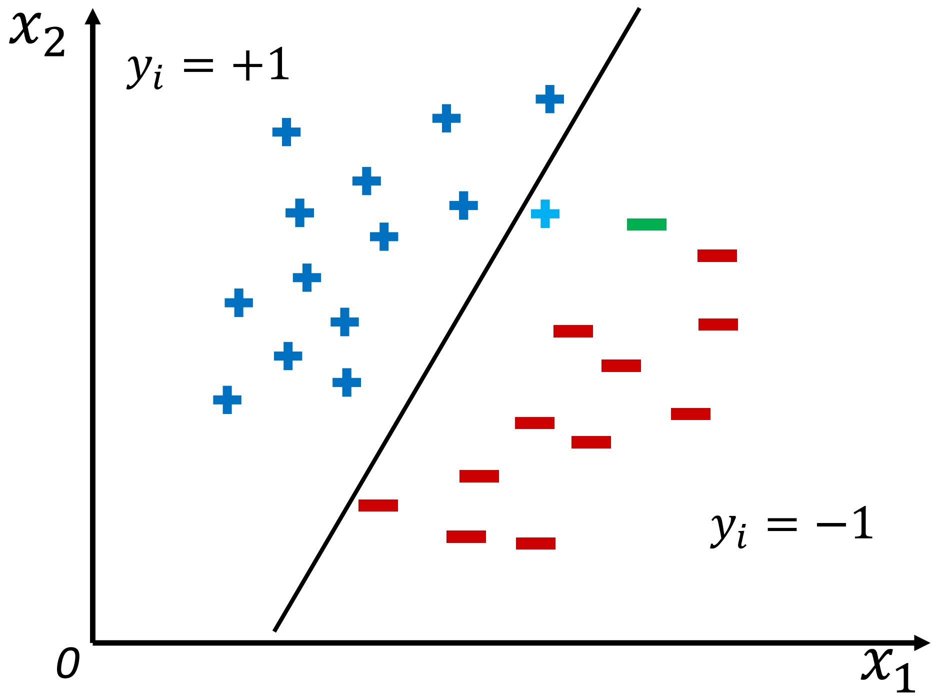
…but which one is the best?

Linear Classifier



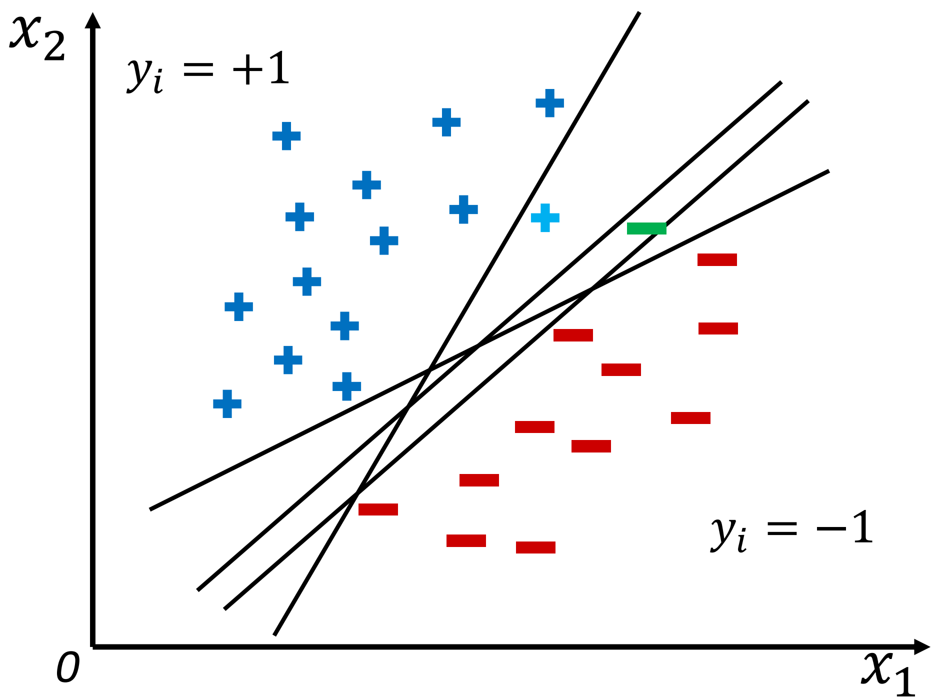
- In addition to correction, we manage to find a robust classifier which has good generalization ability(泛化能力) to new data.

Linear Classifier



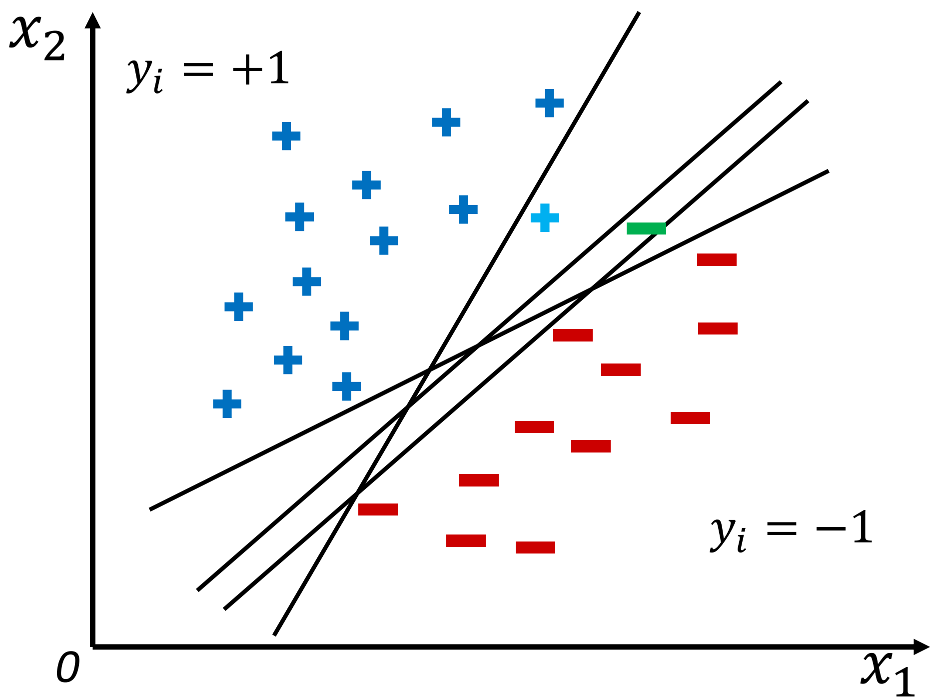
- In addition to correction, we manage to find a robust classifier which has good generalization ability(泛化能力) to new data.

Linear Classifier



- In addition to correction, we manage to find a robust classifier which has good generalization ability(泛化能力) to new data.

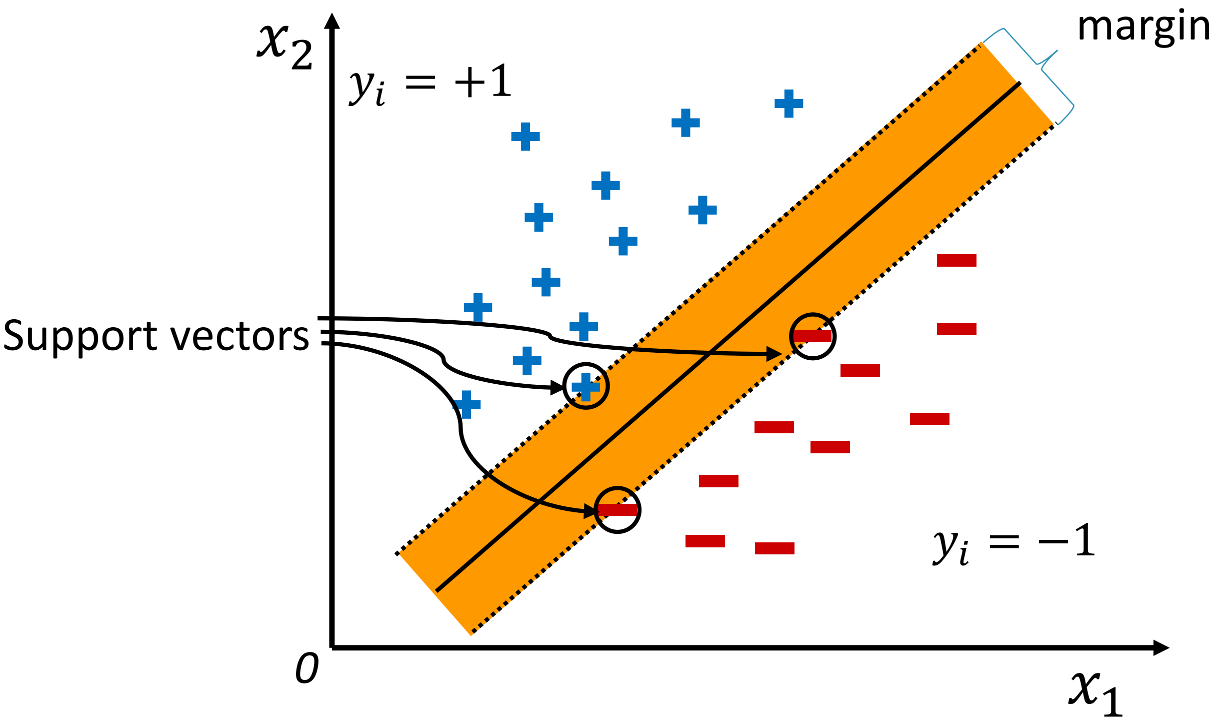
Linear Classifier



In order to classify new data closer to the line correctly , the minimum distance between the line and training samples should be as far as possible.

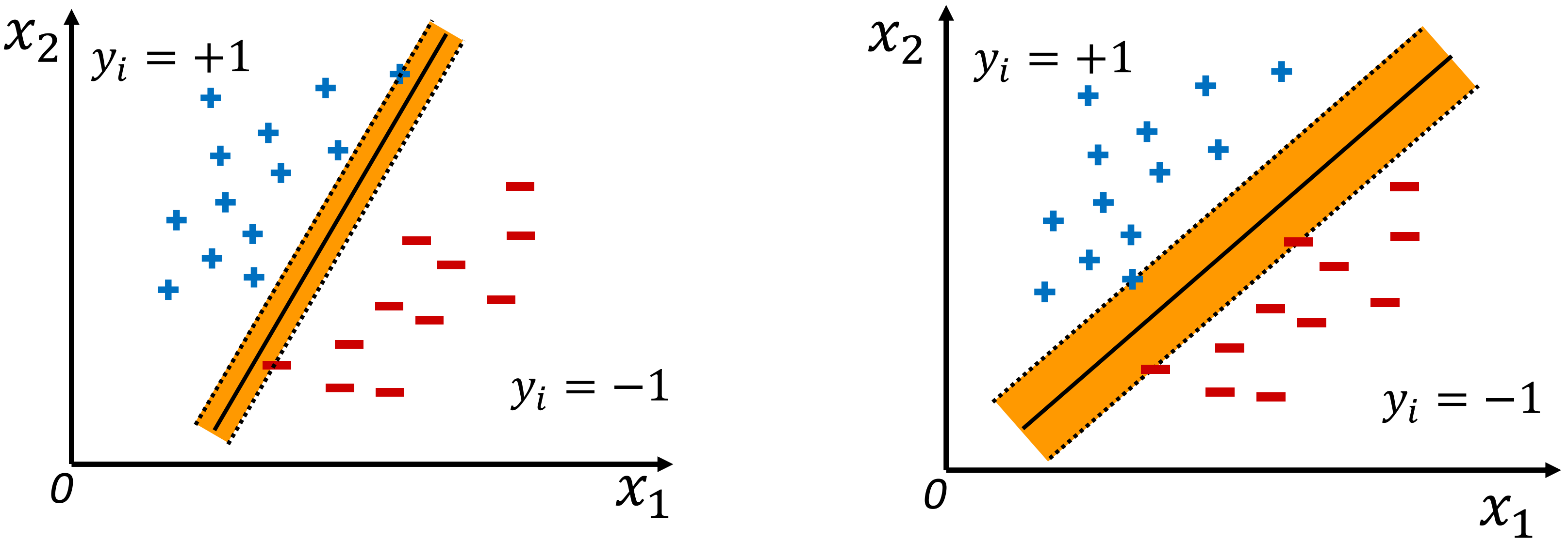
- In addition to correction, we manage to find a robust classifier which has good generalization ability(泛化能力) to new data.

Margin(间隔)



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

Maximum margin classifier



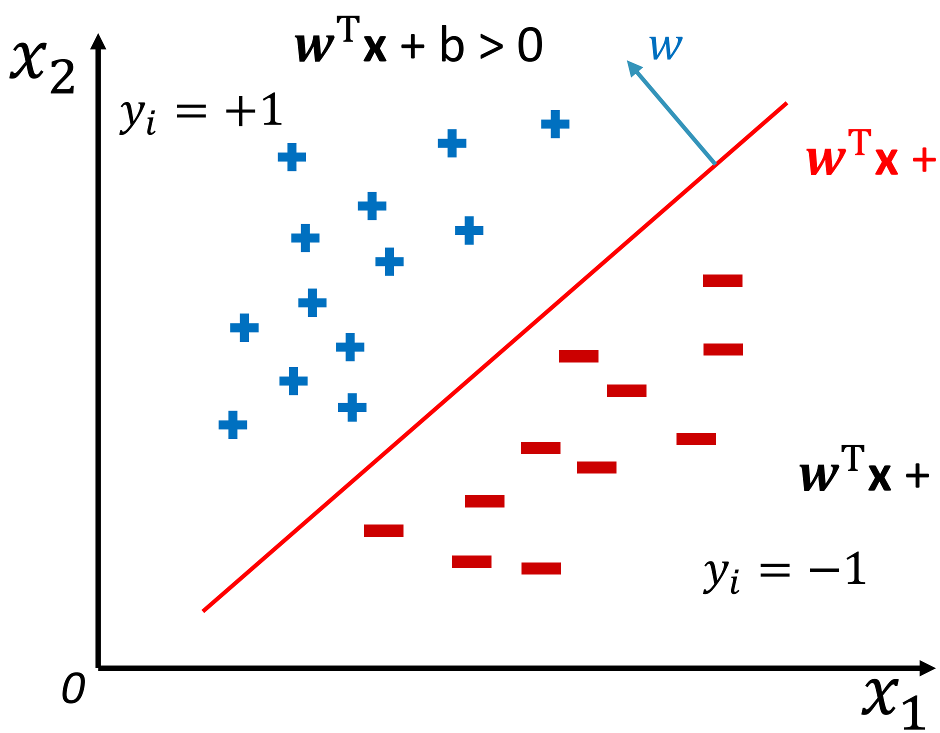
Each classifier corresponds to a margin.

Pick the one with the largest margin !

Specifying a hyper-plane

- A hyper-plane: $\boldsymbol{w}^{\mathrm{T}}\mathbf{x} + \mathrm{b} = 0$
- For $(x_i, y_i) \in D$, if y_i=+1, $\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}_i + \mathrm{b} > 0$; if y_i=-1, $\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}_i + \mathrm{b} < 0$

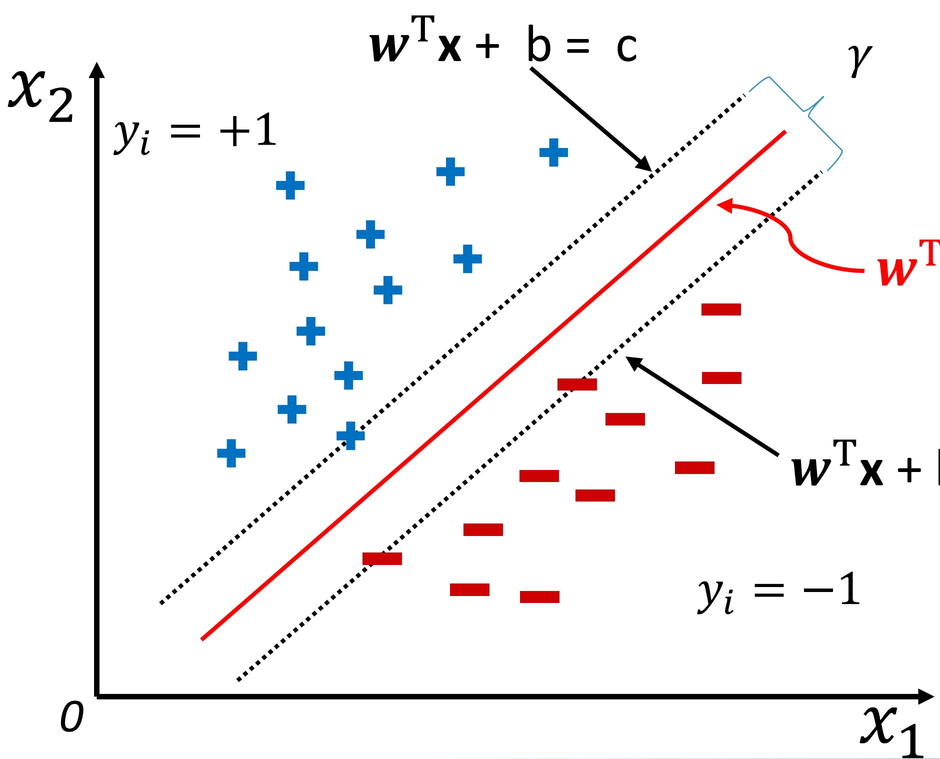
Correction constraints



Maximum margin

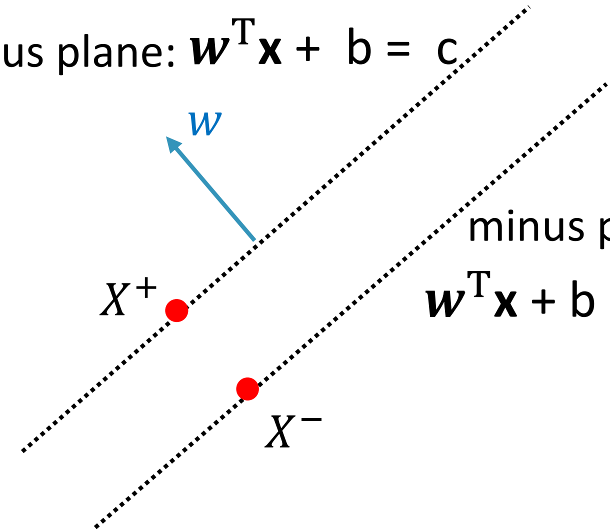
- Margin: γ
- Goal : max{margin}

What is the value of γ ?



Maximum margin

plus plane: $\boldsymbol{w}^{\mathrm{T}}\mathbf{x} + b = c$



minus plane

$\boldsymbol{w}^{\mathrm{T}}\mathbf{x} + b = -c$

- Let X^- be any point on the minus plane
- Let X^+ be the closest plus-plane-point to X^-.

Claim: $X^+ = X^- + \alpha\, \boldsymbol{w}$ for some value of α.

Why?

What we know:

- $\boldsymbol{w}^{\mathrm{T}} X^+ + b = c$
- $\boldsymbol{w}^{\mathrm{T}} X^- + b = -c$
- $X^+ = X^- + \alpha\, \boldsymbol{w}$
- $|X^+ - X^-| = \gamma$

$\boldsymbol{w}^{\mathrm{T}}(X^- + \alpha\, \boldsymbol{w}) + b = c$

=>

$\boldsymbol{w}^{\mathrm{T}} X^- + b + \alpha\, \boldsymbol{w}^{\mathrm{T}}\mathbf{w} = c$

=>

$-c + \alpha\, \boldsymbol{w}^{\mathrm{T}}\mathbf{w} = c$

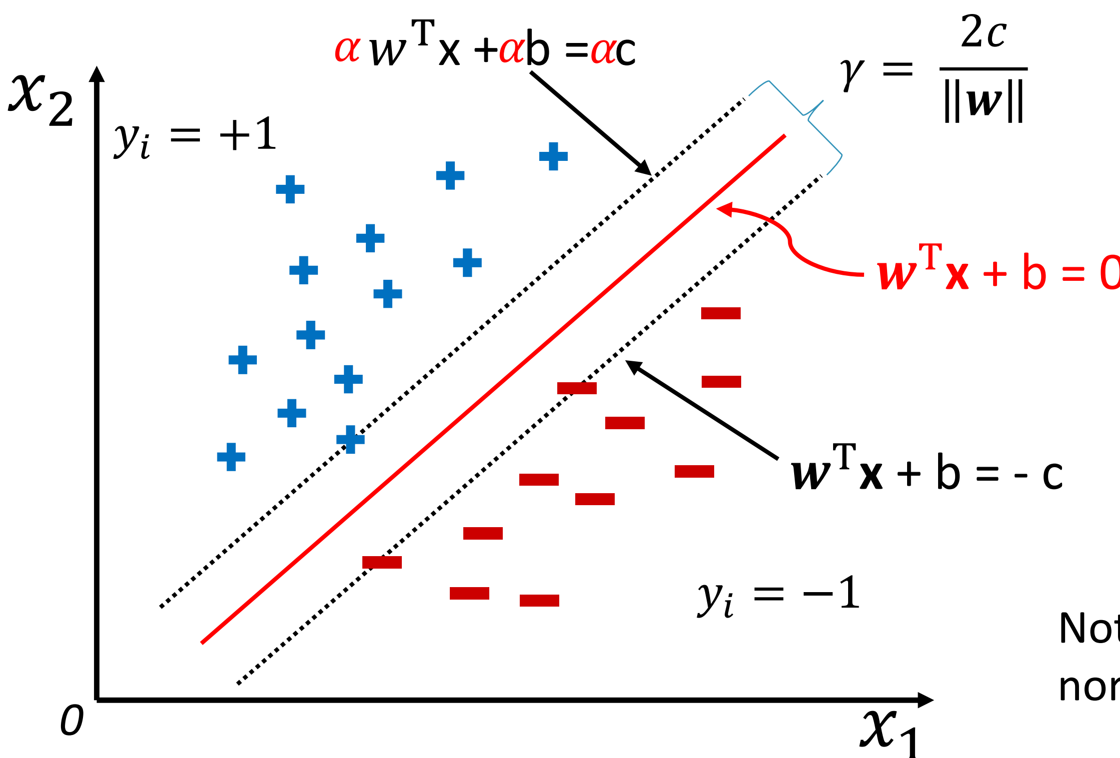
=>

$$\alpha = \frac{2c}{\mathbf{w}^{\mathrm{T}}\,\mathbf{w}} = \frac{2c}{\|\mathbf{w}\|^2}$$

$$\gamma = |X^+ - X^-| = |\alpha\, \boldsymbol{w}| = \alpha |\boldsymbol{w}| = \frac{2c}{\|\mathbf{w}\|^2}\|\boldsymbol{w}\| = \frac{2c}{\|\boldsymbol{w}\|}$$

Maximum margin

- Margin $\gamma = \dfrac{2c}{\|\boldsymbol{w}\|}$
- Goal : max{margin}

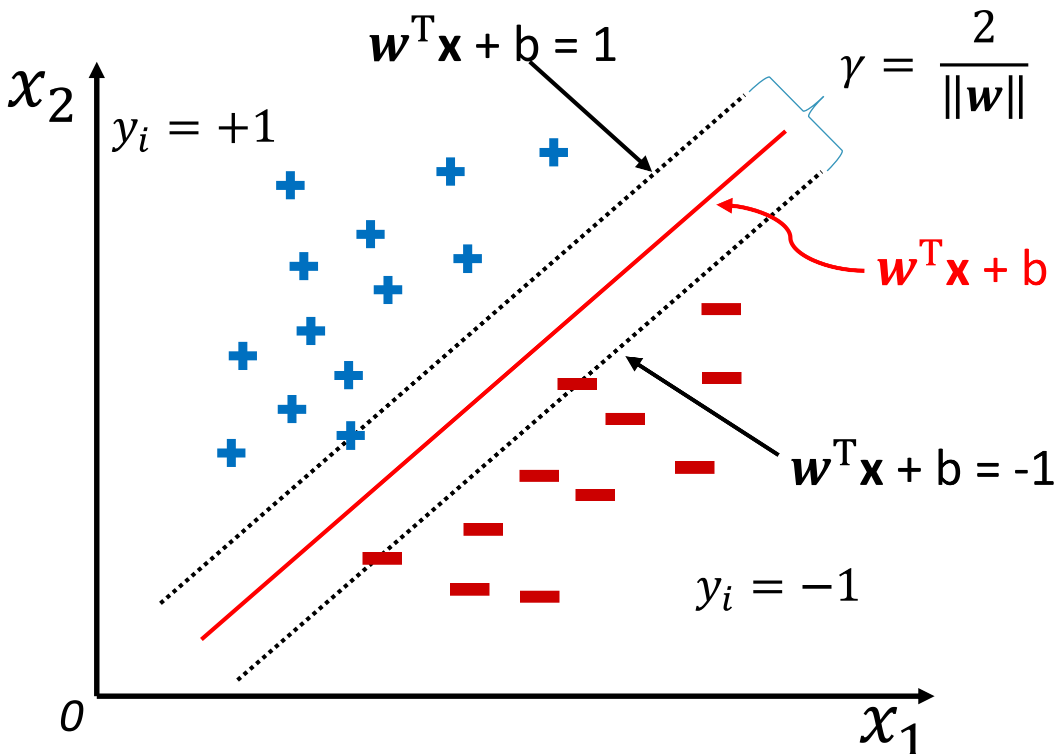


Note: ‘c’ is arbitrary (can normalize equations by c)

Maximum margin

- Margin $\gamma = \dfrac{2}{\|\boldsymbol{w}\|}$
- Goal : max{margin}

$\alpha = \dfrac{1}{c}$



Note: the 'w' here is equal to $\frac{1}{c}$ w of last slide , 'b' too

Primal optimization problem

- $\boldsymbol{w}^T\boldsymbol{x}_i + b \geq 1, \text{ if } y_i = +1$
- $\boldsymbol{w}^T\boldsymbol{x}_i + b \leq -1, \text{ if } y_i = -1$

$\longrightarrow$ $y_i(\boldsymbol{w}^T\boldsymbol{x}_i + b) \geq 1, i = 1,2, \ldots m$

$$\max_{w,b} \frac{2}{\|\boldsymbol{w}\|}$$

$$\text{s.t. } y_i(\boldsymbol{w}^T\boldsymbol{x}_i + b) \geq 1, \quad i = 1,2, \ldots m$$

$\downarrow$

$$\min_{w,b} \frac{1}{2}\|\boldsymbol{w}\|^2$$

$$\text{s.t. } y_i(\boldsymbol{w}^T\boldsymbol{x}_i + b) \geq 1, \quad i = 1,2, \ldots m$$

Outline

- Margin and support vector
- Dual problem
- Kernel function
- Soft margin and regularization
- Support Vector Regression
- Usage of LIBSVM with matlab

Lagrangian duality

- Primal optimization problem:

$$\min_{w,b} \boxed{\frac{1}{2}\|\boldsymbol{w}\|^2} \qquad (*)$$
$$\text{s.t. } y_i(\boldsymbol{w}^T\boldsymbol{x}_i + b) \geq 1, \quad i = 1,2,\ldots m$$

It is a convex quadratic programming problem, can be resolved by existed tool. But we have more efficient method:

- Lagrangian:

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

$$\alpha = (\alpha_1; \alpha_2; \ldots; \alpha_m), \alpha_i \geq 0$$

$$\max_{\alpha:\alpha_i \geq 0} L(\boldsymbol{w}, b, \alpha) = \begin{cases} \boxed{\frac{1}{2}\|\boldsymbol{w}\|^2}, & \text{If } w,b \text{ satisfies primal constraints} \\ \infty, & \text{otherwise} \end{cases}$$

- re-written (*) : $\min_{w,b} \max_{\alpha:\alpha_i \geq 0} L(\boldsymbol{w}, b, \alpha)$

Lagrangian duality

- Recall the primal problem:

$$\min_{w,b} \max_{\alpha:\alpha_i \geq 0} L(\boldsymbol{w}, b, \alpha) = \frac{1}{2}\|\boldsymbol{w}\|^2 + \sum_{i=1}^{m} \alpha_i (1 - y_i(\boldsymbol{w}^T \boldsymbol{x}_i + b))$$

- The dual problem:

$$\max_{\alpha:\alpha_i \geq 0} \min_{w,b} L(\boldsymbol{w}, b, \alpha) = \frac{1}{2}\|\boldsymbol{w}\|^2 + \sum_{i=1}^{m} \alpha_i (1 - y_i(\boldsymbol{w}^T \boldsymbol{x}_i + b))$$

- We change to resolve the dual problem , why?
  - Easier to resolve (can use an efficient algorithm better than generic QP software)
  - Allow us to use kernels used in linearly inseparable problem See later…
- Can we do that ?

$$d^* = \max_{\alpha:\alpha_i \geq 0} \min_{\boldsymbol{w},b} L(\boldsymbol{w}, b, \alpha) \leq \min_{\boldsymbol{w},b} \max_{\alpha:\alpha_i \geq 0} L(\boldsymbol{w}, b, \alpha) = p^*$$

but , under certain conditions: $d^* = p^*$

Fortunately , our problem satisfies the “certain conditions”.

Lagrangian duality

- The dual problem:

$$\max_{\alpha:\alpha_i \geq 0} \boxed{\min_{w,b} L(\boldsymbol{w}, b, \alpha) = \frac{1}{2}\|\boldsymbol{w}\|^2 + \sum_{i=1}^{m} \alpha_i (1 - y_i(\boldsymbol{w}^T \boldsymbol{x}_i + b))}$$

- We minimize *L* with respect to *w* and *b* first:

$$\frac{\partial L}{\partial \boldsymbol{w}} = 0 \longrightarrow \boldsymbol{w} = \sum_{i=1}^{m} \alpha_i y_i \boldsymbol{x}_i \qquad (*)$$

$$\frac{\partial L}{\partial b} = 0 \longrightarrow \sum_{i=1}^{m} \alpha_i y_i = 0 \qquad (**)$$

- Plug (*) back to $L(\boldsymbol{w}, b, \alpha)$, and using (**), we have:

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

Dual problem

- Now we have the dual problem:

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

$$\text{s.t.} \quad \begin{aligned} &\textstyle\sum_{i=1}^{m} \alpha_i y_i = 0, \\ &\alpha_i \geq 0, \quad i = 1,2, \ldots, m \end{aligned}$$

- Resolve the above problem, we will get α, then $\boldsymbol{w}, b$, and then the final model:

$$f(\boldsymbol{x}) = \boldsymbol{w}^T \boldsymbol{x} + b = \sum_{i=1}^{m} \alpha_i y_i \boldsymbol{x}_i{}^T \boldsymbol{x} + b$$

how to resolve the dual problem will be told later…

KKT conditions

- Under ''certain conditions'' , there must exist $\boldsymbol{w}, b, \alpha$ so that $(\boldsymbol{w}, b)$ is the solution to the primal problem, α is the solution to the dual problem, $(\boldsymbol{w}, b, \alpha)$ satisfy the *Karush-Kuhn-Tucker (KKT)* conditions, which are as follows:

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

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

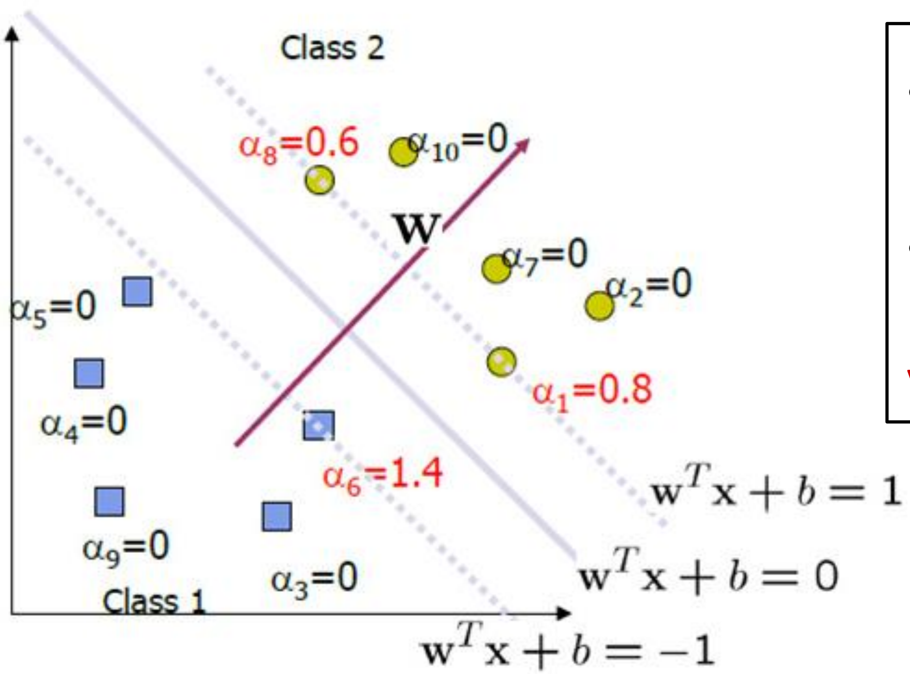
$$\alpha_i \geq 0$$

$$y_i(\boldsymbol{w}^T \boldsymbol{x}_i + b) - 1 \geq 0$$

$$\boxed{\alpha_i(y_i(\boldsymbol{w}^T \boldsymbol{x}_i + b) - 1) = 0.}$$

Support vector

$$\alpha_i(y_i(\boldsymbol{w}^T\boldsymbol{x}_i + b) - 1) = 0$$



- $y_i(\boldsymbol{w}^T\boldsymbol{x}_i + b) - 1 > 0 \Rightarrow \alpha_i = 0$
- $\alpha_i > 0 \Rightarrow y_i(\boldsymbol{w}^T\boldsymbol{x}_i + b) = 1$

$\boldsymbol{x}_i$ is on the margin, is a support vector

- Call the training data points whose α are nonzero the **support vectors** (SV)

Support vector

- The model:

$$f(\boldsymbol{x}) = \boldsymbol{w}^T\boldsymbol{x} + b = \sum_{i=1}^{m} \alpha_i y_i \boldsymbol{x}_i{}^T \boldsymbol{x} + b$$

  - Many α_i are zero.
  - The final model $f(x)$ is determined only by the support vectors , since:

$$f(\boldsymbol{x}) = \boldsymbol{w}^T\boldsymbol{x} + b = \sum_{\alpha_i \neq 0} \alpha_i y_i \boldsymbol{x}_i{}^T \boldsymbol{x} + b$$

- For a test data $\boldsymbol{z}$, compute:

$$f(\boldsymbol{z}) = \boldsymbol{w}^T z + b = \sum_{\alpha_i \neq 0} \alpha_i y_i \boldsymbol{x}_i{}^T \boldsymbol{z} + b$$

classify $\boldsymbol{z}$ as class 1 if the sum is positive, and class -1 otherwise

SMO algorithm

- Now we back to resolve the dual problem:

$$\max_{\alpha} \ L(\alpha) = \sum_{i=1}^{m} \alpha_i - \frac{1}{2}\sum_{i=1}^{m}\sum_{j=1}^{m} \alpha_i \alpha_j y_i y_j \boldsymbol{x}_i{}^T \boldsymbol{x}_j$$

s.t. $\sum_{i=1}^{m} \alpha_i y_i = 0,$
$\alpha_i \geq 0, \quad i = 1,2,\ldots,m$

- SMO(sequential minimal optimization) algorithm:

Repeat till convergence:

1. Select some pair α_i and α_j to update next

2. Re-optimize *L(α)* with respect to α_i and α_j, while holding all the other α_k (*k* ≠ *i*; *j*) fixed.

Determine 'w' & 'b'

- w: $\quad \boldsymbol{w} = \sum_{i=1}^{m} \alpha_i y_i \boldsymbol{x}_i$

- b: For any support vector $(\boldsymbol{x}_s, y_s)$,

$$y_s \left(\sum_{i \epsilon \mathbf{t}} \alpha_i y_i {\boldsymbol{x}_i}^T \boldsymbol{x}_s + b \right) = 1$$

$\boldsymbol{t} = \{i | \alpha_i > 0, i = 1,2, \dots m\}$, the subscript set of support vectors

$$\sum_{i \epsilon \mathbf{t}} \alpha_i y_i {\boldsymbol{x}_i}^T \boldsymbol{x}_s + b = \frac{1}{y_s} = y_s$$

$$b = y_s - \sum_{i \epsilon \mathbf{t}} \alpha_i y_i {\boldsymbol{x}_i}^T \boldsymbol{x}_s$$

summary

$$\min_{w,b} \ \frac{1}{2}\|w\|^2$$
$$\text{s.t. } y_i(w^T x_i + b) \geq 1, \quad i = 1,2,\dots m$$

- $L(w,b,\alpha)$
- $\min\limits_{w,b} \max\limits_{\alpha:\alpha_i \geq 0} L(w,b,\alpha)$
- $\max\limits_{\alpha:\alpha_i \geq 0} \ \min\limits_{w,b} L(w,b,\alpha)$

$$\max_{\alpha} \ L(\alpha) = \sum_{i=1}^{m} \alpha_i - \frac{1}{2}\sum_{i=1}^{m}\sum_{j=1}^{m} \alpha_i \alpha_j y_i y_j \boldsymbol{x}_i{}^T \boldsymbol{x}_j$$

$$\text{s.t.} \quad \sum_{i=1}^{m} \alpha_i y_i = 0,$$
$$\alpha_i \geq 0, \quad i = 1,2,\dots,m$$

- Solve the dual problem , get the value of α
- Calculate ‘w’ and ‘b’ : $\boldsymbol{w} = \sum_{i=1}^{m} \alpha_i y_i \boldsymbol{x}_i$ $\qquad b = y_s - \sum_{i\epsilon t} \alpha_i y_i \boldsymbol{x}_i{}^T \boldsymbol{x}_s$
- The final model:

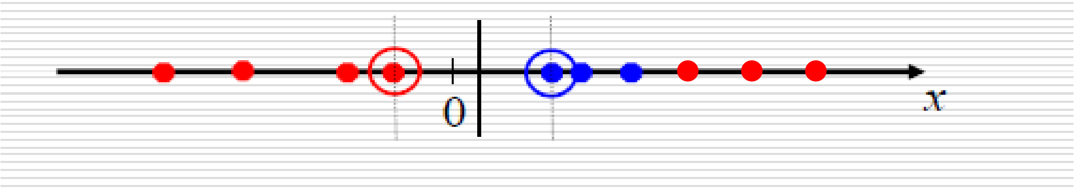
$$f(\boldsymbol{x}) = \boldsymbol{w}^T \boldsymbol{x} + b = \sum_{i=1}^{m} \alpha_i y_i \boldsymbol{x}_i{}^T \boldsymbol{x} + b = \sum_{\alpha_i \neq 0} \alpha_i y_i \boldsymbol{x}_i{}^T \boldsymbol{x} + b$$

Outline

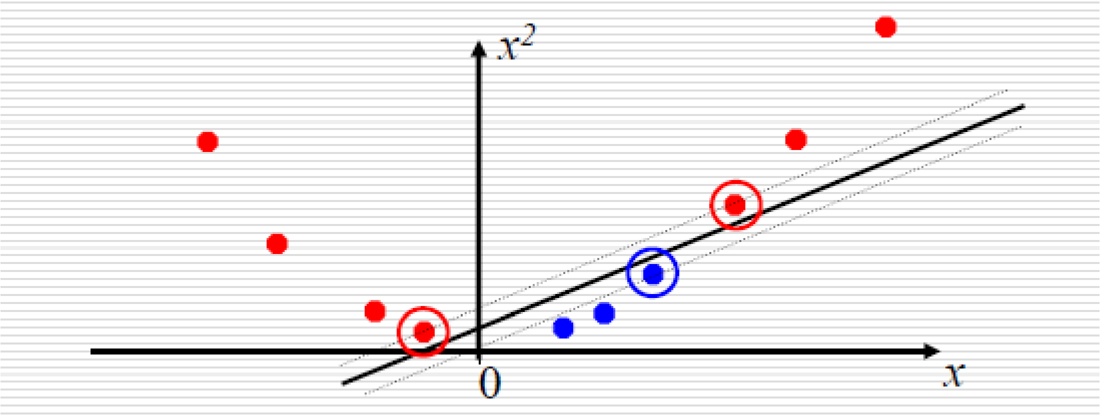
- Margin and support vector
- Dual problem
- Kernel function
- Soft margin
- Support Vector Regression
- Usage of LIBSVM with matlab

Kernel function

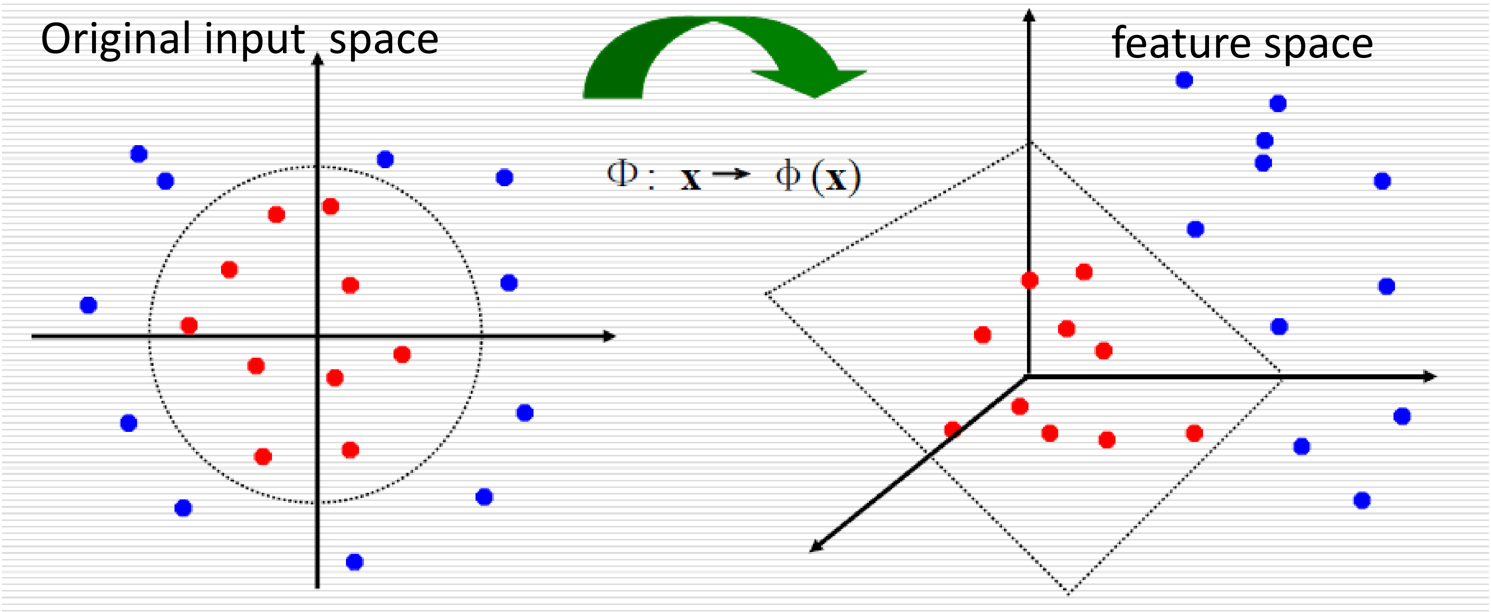
- Datasets are linearly inseparable.



- How about mapping data to a higher dimensional space :



Kernel function



- General idea : the original input space can be mapped into some higher dimensional feature space where the training set can be separable.

Kernel function

With this mapping ,our dual optimization problem becomes :

$$\max_{\alpha} \ L(\alpha) = \sum_{i=1}^{m} \alpha_i - \frac{1}{2}\sum_{i=1}^{m}\sum_{j=1}^{m} \alpha_i\alpha_j y_i y_j \boxed{\phi(\boldsymbol{x}_i)^T \phi(\boldsymbol{x}_j)} \longleftarrow \boxed{\boldsymbol{x}_i{}^T \boldsymbol{x}_j}$$

s.t. $\sum_{i=1}^{m} \alpha_i y_i = 0,$

$\alpha_i \geq 0, \ \ i = 1,2, \dots, m$

- Calculating the inner product of feature vectors in the feature space can be costly because it is high dimensional.
- The kernel trick comes to rescue:

Kernel function $\longleftarrow \boxed{k(\boldsymbol{x}_i, \boldsymbol{x}_j)} = \phi(\boldsymbol{x}_i)^T \phi(\boldsymbol{x}_j)$

the inner product of feature vectors in the feature space $\longrightarrow$ calculation in the original input space by function $k(\cdot,\cdot)$.

Kernel function

- An example:

3-dimension to 9-dimension

Suppose $x, z \in R^3$, and

$\emptyset(x) = [x_1x_1, x_1x_2, x_1x_3, x_2x_1, x_2x_2, x_2x_3, x_3x_1, x_3x_2, x_3x_3]$

Then,

$$\emptyset(x)^T\emptyset(z) = \sum_{i=1}^{3}\sum_{j=1}^{3} x_ix_j z_iz_j$$

We can use the kernel

$$K(x,z) = \sum_{i=1}^{3}\sum_{j=1}^{3} x_ix_j z_iz_j = (x_1z_1 + x_2z_2 + x_3z_3)^2 = (x^Tz)^2$$

$$K(x,z) = (x^Tz)^2 = \emptyset(x)^T\emptyset(z)$$

Kernel function

- Re-written dual optimization problem:

$$\max_{\alpha} \; L(\alpha) = \sum_{i=1}^{m} \alpha_i - \frac{1}{2}\sum_{i=1}^{m}\sum_{j=1}^{m} \alpha_i \alpha_j y_i y_j \boxed{k(\boldsymbol{x}_i, \boldsymbol{x}_j)} \leftarrow \boxed{\emptyset(\boldsymbol{x}_i)^T \emptyset(\boldsymbol{x}_j)}$$

s.t. $\sum_{i=1}^{m} \alpha_i y_i = 0,$

$\alpha_i \geq 0, \quad i = 1,2, \ldots, m$

- Resolve it and we can get：

$$f(\boldsymbol{x}) = \boldsymbol{w}^T \emptyset(\boldsymbol{x}) + b = \sum_{i=1}^{m} \alpha_i y_i \emptyset(\boldsymbol{x}_i)^T \emptyset(\boldsymbol{x}) + b = \sum_{i=1}^{m} \alpha_i y_i k\left(\boldsymbol{x}_i, \boldsymbol{x}_j\right) + b$$

Kernel function

$$k(\boldsymbol{x}_i, \boldsymbol{x}_j) = \emptyset(\boldsymbol{x}_i)^T \emptyset(\boldsymbol{x}_j)$$

- As long as we can calculate the inner product in the feature space, we do not need the mapping explicitly.

What kind of function k can be used as a kernel function?

- **Theorem (Mercer).** Let $k : R^n \times R^n \rightarrow R$ be given. Then for k to be a valid (Mercer) kernel, it is necessary and sufficient that for any data $\{\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots \boldsymbol{x}_m\}$, (m < ∞), the corresponding kernel matrix is symmetric positive semi-definite(对称 正半定).

kernel matrix：

$$\begin{bmatrix} k(\boldsymbol{x}_1, \boldsymbol{x}_1) & \cdots & k(\boldsymbol{x}_1, \boldsymbol{x}_m) \\ \vdots & \ddots & \vdots \\ k(\boldsymbol{x}_m, \boldsymbol{x}_1) & \cdots & k(\boldsymbol{x}_m, \boldsymbol{x}_m) \end{bmatrix}$$

Kernel function

- Examples of commonly-used kernel functions:

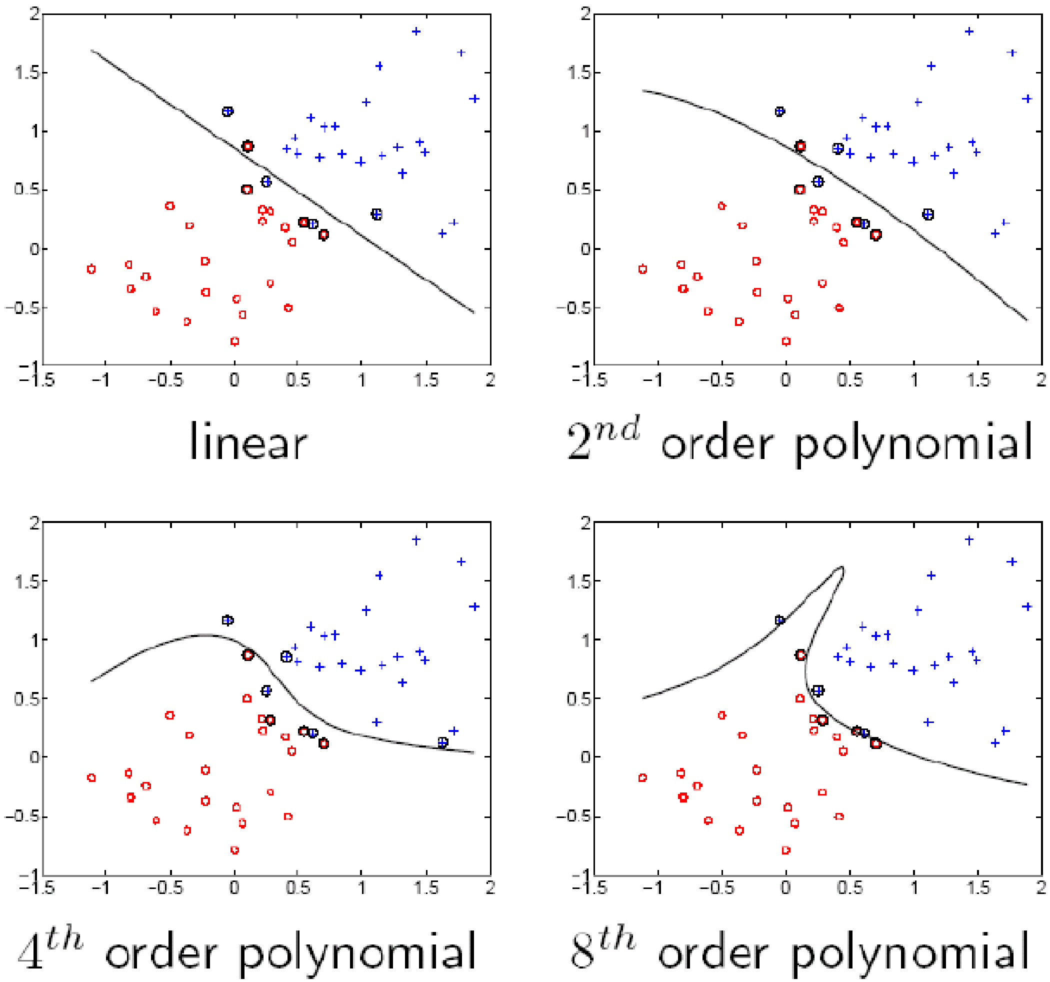
  Linear kernel: $K(x_i, x_j) = x_i^T x_j$

  Polynomial kernel: $K(x_i, x_j) = (1 + x_i^T x_j)^p$

  Gaussian (Radial-Basis Function(RBF)) kernel: $K(x_i, x_j) = \exp(-\frac{\|x_i - x_j\|^2}{2\sigma^2})$

  Sigmoid kernel: $K(x_i, x_j) = \tanh(\beta_0 x_i^T x_j + \beta_1)$

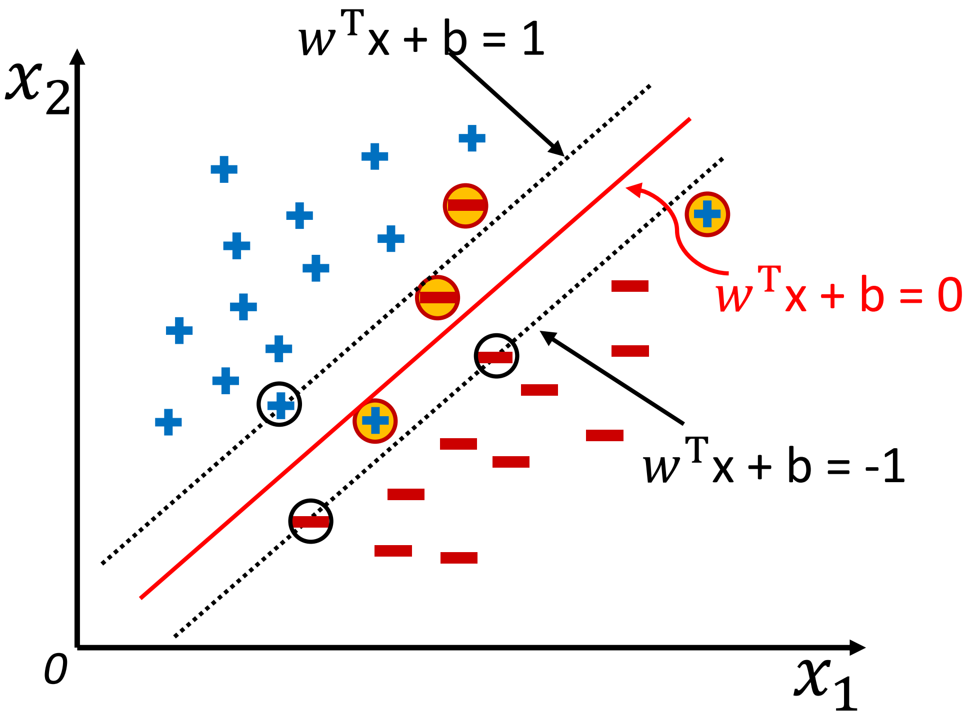
SVM examples



outline

- Margin and support vector
- Dual problem
- Kernel function
- Soft margin
- Support Vector Regression
- Usage of LIBSVM with matlab

Data with noise



- Allow some training samples don't satisfy the constraint:
  $$y_i(w^T x_i + b) \geq 1$$
- Maximize margin and minimize # training samples that don't satisfy the constraint

Optimization object

Maximize margin

Minimize # training samples that don't satisfy the constraint

$$\min_{w,b} \boxed{\frac{1}{2}\|w\|^2} + C \boxed{\sum_{i=1}^{m} l_{0/1}\left(y_i(w^T x_i + b) - 1\right)}$$

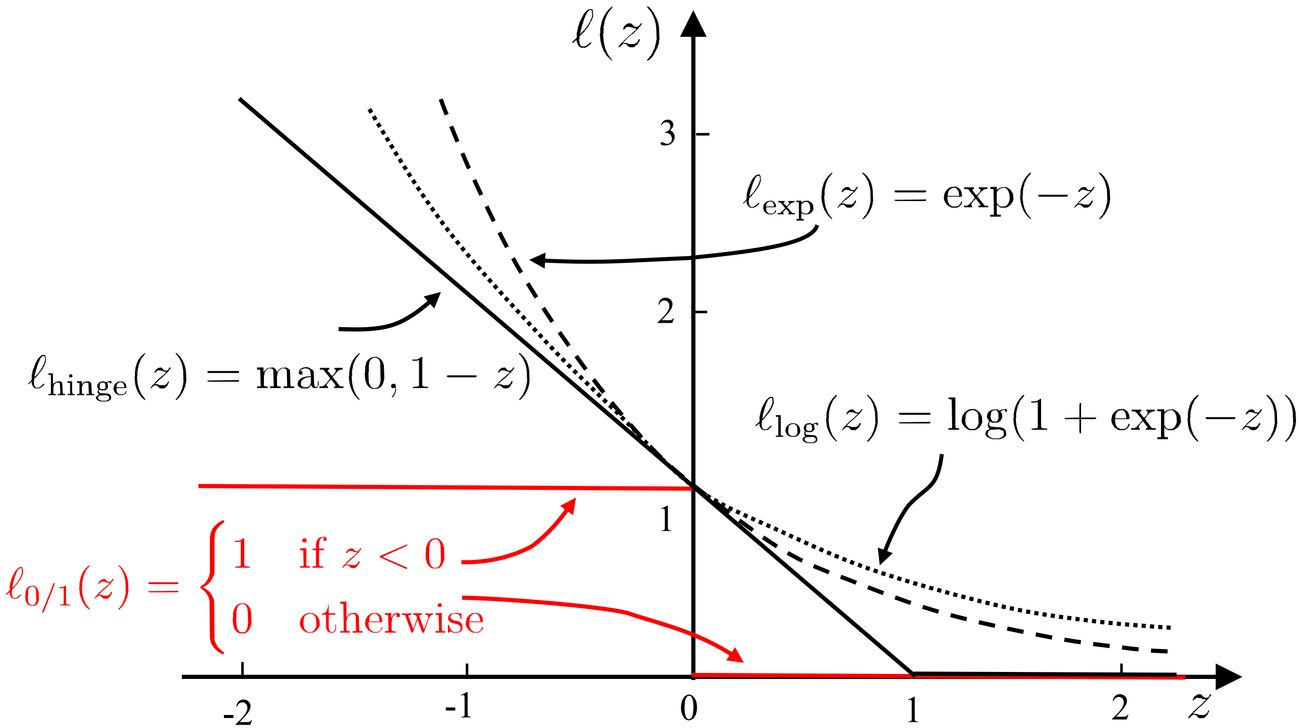
- $l_{0/1}$---loss function

$$l_{0/1}(z) = \begin{cases} 1, if\ z < 0; \\ 0, \text{otherwise.} \end{cases}$$

$C > 0$, tradeoff parameter; $C = \infty$ ⟶ hard margin SVM

But $l_{0/1}$ is not convex, discontinuous (非凸、非连续)

Surrogate loss function(替代损失)



hinge loss: $l_{hinge}(z) = \max(0,1-z)$

exponential loss(指数损失):$l_{exp}(z) = \exp(-z)$

logistic loss(对率损失)：$l_{log}(z) = \log(1+\exp(-z))$

Soft margin SVM

If we apply hinge loss , the optimization object becomes:

$$\min_{w,b} \quad \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{m} \max\left(0,1 - y_i(w^T x_i + b)\right) \qquad (*)$$

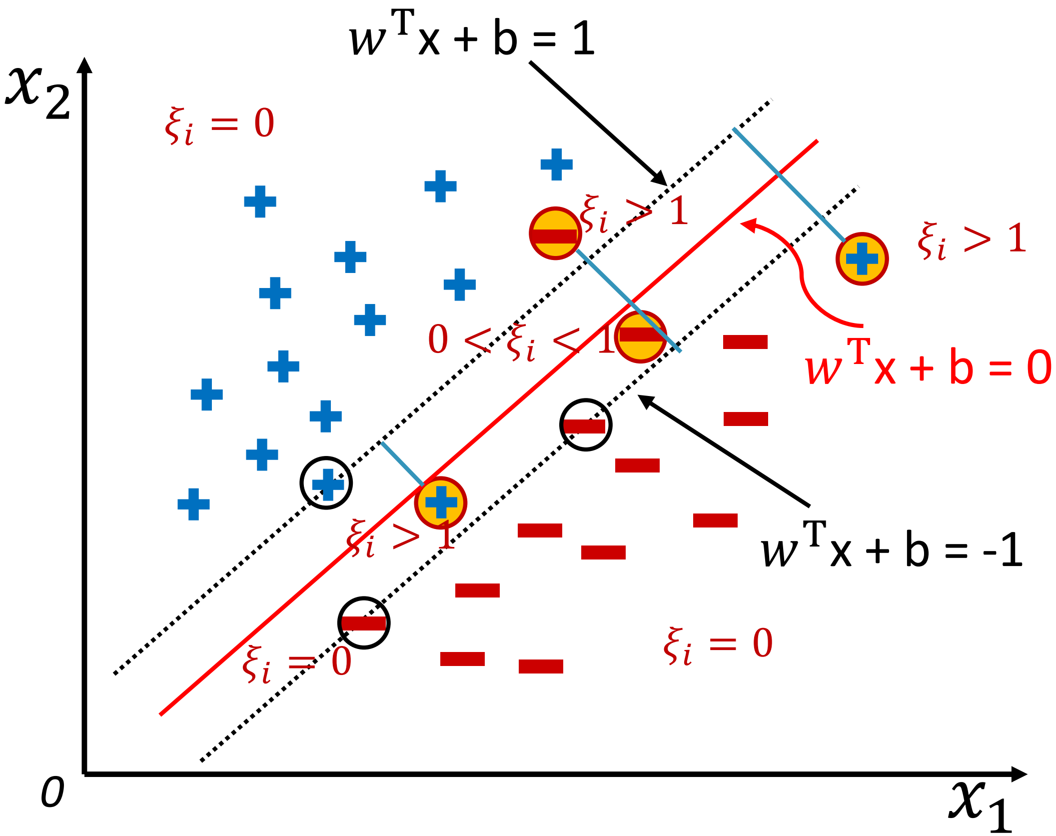
Introduce 'slack variables'$-\xi_i$, rewrite $(*)$:

Soft margin SVM

$$\boxed{\begin{aligned} &\min_{w,b,\xi_i} && \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{m} \xi_i \\ &\text{s.t.} && y_i(w^T x_i + b) \geq 1 - \xi_i \\ & && \xi_i \geq 0, i = 1,2, \dots m \end{aligned}}$$

- every sample have a ξ_i , ξ_i denotes the degree of disobeying constraints.

Soft margin SVM



Soften the constrains:

$$y_i(w^T x_i + b) \geq 1 - \xi_i$$

$$\xi_i \geq 0, i = 1,2, \ldots m$$

Penalty for misclassifying:

$$C\xi_i$$

Lagrangian multipliers method

- Lagrangian:

$$L(w,b,\alpha,\xi,\mu)=\frac{1}{2}\|w\|^2+C\sum_{i=1}^{m}\xi_i+\sum_{i=1}^{m}\alpha_i\big(1-\xi_i-y_i(w^T x_i+b)\big)-\sum_{i=1}^{m}\mu_{i\xi_i}$$

$\alpha_i \geq 0, \mu_i \geq 0$ are Lagrangian multipliers.

let L's partial derivatives with respect to w, b, ξ_i to zero, we can get :

$$w=\sum_{i=1}^{m}\alpha_i y_i x_i$$

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

$$\mathrm{c}=\alpha_i+\mu_i$$

The Optimization Problem

- The dual of this new constrained optimization problem is:

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

s.t. $\sum_{i=1}^{m} \alpha_i y_i = 0,$

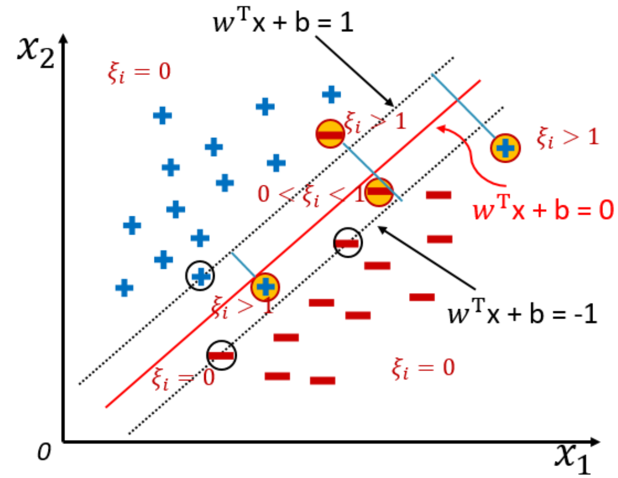
$$0 \le \alpha_i \le c, \quad i = 1,2,\dots,m$$

- 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
- Again , SMO algorithm can be used to find α_i

KKT conditions

- KKT conditions:

$$\begin{cases} \alpha_i \geq 0, \mu_i \geq 0 \\ y_i(w^T x_i + b) - 1 + \xi_i \geq 0 \\ \alpha_i(y_i f(x_i) - 1 + \xi_i) = 0 \\ \xi_i \geq 0, \mu_i \xi_i = 0 \end{cases}$$



For every sample(x_i, y_i), always have: $\alpha_i = 0$ or $y_i f(x_i) = 1 - \xi_i$

If $\alpha_i = 0$, the sample has no effect on $f(x)$

If $\alpha_i > 0$, then $y_i f(x_i) = 1 - \xi_i$, the sample is a support vector

If $\alpha_i < C$, then $\mu_i > 0$, and then $\xi_i = 0$, the sample is on the margin

If $\alpha_i = C$, then $\mu_i = 0$

If $\xi_i \leq 1$, the sample is in the margin

If $\xi_i > 1$, the sample is misclassified

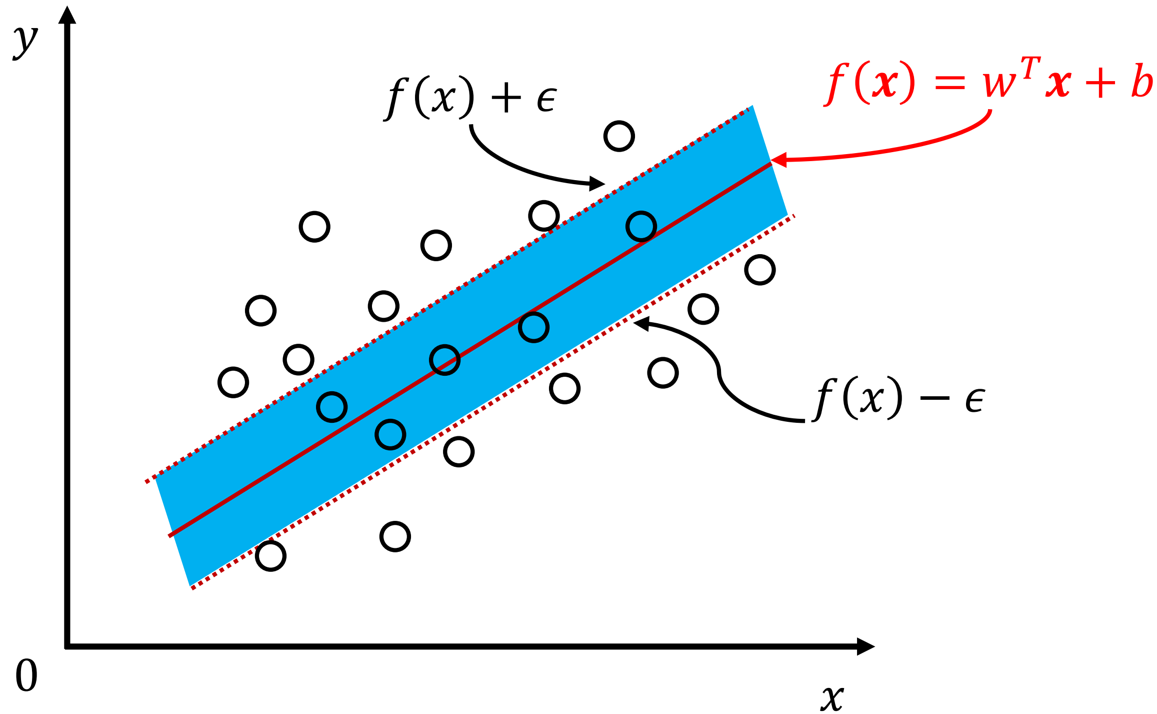
Outline

- Margin and support vector
- Dual problem
- Kernel function
- Soft margin
- Support Vector Regression
- Usage of LIBSVM with matlab

Support Vector Regression

- Data set D= $\{(\boldsymbol{x}_1, y_1),(\boldsymbol{x}_2,y_2),\ldots,(\boldsymbol{x}_m,y_m)\}$, $y_i \in R$
- We want to find a model $f(\boldsymbol{x}) = w^T\boldsymbol{x} + b$ that $f(\boldsymbol{x}) \sim$ y

Samples in the blue area
don't count loss



Formulation

$$\min_{w,b} \quad \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{m} l_\epsilon(f(x_i) - y_i) \qquad (*)$$

$$l_\epsilon(z) = \begin{cases} 0, & if\ |z| \le \epsilon \\ |z| - \epsilon, & \text{otherwise} \end{cases}$$

Introduce 'slack variables' $-\xi_i$ and $\hat{\xi}_i$, rewrite $(*)$:

The slack degree of two sides can be different

$$\min_{w,b,\xi_i,\hat{\xi}_i} \quad \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{m}(\xi_i + \hat{\xi}_i)$$

s.t.

$$f(x_i) - y_i \le \epsilon + \xi_i$$

$$y_i - f(x_i) \le \epsilon + \hat{\xi}_i$$

$$\xi_i \ge 0, \hat{\xi}_i \ge 0, i = 1,2, \ldots m$$

Resolve the problem

- Again

  lagrangian multiplier method

  dual problem

  KKT conditions

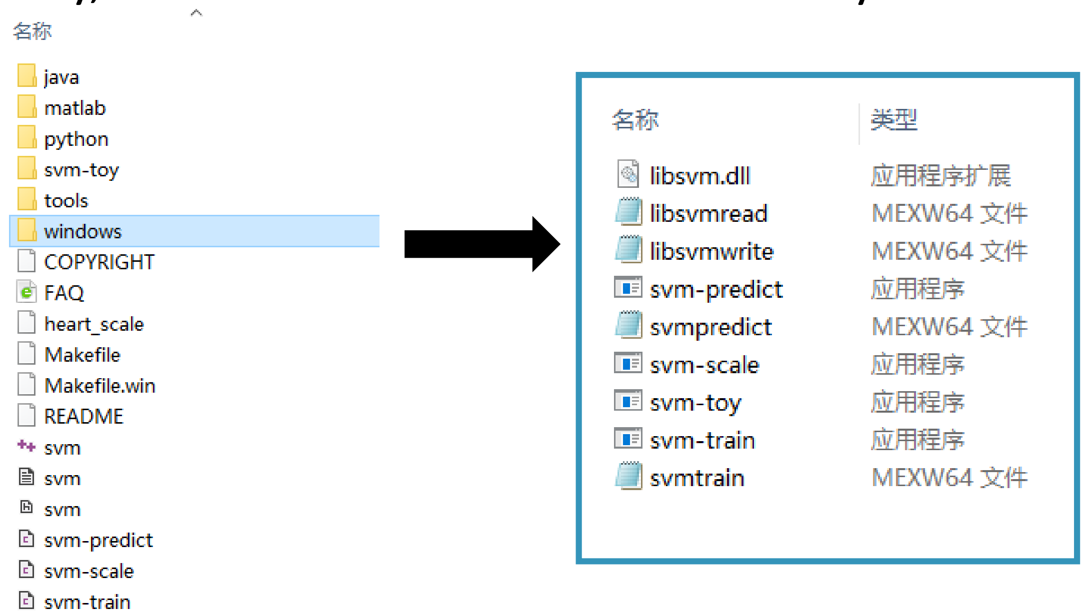
Outline

- Margin and support vector
- Dual problem
- Kernel function
- Soft margin
- Usage of LIBSVM with matlab

LIBSVM

- Download LIBSVM： https://www.csie.ntu.edu.tw/~cjlin/libsvm/

（Generally, we need “windows” file folder only.）



- Add path in matlab（设置路径—添加并包含子文件夹—libsvm-3.22 ）

Now, we can use LIBSVM to train our model.

A toy example

100 images: 60 for training, 40 for testing

120 images: 70 for training, 50 for testing

- Data preparation

  data_train.mat: 130*n, the first 60 rows are cat samples; n is feature dimension
  data_test.mat: 90*n, the first 40 rows are cat samples, the rest are dog samples
  label_train.mat: 130*1, first 60 are 1 which indicate cat category, the rest are -1
  label_test.mat: 90*1, first 40 are 1 which indicate cat category, the rest are -1

- Model training

  参数具体含义见libsvm官网

  model = **svmtrain**(label_train, data_train, '-s 0 -t 1 -c 10 -d 3')

- Model prediction/testing

  [predict_label , accuracy, dec_values] = **svmpredict**(label_test, data_test)

Take Home Message

- ‘Maximum margin’ idea
- Dual problem
- The KKT condition and support vectors
- Feature mapping—deal with linearly inseparable problem ; kernel function
- Soft margin—deal with data with noise

Resources

- http://www.csie.ntu.edu.tw/~htlin/mooc 台湾大学林轩田
- http://www.powercam.cc/home.php?user=chli&f=slide&v=list&fid=4097(李政轩)——kernel method & SVM
- http://cs229.stanford.edu/materials.html ——Andrew Ng's Lecture Notes(斯坦福机器学习课程笔记).
- 一些博客：
    - http://blog.pluskid.org/?page_id=683
    - http://blog.csdn.net/v_july_v/article/details/7624837
- https://www.csie.ntu.edu.tw/~cjlin/libsvm/——LIBSVM
- https://www.csie.ntu.edu.tw/~cjlin/liblinear/——LIBLINEAR -- A Library for Large Linear Classification

Thanks !