Chapter 6 Kernel method

We considered linear parametric models in which of the form of mapping $y(x, w)$ is governed by w of adaptive (trainable) parameters

Training set is used either to find a point estimate of w or to determine a posterior distribution of w

There is a class of ML in which the training data points are kept and used during the prediction phase

E.g. K-NN and Parzen probability density model

They require a metric to be defined that measures the similarity of any two vectors in input space

Linear parametric models can be written in a dual representation. In this form, predictions are made using linear combination of kernel functions, which are calculated using the training data points

This works well when the model uses a fixed non-linear mapping to transform the input data into a

feature space

$$K(x, x') = \Phi(x)^T \Phi(x')$$

The simplest kernel function comes from using the identity mapping for the feature transformation $\Phi(x) = x$

In this case,

$$K(x, x') = x^T x'$$

This called the linear kernel.

Kernel trick (kernel substitution)

When an algorithm uses input vectors only through dot products, replace the dot product with a kernel function to make it work in a nonlinear feature space

Dual representation

Consider a linear regression model with regularized SSE error function given by

$$J(\mathbf{w}) := \frac{1}{2}\sum_{n=1}^{N} \{\mathbf{w}^T \boldsymbol{\Phi}(\mathbf{x}_n) - t_n\}^2 + \frac{\lambda}{2}\mathbf{w}^T\mathbf{w} \qquad (6.2)$$

where $\lambda \geq 0$.

Set $\nabla_{\mathbf{w}} J(\mathbf{w}) = \mathbf{0}$. Then we see that the solution for $\mathbf{w}$ takes the form of a linear combination of the vectors $\boldsymbol{\Phi}(\mathbf{x}_n)$ with coefficients that are function of $\mathbf{w}$

$$\mathbf{w} = -\frac{1}{\lambda}\sum_{n=1}^{N} \{\mathbf{w}^T \boldsymbol{\Phi}(\mathbf{x}_n) - t_n\} \boldsymbol{\Phi}(\mathbf{x}_n) = \sum_{n=1}^{N} a_n \boldsymbol{\Phi}(\mathbf{x}_n) = \underset{M\times N}{\boldsymbol{\Phi}^T} \overset{N\times 1}{\mathbf{a}} \qquad (6.3)$$

where Φ is the design matrix and $a = (a_1, \ldots, a_N)^T$ with

$$a_n = -\frac{1}{\lambda}\{ w^T \phi(x_n) - t_n \}$$

we will reformulate the least squares algorithm in terms of the parameter vector a, which leads to a dual representation of the algorithm

Substitute $w = \Phi^T a$ into $J(w)$ (6.2)

$$J(a) = \frac{1}{2} a^T \Phi \Phi^T \Phi \Phi^T a - a^T \Phi \Phi^T t + \frac{1}{2} t^T t + \frac{\lambda}{2} a^T \Phi \Phi^T a$$

where $t = (t_1, t_2, \ldots t_N)^T$.

Now we define the gram matrix $K = \Phi\Phi^T$ which is an $N \times N$ symmetric matrix with elements

$$K_{nm} = \Phi(x_n)^T \Phi(x_m) = K(x_n, x_m)$$

In terms of the gram matrix, SSE error can be written as

$$J(a) = \frac{1}{2} a^T K K a - a^T K t + \frac{1}{2} t^T t + \frac{\lambda}{2} a^T K a$$

Using (6.3) to eliminate w/ from (6.4) and solving for a we obtain

$$a = (K + \lambda I_N)^{-1} t$$

For linear regression model, we obtain the following

prediction for a new input x

$$y(x) = w^T \phi(x) = a^T \Phi \phi(x) = k(x)^T \overbrace{(K + \lambda I_N)^{-1}}^{a} t \qquad (6.9)$$

where $k(x)$ with $k_n(x) = k(x_n, x) = \phi(x_n)^T \phi(x)$

for $n = 1, 2, \ldots N$

The dual formulation allows the solution to the least

square problem to be expressed by $k(x, x')$

Thus we see that the formulation allows the solution

to the least square problem to be expressed entirely in

terms of the kernel function $k(x, x')$ (dual formulation)

a can be written as a linear combination of the feature vector $\phi(x)$. So we can recover the original parameter vector w.

Remark

- To determine w, we need $O(M^3)$ calculations
- In the dual formulation, to determine a, we need $O(N^3)$
- The dual formulation uses only the kernel function,

  i.e. we do not need to compute the feature vector

  $\Phi(x)$ directly
- We can work in high (even infinite) dimensional space

  without extra cost (all through the kernel)

기존 basis function 방법

1. Fix a basis function $\Phi(x) = (\phi_0(x), \phi_1(x), \dots \phi_{M-1}(x))^T$

2. Model $y(x) = w^T \Phi(x)$ with weight vector w

3. Finding w minimizing the cost function below

$$J(w) = \frac{1}{2} \sum_{n=1}^{N} \{ w^T \Phi(x_n) - t_n \}^2 + \frac{\lambda}{2} w^T w$$

where $\lambda \geq 0$.

kernel trick (dual representation)

1. Fix a kernel function $k(x, x')$

2. Model $y(x) = \mathbf{k}(x)^T a$ where $\mathbf{k}(x) = (k_1(x), k_2(x), \ldots k_N(x))^T$

and $k_1(x) = k(x_1, x)$

3. Finding a minimizing the cost function below

$$J(a) = \frac{1}{2} a^T K K a - a^T K t + \frac{1}{2} t^T t + \frac{\lambda}{2} a^T K a$$

where $k_{nm} = k(x_n, x_m)$, $\lambda \geq 0$

6.2 Constructing kernels

In order to use kernel substitution, we need to construct valid kernel functions.

One approach is to choose a feature space mapping $\Phi(x)$ and then use this to find the corresponding kernel

$$k(x, x') := \Phi(x)^T \Phi(x') = \sum_{i=1}^{M} \phi_i(x) \phi_i(x')$$

where $\phi_i(x)$ are basis functions.

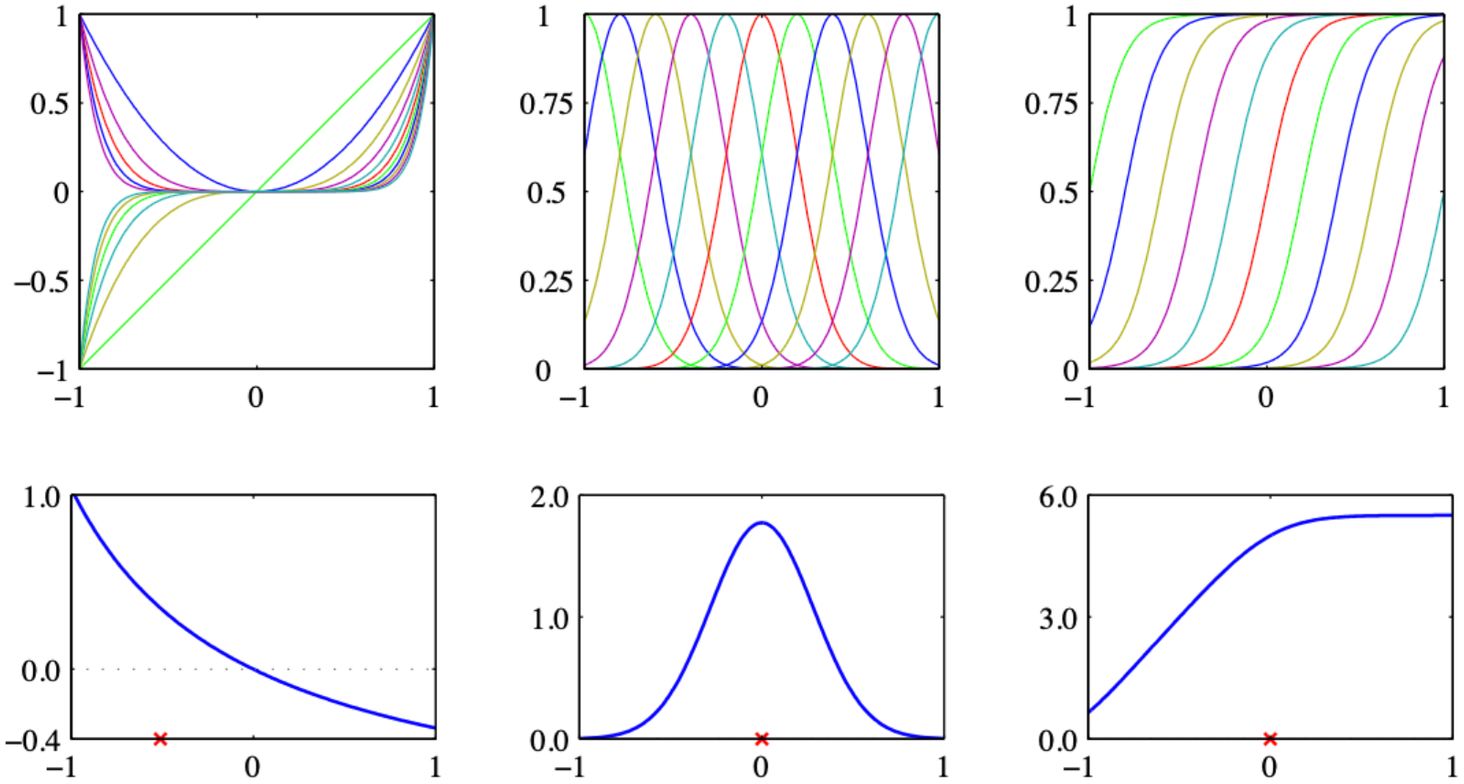


Figure 6.1 Illustration of the construction of kernel functions starting from a corresponding set of basis functions. In each column the lower plot shows the kernel function $k(x, x')$ defined by (6.10) plotted as a function of x, where x' is given by the red cross ($\times$), while the upper plot shows the corresponding basis functions given by polynomials (left column), 'Gaussians' (centre column), and logistic sigmoids (right column).

Alternative approach is to build kernel function directly without thinking about the feature mapping $\phi(x)$

However, to be a valid kernel, the function must satisfy certain conditions: It should correspond to a dot product in some feature space (even infinite dimensional)

Mathematically, this means the kernel matrix k where $k_{nm} = k(x_n, x_m)$ must be

- symmetric $k_{nm} = k_{mn}$
- Positive semi-definite

E.g. Consider the kernel function given by

$$K(x, z) := (x^T z)^2$$

In case of $D = 2$, we can expand out the terms and identify the corresponding nonlinear feature mapping

$$K(x, z) = (x^T z)^2 = (x_1 z_1 + x_2 z_2)^2$$

$$= x_1^2 z_1^2 + 2 x_1 z_1 x_2 z_2 + x_2^2 z_2^2$$

$$= (x_1^2, \sqrt{2} x_1 x_2, x_2^2)(z_1^2, \sqrt{2} z_1 z_2, z_2^2)^T$$

$$= \Phi(x)^T \Phi(z)$$

Feature mapping $\Phi(x) = (x_1^2, \sqrt{2} x_1 x_2, x_2^2)^T$

Here are the main properties that allow combining simpler kernels to build more powerful one:

Techniques for Constructing New Kernels.

Given valid kernels $k_1(\mathbf{x}, \mathbf{x}')$ and $k_2(\mathbf{x}, \mathbf{x}')$, the following new kernels will also be valid:

$$k(\mathbf{x}, \mathbf{x}') = ck_1(\mathbf{x}, \mathbf{x}') \tag{6.13}$$
$$k(\mathbf{x}, \mathbf{x}') = f(\mathbf{x})k_1(\mathbf{x}, \mathbf{x}')f(\mathbf{x}') \tag{6.14}$$
$$k(\mathbf{x}, \mathbf{x}') = q\left(k_1(\mathbf{x}, \mathbf{x}')\right) \tag{6.15}$$
$$k(\mathbf{x}, \mathbf{x}') = \exp\left(k_1(\mathbf{x}, \mathbf{x}')\right) \tag{6.16}$$
$$k(\mathbf{x}, \mathbf{x}') = k_1(\mathbf{x}, \mathbf{x}') + k_2(\mathbf{x}, \mathbf{x}') \tag{6.17}$$
$$k(\mathbf{x}, \mathbf{x}') = k_1(\mathbf{x}, \mathbf{x}')k_2(\mathbf{x}, \mathbf{x}') \tag{6.18}$$
$$k(\mathbf{x}, \mathbf{x}') = k_3\left(\boldsymbol{\phi}(\mathbf{x}), \boldsymbol{\phi}(\mathbf{x}')\right) \tag{6.19}$$
$$k(\mathbf{x}, \mathbf{x}') = \mathbf{x}^{\mathrm{T}}\mathbf{A}\mathbf{x}' \tag{6.20}$$
$$k(\mathbf{x}, \mathbf{x}') = k_a(\mathbf{x}_a, \mathbf{x}'_a) + k_b(\mathbf{x}_b, \mathbf{x}'_b) \tag{6.21}$$
$$k(\mathbf{x}, \mathbf{x}') = k_a(\mathbf{x}_a, \mathbf{x}'_a)k_b(\mathbf{x}_b, \mathbf{x}'_b) \tag{6.22}$$

where $c > 0$ is a constant, $f(\cdot)$ is any function, $q(\cdot)$ is a polynomial with nonnegative coefficients, $\boldsymbol{\phi}(\mathbf{x})$ is a function from $\mathbf{x}$ to $\mathbb{R}^M$, $k_3(\cdot, \cdot)$ is a valid kernel in $\mathbb{R}^M$, $\mathbf{A}$ is a symmetric positive semidefinite matrix, $\mathbf{x}_a$ and $\mathbf{x}_b$ are variables (not necessarily disjoint) with $\mathbf{x} = (\mathbf{x}_a, \mathbf{x}_b)$, and k_a and k_b are valid kernel functions over their respective spaces.

Polynomial kernel (example)

$$k(x, x') = (x^T x')^2$$

includes only terms of degree 2.

If we add a constant before squaring like

$$K(\mathbf{x}, \mathbf{x}') = (\mathbf{x}^T \mathbf{x}' + c)^2 \quad \text{where} \quad c > 0$$

then the feature space becomes richer.

This kernel includes a constant (bias), linear terms (x_1, x_2)

and degree 2 terms ($x_1 x_2$ or x_1^2)

$$K(\mathbf{x}, \mathbf{x}') = (x_1 x_1' + x_2 x_2' + c)^2 = (x_1 x_1')^2 + 2 x_1 x_1' x_2 x_2' + (x_2 x_2')^2$$

$$+ 2c\, x_1 x_1' + 2c\, x_2 x_2' + c^2$$

$$\mathbf{x} = \begin{pmatrix} x_1 \\ x_2 \end{pmatrix}$$

$$= \Phi(\mathbf{x})^T \Phi(\mathbf{x}')$$

$$\Phi(\mathbf{x}) = (\sqrt{2c}\, x_1,\ \sqrt{2c}\, x_2,\ x_1^2,\ \sqrt{2}\, x_1 x_2,\ x_2^2,\ c)^T$$

In general,

$$k(x, x') = (x^T x')^M$$

gives a feature space with all monomials up to degree M

One commonly used kernel is the Gaussian kernel (a.k.a RBF kernel).

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

This kernel measures how close two vectors x and x' are.

$K(x, x')$ is close to 1 if x and x' are similar

$K(x, x')$ is close to 0 if x and x' are far apart

Although it looks like a Gaussian probability density, here it is not used as a probability

RBF kernel is a valid kernel

$$\|x - x'\|^2 = x^T x + (x')^T x' - 2 x^T x'$$

So

$$k(x, x') = \exp\left(-\frac{x^T x}{2\sigma^2}\right) \exp\left(\frac{x^T x'}{\sigma^2}\right) \exp\left(-\frac{(x')^T x'}{2\sigma^2}\right)$$

Because of (6.14) and (6.16), together with the validity of the linear kernel $k(x, x') = x^T x'$

Remark

- The Gaussian kernel corresponds to an infinite-dimensional feature mapping $\phi(x)$ (exercise 6.11)
- The Gaussian kernel is not restricted to the use of Euclidean distance. I.e. $x^T x'$ can be replaced with any nonlinear kernel $\tilde{k}(x, x')$.

$$k(x, x') = \exp\left\{-\frac{1}{2\sigma^2}\left(\tilde{k}(x, x) + \tilde{k}(x', x') - 2\tilde{k}(x, x')\right)\right\}$$

One useful way to build kernel is by using a probabilistic generative model

Suppose we have a generative model $p(x)$. Then we can define a kernel as

$$k(x, x') = p(x)\, p(x')$$

This is a valid because it can be written as an inner product:

$$k(x, x') = \phi(x)^T \phi(x) \quad \text{where} \quad \phi(x) = p(x)$$

Note that it is a very simple feature mapping, where each input x is just mapped to a scalar value $P(x)$.

In practice, more advanced versions use likelihood function or posterior distributions as features

It says that x and x' are similar if they both have high probabilities

Example: kernel from a Gaussian Mixture Model

Assume $$p(x) = \sum_{k=1}^{K} \pi_k \, N(x \mid \mu_k, \Sigma_k)$$

$$p(z=k) \quad p(x \mid z=k)$$

and estimates π_k, μ_k, Σ_k for $k = 1, 2, \ldots K$

For each x, calculate the posterior responsibility

$$\phi_k(x) = p(z=k \mid x) = \frac{\pi_k \, N(x \mid \mu_k, \Sigma_k)}{\sum_{j=1}^{K} \pi_j \, N(x \mid \mu_j, \Sigma_j)}$$

Define $\Phi(x) := (\phi_1(x), \phi_2(x), \ldots \phi_K(x))^T$ and

$$K(x, x') = \Phi(x)^T \Phi(x')$$

This kernel measures the similarity between two vectors based on how responsible each Gaussian component is for generating them.

Kernels from mixture models with latent variable

We can build more flexible kernels by summing over multiple components in a probabilistic mixture model

1. Discrete case

Suppose we have

- latent variable $i \in \{1, 2, \dots K\}$
- prob. distribution for each component $p(x \mid i)$
- prior over components $p(i)$

The kernel $k(x, x') = \sum_{k=1}^{K} p(x \mid i)\, p(x' \mid i)\, p(i)$

Remark

- This kernel is large when x and x' both have high likelihood under the same components
- It reflects how similarly the model explains the two inputs
- The latent variable i can be seen as a latent variable (e.g., cluster index in a mixture model)

2. Continuous case

If the latent variable z is continuous, we replace the sum with an integral

$$K(x, x') = \int p(x|z)\, p(x'|z)\, p(z)\, dz$$

where z is a continuous latent variable.

Another example of a kernel function is the sigmoidal kernel

$$k(x, x') = \tanh(a\, x^T x' + b)$$

Remark

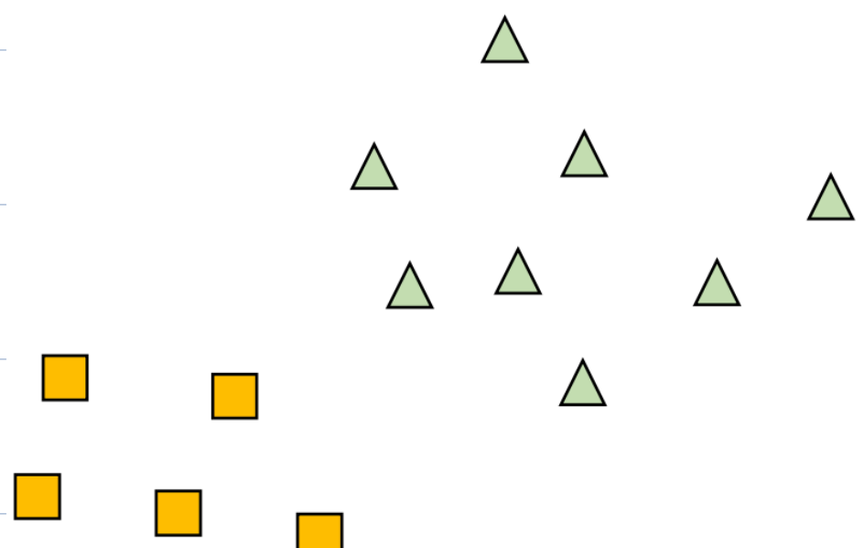
- This function looks like the activation function in neural network
- This kernel is not always valid (its gram matrix is not guaranteed to be positive semidefinite for all a, b and data set).

Support vector machine

Two class classification problem

Linearly separable data set. $x \in \mathbb{R}^D$, $t = -1$ or 1

There are infinitely many decision boundaries. What is the best decision boundary ?



SVM

$w^T x + b$

$$\min_{w} \; w^T w$$

quadratic objective

$$\text{s.t.} \quad t_i (w^T x_i + b) \geq 1 \quad \forall_{i=1 \ldots N}$$

linear constraints

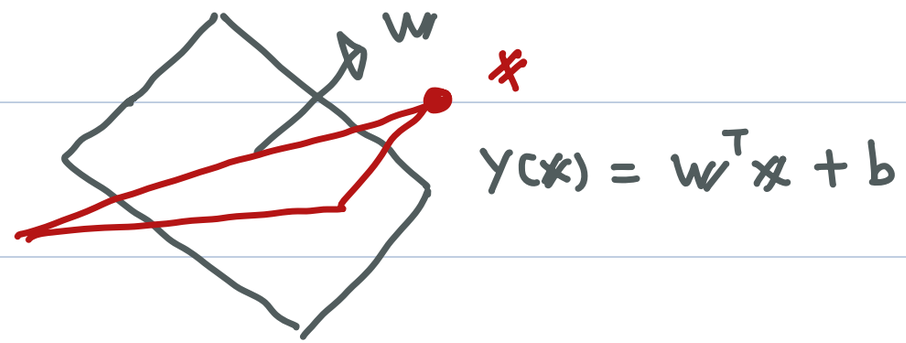
This is a quadratic optimization problem.

The one method that maximizes the distance to the closest data points from both classes

- Maximal margin

Decision boundary

$y(x) = w^T x + b$ is a hyperplain



$$x = x_p + r\frac{w}{\|w\|}$$

depends on x

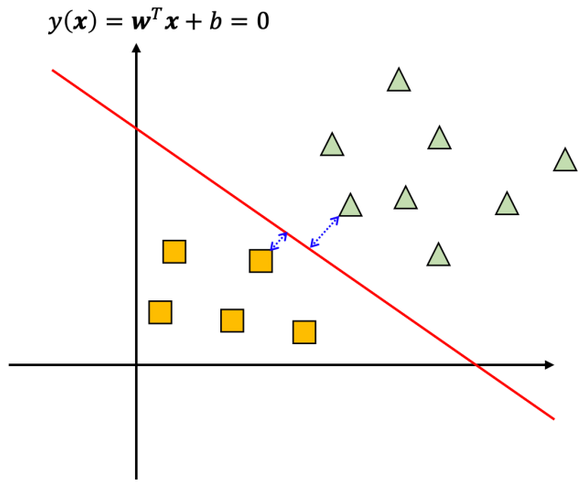
$$y(x) = w^T x + b = w^T x_p + b + r\frac{w^T w}{\|w\|} = r\|w\|$$

Define a function $r : \mathbb{R}^D \longrightarrow \mathbb{R}$ as

$$r(x) = \frac{y(x)}{\|w\|}$$

signed distance of x from the decision surface

Margin : The smallest distance between the decision boundary and any data points



w, b 으로 결정

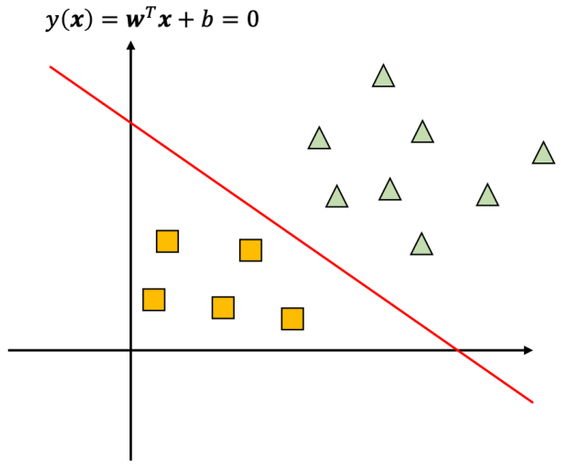
$$\gamma(w, b) = \min_{x \in D} |r(x)| = \min_{x \in D} \frac{|y(x)|}{\|w\|} = \min_{x \in D} \frac{|w^T x + b|}{\|w\|}$$

Remark : $\gamma(w, b)$ is scale invariant. I.e.,

$$\gamma(\alpha w, \alpha b) = \gamma(w, b) \quad \forall \alpha \neq 0$$

SVM : maximize margin

$$\max_{w,b} r(w,b) = \max_{w,b} \min_{x \in D} \frac{|w^T x + b|}{\|w\|}$$



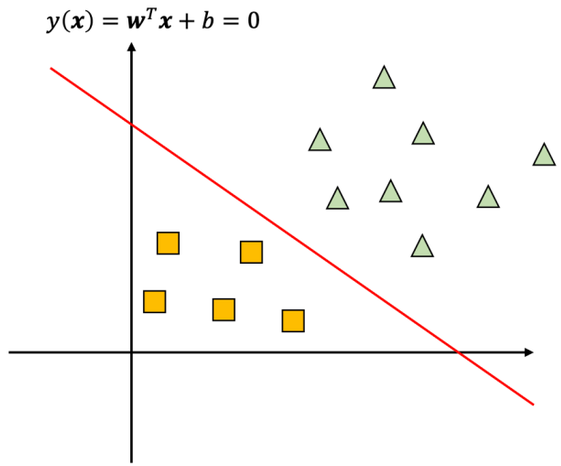
We need constraints

$$t_i (w^T x_i + b) \geq 0 \quad \forall i = 1, 2, \ldots N$$

I.e., max margin classifier with constraints

$$\max_{w,b} \; r(w,b) = \max_{w,b} \; \min_{x \in D} \; \frac{|w^T x + b|}{\|w\|}$$

s.t. $\quad t_i (w^T x_i + b) \geq 0 \qquad \forall i = 1, 2, \ldots N$



Quite complicated. Need to simplify the problem

1. $\|w\|$ does not depend on x

$$\Rightarrow \max_{w,b} \min_{x \in D} \frac{|w^T x + b|}{\|w\|} = \max_{w,b} \frac{1}{\|w\|} \left(\min_{x \in D} |w^T x + b| \right)$$

2. $\gamma(w, b)$ is scale invariant

$\Rightarrow$ so we can add a constraint

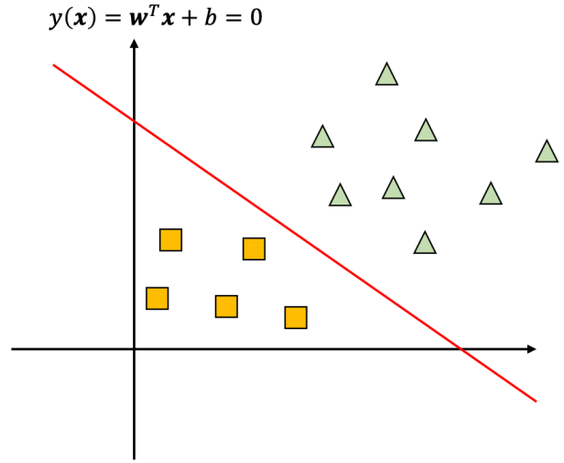
$$\min_{x \in D} |w^T x + b| = 1$$

3. 'max' can be replaced by 'min'

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

Max margin classifier

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



s.t. $t_i (w^T x_i + b) \geq 0 \quad \forall i = 1, \ldots N$

$$\min_i |w^T x_i + b| = 1$$

We need to simplify the constraints

Max margin classifier

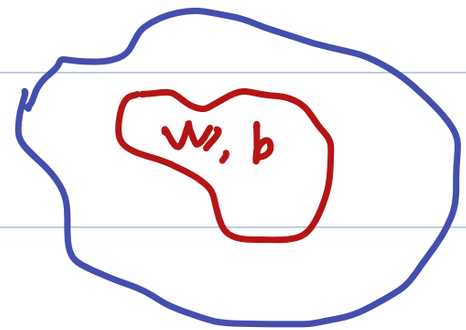
$$\min_{w,b} \frac{1}{2}\|w\|^2 \quad \text{s.t.} \quad t_i(w^T x_i + b) \geq 0 \quad \forall i = 1, \dots N$$

$$\min_i |w^T x_i + b| = 1 \quad (1)$$

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

constraints 를 만족하는

w, b 집합



Assume w_*, b_* is the optimal solution and

$$\min_i \; t_i (w_*^T x_i + b_*) = \alpha > 1$$

Let $\tilde{w} = \frac{1}{\alpha} w_*$ and $\tilde{b} = \frac{1}{\alpha} b_*$. Then $(\tilde{w}, \tilde{b})$ is

a feasible solution i.e. $t_i (\tilde{w}^T x_i + \tilde{b}) \geq 1 \quad \forall i = 1, \ldots N.$

and $\|\tilde{w}\| = \frac{1}{\alpha} \|w_*\| < \|w_*\|$

It contradicts with the assumption that is w_* optimal.

$\Rightarrow$ Optimal solution must have $t_i (w^T x_i + b) = 1$ for some i

SVM

$$\min_{w} \frac{1}{2} w^T w$$ quadratic objective

s.t. $t_i (w^T x_i + b) \geq 1 \quad \forall i = 1 \dots N$ linear constraints

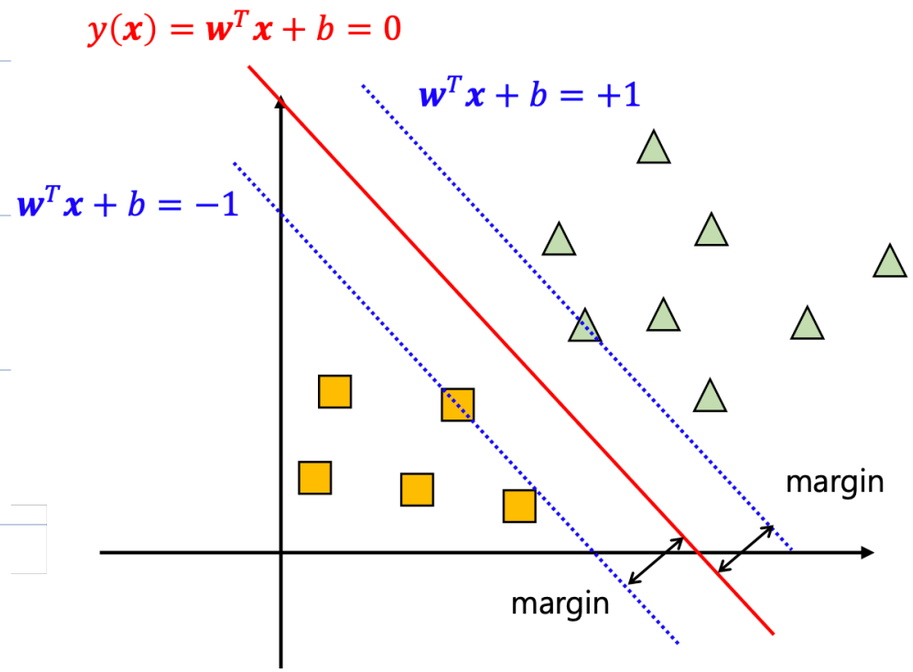
Convex problem

Support vector

- $t_i (w^T x_i + b) = 1$.

Decision boundary

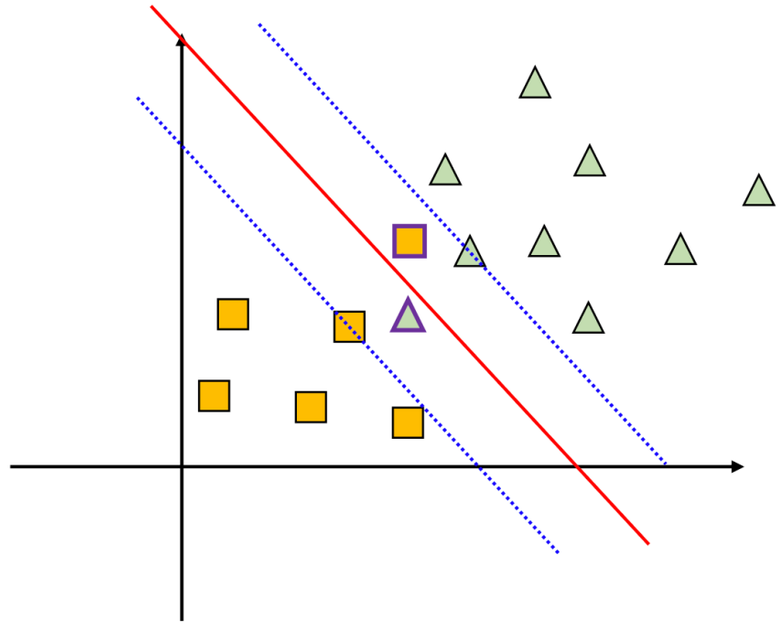
- lies in the middle of the region separating data points



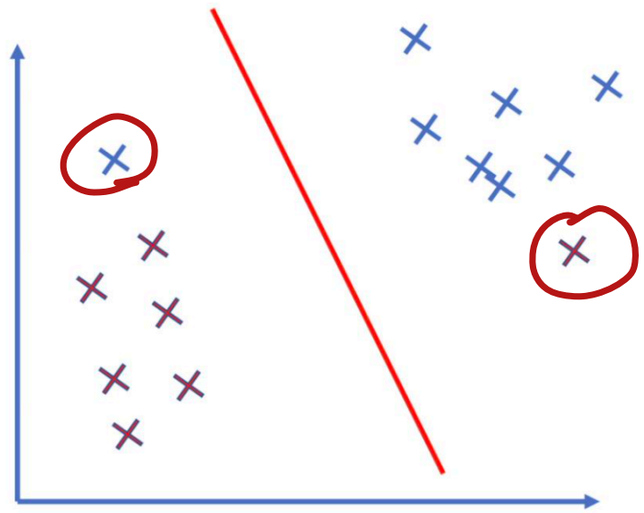
Support vector machine

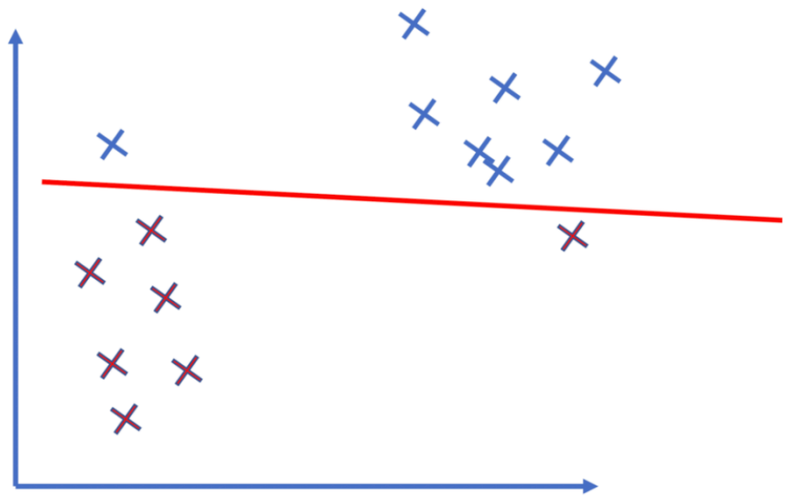
Overlapping classes

- Non - linearly separable
- Soft constraints
- Soft margin



Which one is better?





It looks OK.

But train error is not zero.

Train erro is zero.

But over-fitted

$$t_i(w^T x_i + b) \neq 1 \quad \forall_i$$

SVM

$$\min_{w} \frac{1}{2} \|w\|^2$$

$$\text{s.t.} \quad t_i (w^T x_i + b) \geq 1 \qquad \forall i = 1 \dots N$$

$$\Downarrow$$

$$\min_{w, \xi} \frac{1}{2} \|w\|^2 + C \cdot \sum_{i=1}^{N} \xi_i$$

$$\text{s.t.} \quad t_i (w^T x_i + b) \geq 1 - \xi_i \qquad \forall i = 1 \dots N$$

$$\xi_i \geq 0$$

Slack variable ξ

- Data points are allowed to be on 'wrong side' of the margin boundary but with a penalty that increases with the distance from that boundary

$$\min_{w,\xi} \frac{1}{2}\|w\|^2 + C \cdot \sum_{i=1}^{N} \xi_i$$

$$\text{s.t.} \quad t_i (w^T x_i + b) \geq 1 - \xi_i \qquad \forall_{i=1 \ldots N}$$

$$\xi_i \geq 0$$

$\Phi(x)^T \Phi(x')$ or $x^T x'$

$\Downarrow$

$K(x, x')$

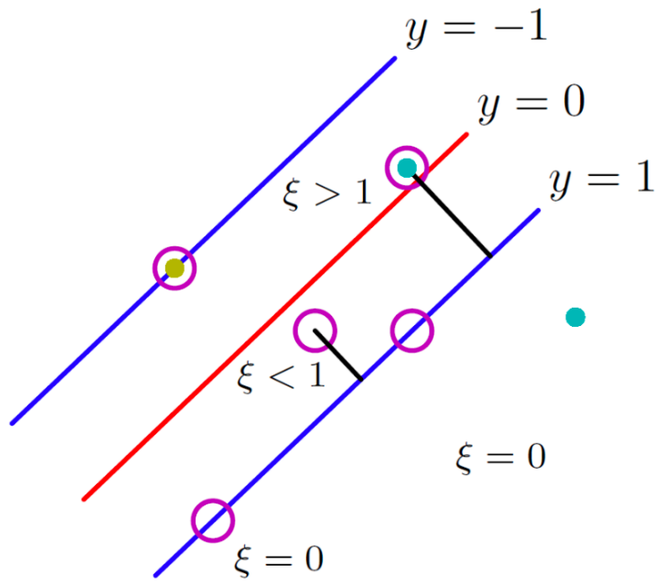
$\xi_i = 0$: correctly classified

$0 < \xi_i \leq 1$: inside the margin

but on the correct side

$\xi_i > 1$: on the wrong side of the

decision boundary



$$\min_{w,\xi} \frac{1}{2}\|w\|^2 + C \cdot \sum_{i=1}^{N} \xi_i$$

s.t. $t_i(w^T x_i + b) \geq 1 - \xi_i$

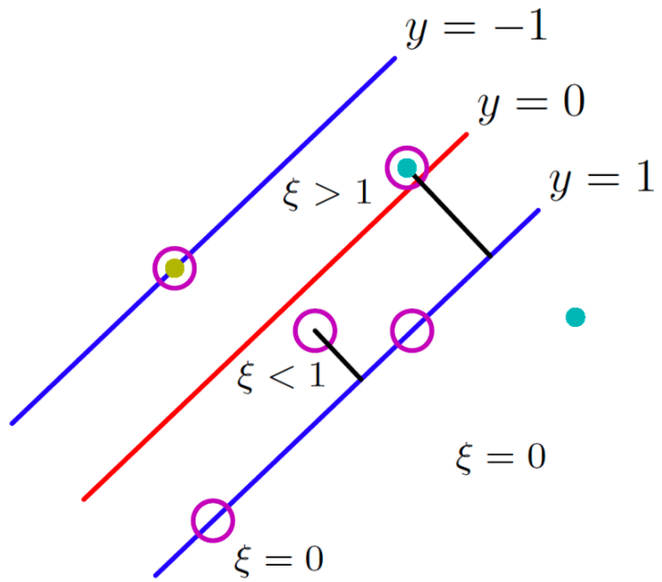
$\forall_{i=1 \ldots N}$

$\xi_i \geq 0$

If C is large, then $\xi \to 0$.

SVM tries to classify all training data correctly

Overfitting and weak regularization



$$\min_{w, \xi} \frac{1}{2} \|w\|^2 + C \cdot \sum_{i=1}^{N} \xi_i$$

$$\text{s.t.} \quad t_i (w^T x_i + b) \geq 1 - \xi_i \qquad \forall_{i=1 \ldots N}$$

$$\xi_i \geq 0$$

Overlapping classes : Hinge loss

$$\min_{w,\xi} \frac{1}{2}\|w\|^2 + C \cdot \sum_{i=1}^{N} \xi_i$$

$x^T x' \rightarrow K(x, x')$

s.t. $t_i (w^T x_i + b) \geq 1 - \xi_i$

$\forall i = 1 \dots N$

$\xi_i \geq 0$

$\Rightarrow \quad \xi_i \geq 0 \quad$ and $\quad \xi_i \geq 1 - t_i \overbrace{(w^T x_i + b)}^{y(x_i)}$

$\Rightarrow \quad \xi_i := \max(0, 1 - t_i\, y(x_i))$

$$\min_{w, \xi} \; \frac{1}{2} \|w\|^2 + C \cdot \sum_{i=1}^{N} \xi_i$$

$\Rightarrow$

s.t. $\quad \xi_i = \max(0, 1 - t_i y(x_i)) \quad \forall i = 1, \ldots N$

where $\quad y(x_i) = w^T x_i + b$

$\Rightarrow$

$$\min_{w, \xi} \; \frac{1}{2} \|w\|^2 + C \cdot \sum_{i=1}^{N} \max(0, 1 - t_i y(x_i))$$

or

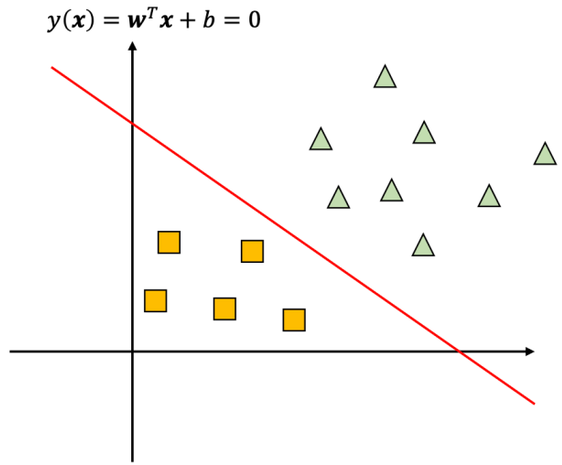
$$\min_{w, \xi} \; \sum_{i=1}^{N} \max(0, 1 - t_i y(x_i)) + \frac{1}{2C} \|w\|^2$$

where $\quad y(x_i) = w^T x_i + b$

SVM = Linear classifier with Hinge loss

$$\min_{w,b} \sum_{i=1}^{N} \max(0, 1 - t_i y(x_i)) + \frac{1}{2C} \|w\|^2$$

where $y(x_i) = w^T x_i + b$



Kernel SVM

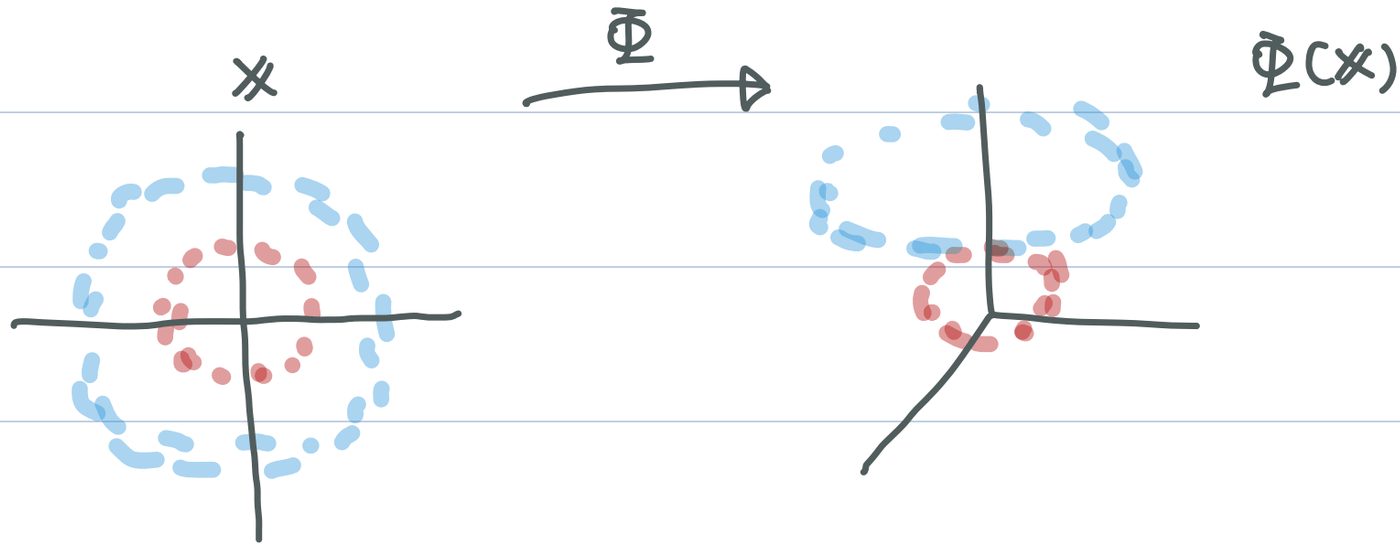
kernel method (kernel trick or technique)

Purpose : 예측, 분류 모델에 non-linearity 를 부여

(x, w 모두)

non - linearity 를 주는 다른 방법?

- feature transformation or basis function



공간 변환

Input space

Feature space

$x_1, x_2 \dots x_N$

$\Phi(x_1), \Phi(x_2) \dots \Phi(x_N)$

Kernel method

방법 (key idea)

- If an algorithm only uses dot products of vectors, then we can replace the dot product with a kernel function

$$x^T x' \longrightarrow K(x, x')$$

$$\Phi(x)^T \Phi(x')$$

유사도 또는 상관 정도 측정

How to construct a valid kernel function ?

Generalized linear regression

Linear model : $y(\mathbf{x}) = \mathbf{w}^T \mathbf{x} + b$

Basis function method

1. Fix a basis function $\Phi(\mathbf{x}) = (\phi_0(\mathbf{x}), \phi_1(\mathbf{x}), \dots \phi_{M-1}(\mathbf{x}))^T$

2. Model $y(\mathbf{x}) = \mathbf{w}^T \Phi(\mathbf{x})$ with weight vector $\mathbf{w}$

3. Finding $\mathbf{w}$ minimizing the cost function below

$$J(\mathbf{w}) = \frac{1}{2} \sum_{n=1}^{N} \{ \mathbf{w}^T \Phi(\mathbf{x}_n) - t_n \}^2 + \frac{\lambda}{2} \mathbf{w}^T \mathbf{w}$$

where $\lambda \geq 0$.

4. Inference : $y(\hat{\mathbf{x}}) = \mathbf{w}^T \Phi(\hat{\mathbf{x}})$

$$J(\mathbf{w}) = \frac{1}{2}(\boldsymbol{\Phi}\mathbf{w} - \mathbf{t})^T(\boldsymbol{\Phi}\mathbf{w} - \mathbf{t}) + \frac{\lambda}{2}\mathbf{w}^T\mathbf{w}$$

Set $\nabla_{\mathbf{w}} J(\mathbf{w}) = \mathbf{0}$. Then we see

$$\mathbf{w} = \sum_{n=1}^{N} a_n \boldsymbol{\phi}(\mathbf{x}_n) = \underset{M\times N}{\boldsymbol{\Phi}^T} \mathbf{a}^{N\times 1} \qquad (6.3)$$

where $\boldsymbol{\Phi}$ is the design matrix and $\mathbf{a} = (a_1, \ldots, a_N)^T$ with

$$a_n = -\frac{1}{\lambda}\{\mathbf{w}^T\boldsymbol{\phi}(\mathbf{x}_n) - t_n\}$$

Substitute $w = \Phi^T a$ into $J(w)$ (6.2)

$$J(a) = \frac{1}{2} a^T \Phi \Phi^T \Phi \Phi^T a - a^T \Phi \Phi^T t + \frac{1}{2} t^T t + \frac{\lambda}{2} a^T \Phi \Phi^T a$$

where $t = (t_1, t_2, \dots t_N)^T$.

$$\Phi \Phi^T = \begin{pmatrix} - \phi(x_1)^T - \\ - \phi(x_2)^T - \\ \vdots \\ - \phi(x_N)^T - \end{pmatrix} \begin{pmatrix} | & | & & | \\ \phi(x_1) & \phi(x_2) & \cdots & \phi(x_N) \\ | & | & & | \end{pmatrix} \qquad N \times N$$

$$(\Phi \Phi^T)_{ij} = \phi(x_i)^T \phi(x_j)$$

↘ replace K $N \times N$

$$K_{ij} = k(x_i, x_j)$$

kernel trick (dual representation)

1. Fix a kernel function $k(x, x')$

2. Model $y(x) = \mathbf{k}(x)^T a$ where $\mathbf{k}(x) = (k_1(x), k_2(x), \ldots k_N(x))^T$

and $k_1(x) = k(x_1, x)$

3. Finding a minimizing the cost function below

$$J(a) = \frac{1}{2} a^T K K a - a^T K t + \frac{1}{2} t^T t + \frac{\lambda}{2} a^T K a$$

where $k_{nm} = k(x_n, x_m)$, $\lambda \geq 0$

4. Inference $y(\hat{x}) = \mathbf{k}(\hat{x})^T a = \sum_{i=1}^{N} a_i k(x_i, \hat{x})$

kernel support vector machine

Primial problem

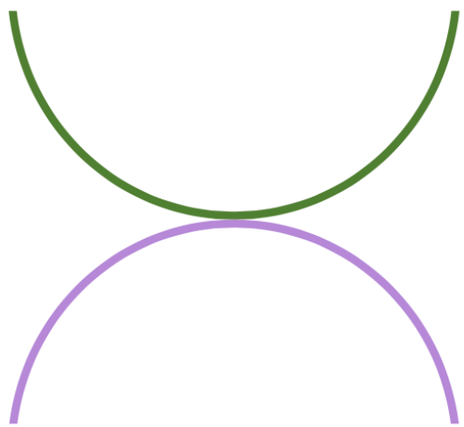
$$\min_{w,b} \frac{1}{2}\|w\|^2 \quad \text{s.t.} \quad t_i(w^T x_i + b) \geq 1 \quad \forall_{i=1\ldots N}$$

$x^T x'$

Primal problem vs Dual problem

Primal problem : original objective function

Minimize objective function



Dual problem : lower bound

maximize lower bound

Primal problem (hard margin)

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

Construct Lagrangian

To handle the inequality, introduce Lagrangian multipliers $\alpha_i \geq 0$ and form the Lagrangian:

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

$$\leq \frac{1}{2}\|w\|^2$$

$$\alpha := (\alpha_1, \dots \alpha_N)^T$$

Derive the dual

To obtain the dual, we minimize the Lagrangian w.r.t. $\mathbf{w}$ and b, and maximize w.r.t. α. I.e.

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

$$- \frac{\partial L}{\partial \mathbf{w}} = 0 \quad \Rightarrow \quad \mathbf{w} = \sum_{i=1}^{N} \alpha_i t_i \mathbf{x}_i$$

$$- \frac{\partial L}{\partial b} = 0 \quad \Rightarrow \quad \sum_{i=1}^{N} \alpha_i t_i = 0$$

Substitute these back into the Lagrangian

Eliminating w and b from $L(w, b, \alpha)$ using these conditions. Then the gives the dual representation

$$\tilde{L}(\alpha) := \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j t_i t_j x_i^T x_j$$

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

Dual optimization problem

$$\max_{\alpha} \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j t_i t_j x_i^T x_j$$

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

kernel trick (kernel SVM)

If the data is not linearly separable, we map the input data to a higher-dimensional space using a basis function $\Phi(\cdot)$ or we replace the dot product $x^T x'$ with a kernel $K(x, x')$. I.e.

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

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

Prediction function

Once the optimal α_i are found, the decision function for a new input $\mathbf{x}$ is:

$$f(\mathbf{x}) = \sum_{i=1}^{N} \alpha_i t_i K(\mathbf{x}_i, \mathbf{x}) + b$$

where b is a bias.

SVM: Overlapping classes (recall)

$$\min_{w, \xi} \ \frac{1}{2}\|w\|^2 + C \cdot \sum_{i=1}^{N} \xi_i$$ primal problem

$$\text{s.t.} \quad t_i (w^T x_i + b) \geq 1 - \xi_i \qquad \forall_{i=1 \ldots N}$$

$$\xi_i \geq 0$$

Here $C > 0$ controls the trade-off between the slack variable penalty and the margin

To obtain the dual, we minimize the Lagrangian w.r.t. $\mathbf{w}$ and b, and maximize w.r.t. $\boldsymbol{\alpha}$ and $\boldsymbol{\mu}$

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

where $\mathcal{L}(\mathbf{w}, b, \boldsymbol{\xi}, \boldsymbol{\alpha}, \boldsymbol{\mu})$

$$= \frac{1}{2} \|\mathbf{w}\|^2 + C \sum_{n=1}^{N} \xi_n - \sum_{n=1}^{N} \alpha_n (t_n y(\mathbf{x}_n) - 1 + \xi_n) - \sum_{n=1}^{N} \mu_n \xi_n$$

$\{\alpha_n \geq 0\}$ and $\{\mu_n \geq 0\}$ are Lagrange multipliers.

$$\alpha_n (t_n y(\mathbf{x}_n) - 1 + \xi_n)$$

We optimize out w, b and $\{\xi_i\}$

$$- \frac{\partial L}{\partial w} = 0 \quad \Rightarrow \quad w = \sum_{i=1}^{N} \alpha_i t_i x_i$$

$$- \frac{\partial L}{\partial b} = 0 \quad \Rightarrow \quad \sum_{i=1}^{N} \alpha_i t_i = 0$$

$$- \frac{\partial L}{\partial \xi_i} = 0 \quad \Rightarrow \quad \alpha_i = C - \mu_i$$

Using these results to eliminate w, b and $\{\xi_i\}$ from the Lagrangian, we obtain the dual Lagrangian

$$\tilde{L}(\alpha) = \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j t_i t_j x_i^T x_j$$

We note that $\alpha_i \geq 0$ and $\alpha_i = C - \mu_i$

Since $\mu_i \geq 0$, we have $\alpha_i \leq C$

Therefore we have to minimize $\tilde{L}(\alpha)$ w.r.t $\{\alpha_i\}$

subject to

$$0 \leq \alpha_i \leq C \quad \text{for} \quad i = 1, \ldots N$$

$$\sum_{i=1}^{N} \alpha_i t_i = 0$$

Kernel SVM: Overlapping classes

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

s.t. $0 \leq \alpha_i \leq C \quad \forall_{i=1,\ldots N}, \qquad \sum_{i=1}^{N} \alpha_i t_i = 0$ //