LECTURE 8 : SVM
SUPPORT VECTOR MACHINE

[1] Maximum Margin Classifiers

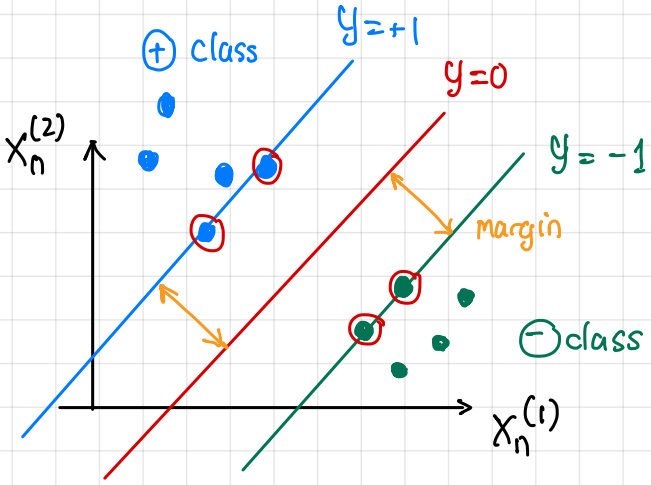
Considering two-class classification : $\{\underset{D\times 1}{\bar{x}_n}, \underset{1\times 1}{t_n}\}_{n=1}^{N}$ where $t_n \in \{+1, -1\}$

Classification model :

$$\underset{1\times 1}{y(\bar{x}_n)} = w_0 + \sum_{d=1}^{D} x_n^{(d)} = \sum_{d=0}^{D} w_0 x_n^{(d)}$$

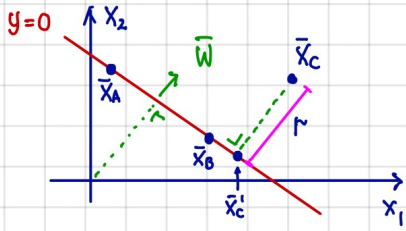
$$y(\bar{x}_n) = \bar{w}^T \bar{x}_n$$

; $\underset{(D+1)\times 1}{\bar{w}} = \begin{bmatrix} w_0 \\ w_1 \\ \vdots \\ w_D \end{bmatrix}$ $\underset{(D+1)\times 1}{\bar{x}_n} = \begin{bmatrix} 1 \\ x_n^{(1)} \\ x_n^{(2)} \\ \vdots \\ x_n^{(D)} \end{bmatrix}$



Green & blue lines are support vector lines

Basic idea : To maximize the margins using the support vectors (points circled)



$y(\bar{x}_A) = y(\bar{x}_B)$

$\bar{w}^T(\bar{x}_A - \bar{x}_B) = 0$ → Assuming polynomial basis

$\bar{w}$ is perpendicular to $(\bar{x}_A - \bar{x}_B)$

(Recall $\bar{a}\cdot\bar{b} = \|\bar{a}\|\|\bar{b}\|\cos\theta$)

From the above illustration : $\bar{x}_C = \bar{x}_C' + r\frac{\bar{w}}{\|\bar{w}\|}$

since $y(\bar{x}_C) = \bar{w}^T\bar{x}_C$ →

$$y(\bar{x}_C) = \bar{w}^T\left(\bar{x}_C' + r\frac{\bar{w}}{\|\bar{w}\|}\right) = \bar{w}^T\bar{x}_C' + r\,\bar{w}^T\frac{\bar{w}}{\|\bar{w}\|}$$

$$= 0 + r\|\bar{w}\|$$

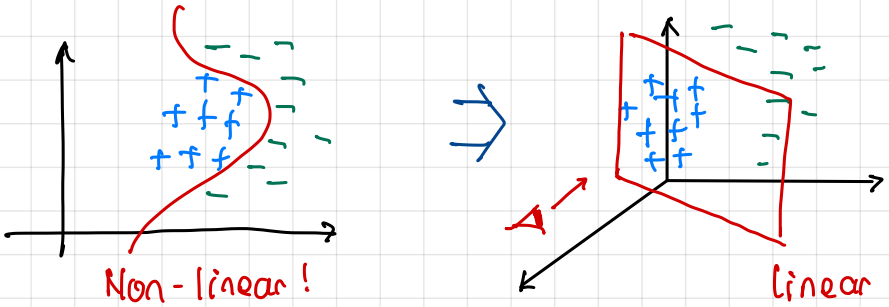
$$r = \frac{y(\bar{x}_C)}{\|\bar{w}\|}$$

Reading : Pattern Recognition & Machine Learning , Bishop
Chapter 7 : Sect. 7.1 (7.1.1 ~ 7.1.3)

[2] Basis Function: Space Transformation

We defined: $y(\bar{x}_n) = \bar{w}^T \bar{x}_n$ → Linear function / classifier

Problem:



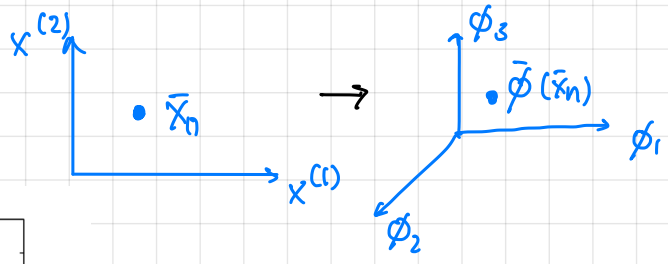
Higher dimensionality addresses the problem.

Q: How to increase the dimensionality of the data?

A: Basis functions: $y(\bar{x}_n) = \underset{1\times M}{\bar{w}^T} \underset{M\times 1}{\bar{\phi}}(\bar{x}_n) = \sum_{m=0}^{M-1} w_m \phi_m(\bar{x}_n)$

$$\underset{M\times 1}{\bar{w}} = \begin{bmatrix} w_0 \\ w_1 \\ \vdots \\ w_{M-1} \end{bmatrix} ; \quad \underset{(D+1)\times 1}{\bar{x}_n} = \begin{bmatrix} 1 \\ x_n^{(1)} \\ \vdots \\ x_n^{(D)} \end{bmatrix} ; \quad \underset{M\times 1}{\bar{\phi}(\bar{x}_n)} = \begin{bmatrix} 1 \\ \phi_1(\bar{x}_n) \\ \phi_2(\bar{x}_n) \\ \vdots \\ \phi_{M-1}(\bar{x}_n) \end{bmatrix}$$

e.g.: $\phi_m(\bar{x}_n) = \exp\left(-\frac{\|\bar{x}_n - \bar{\mu}_m\|_2^2}{2s^2}\right)$ → Gaussian Basis Function



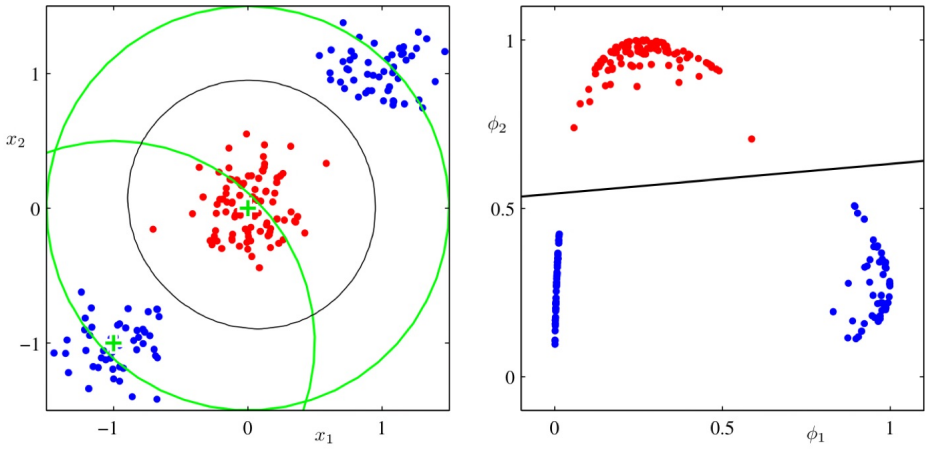


Figure 4.12 Illustration of the role of nonlinear basis functions in linear classification models. The left plot shows the original input space (x_1, x_2) together with data points from two classes labelled red and blue. Two 'Gaussian' basis functions $\phi_1(\mathbf{x})$ and $\phi_2(\mathbf{x})$ are defined in this space with centres shown by the green crosses and with contours shown by the green circles. The right-hand plot shows the corresponding feature space (ϕ_1, ϕ_2) together with the linear decision boundary obtained given by a logistic regression model of the form discussed in Section 4.3.2. This corresponds to a nonlinear decision boundary in the original input space, shown by the black curve in the left-hand plot.

Reading: Pattern Recognition and Machine Learning, Chapter 3, Sect. 3.1.

The distance of a point x_n to the decision surface:

$$r(\bar{x}_n) = \frac{t_n\, y(x_n)}{\|w\|} = \frac{t_n\,(\bar{w}^T \phi(x_n) + b)}{\|w\|} \quad ; t_n \in \{+1, -1\}$$

We want to maximize the margin between the decision line (red line) and the support vector lines. The problem is that we don't know both of them.

Mathematically, finding the support-vector points can be expressed:

$$r(\bar{x}_n) = \min_n \frac{t_n\, y(\bar{x}_n)}{\|\bar{w}\|} = \frac{1}{\|\bar{w}\|} \min_n t_n\, y(\bar{x}_n)$$

Find points that minimize the margin to the line ($\bar{w}$) → Support vectors
(red)

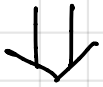
At the same time, we need to maximize the margins (the distance between the support vectors and the red line, $\bar{w}$): n & $\bar{w}$ are both unknown.

$$\{\bar{w}, b\}^* = \underset{\{\bar{w}, b\}}{\arg\max}\; r(\bar{x}_n) \quad ; b = w_0$$

$$= \underset{\{\bar{w}, b\}}{\arg\max}\; \frac{1}{\|\bar{w}\|} \min_n t_n\, y(\bar{x}_n)$$

$$\{\bar{w}, b\} = \underset{\{\bar{w}, b\}}{\arg\min}\; \frac{1}{2}\|\bar{w}\|^2 \min_n t_n\, y(\bar{x}_n)$$

$$\{\bar{w}, b\}^* = \underset{\{\bar{w}\}\{b\}}{\arg\min} \; \frac{1}{2}\|\bar{w}\|^2 \; \min_n \; t_n \, y(\bar{x}_n)$$



The equation can be expressed in a quadratic programming form:

$$\{\bar{w}, b\}^* = \underset{}{\arg\min} \; \frac{1}{2}\|w\|^2$$

$$\text{s.t.} \quad t_n(\bar{w}^T \phi(x_n) + b) \geq 1 \quad \text{for every } n$$

$t_n(\bar{w}^T \phi(x_n) + b) = 1$ for x_n that is closest to the decision surface.

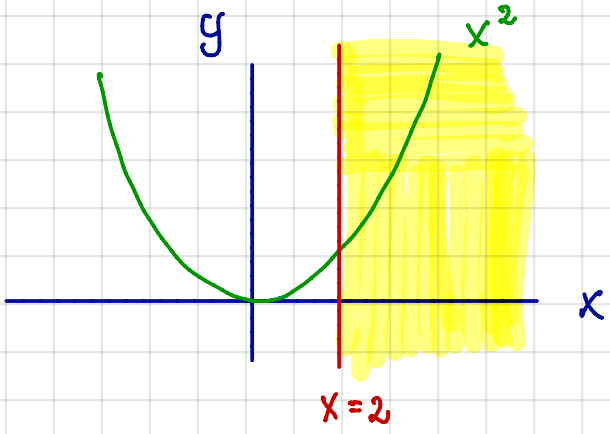
also recall $t_n \, y(\bar{x}_n) = 1$ as $t_n \in \{+1, -1\}$

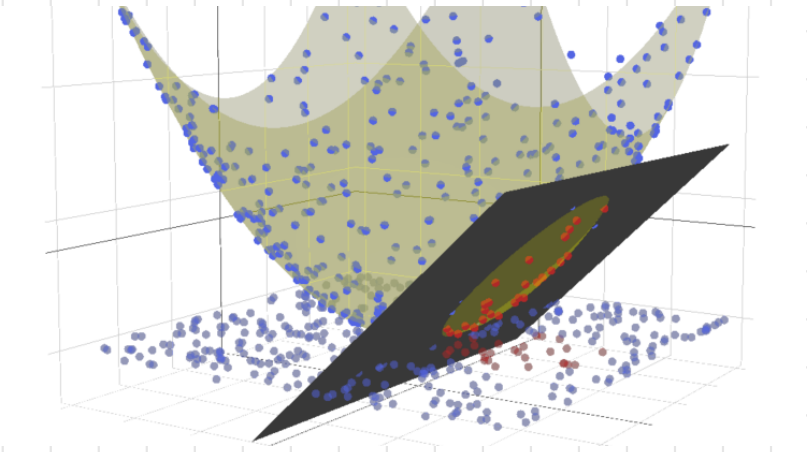
Note on quadratic programming:

(*) Simple example:

$$\min \quad x^2$$

$$\text{s.t.} \quad x \geq 2$$





$$\{\bar{w}, b\}^* = \arg\min \frac{1}{2}\|w\|^2$$

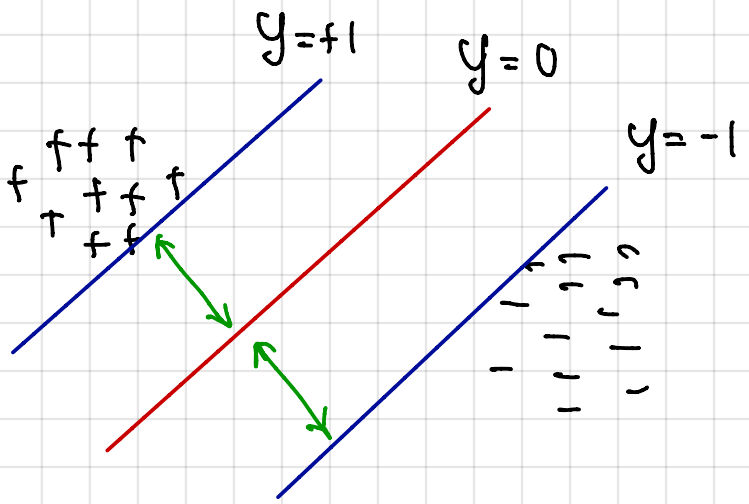
$$\text{s.t.} \quad t_n(\bar{w}^T \phi(x_n) + b) \geq 1 \quad \text{for every } n$$

$$\Downarrow$$

Q: What is the meaning of the constraint?

A: $t_n y_n \geq 1$ means: The distance between the separation line and the sample points must be at least 1.

(1 indicate the closest distance)



$t_n y_n = 1 \rightarrow$ Implying $\bar{x}_n$ is a support-vector point

We can solve the above equation using quadratic programming. but $\phi(\bar{x}_n)$ needs M to be set. Moreover, there are many hyperparameters in ϕ.

Note: Lagrangian Duality:

Primal equation (Quadratic programming):

$$\min. \quad f_0(x) \rightarrow \text{quadratic equation}$$
$$\text{s.t.} \quad g_i(x) \leq 0 \quad ; \quad i = 1, \ldots, m$$

Dual equation (Lagrangian equation):

$$L(x, \bar{\lambda}, \bar{v}) = f_0(x) + \sum_{i=1}^{m} \lambda_i g_i(x)$$

Using the Lagrangian equation:

$$\{\bar{w}, b\}^* = \underset{\frac{1}{2}}{\operatorname{argmin}} \|w\|^2$$
$$\text{s.t.} \quad t_n(\bar{w}^T \phi(x_n) + b) \geq 1 \qquad \forall n$$

$\Downarrow$ Lagrange Duality

$$L(\bar{w}, b, \bar{\lambda}) = \frac{1}{2}\|\bar{w}\|^2 - \sum_{n=1} \lambda_n \left(t_n(\bar{w}^T \bar{\phi}(x_n) + b) - 1 \right)$$

where $\bar{\lambda} = (\lambda_1, \ldots, \lambda_N)^T$ the Lagrangian multipliers ; $\lambda_n \geq 0$

$$\frac{\partial L(\bar{w}, b, \bar{\lambda})}{\partial \bar{w}} = 0 \quad \rightarrow \quad \bar{w} = \sum_{n=1}^{N} \lambda_n t_n \phi(x_n)$$

$$\frac{\partial L(\bar{w}, b, \bar{\lambda})}{\partial b} = 0 \quad \rightarrow \quad 0 = \sum_{n=1}^{N} \lambda_n t_n$$

Note: $\|\bar{w}\| = \sqrt{\sum_m w_m^2} \quad \rightarrow \|\bar{w}\|^2 = \bar{w}^T \bar{w} = \sum_m w_m^2$

$$\underset{M \times 1}{\frac{\partial \|\bar{w}\|^2}{\partial \bar{w}}} = \begin{bmatrix} \frac{\partial}{\partial w_1} \|\bar{w}\|^2 \\ \vdots \\ \frac{\partial}{\partial w_M} \|\bar{w}\|^2 \end{bmatrix} = \begin{bmatrix} \frac{\partial}{\partial w_1} \sum_m w_m^2 \\ \vdots \\ \frac{\partial}{\partial w_M} \sum_m w_m^2 \end{bmatrix} = 2 \begin{bmatrix} w_1 \\ \vdots \\ w_M \end{bmatrix} = 2\bar{w}$$

Substituting $\bar{w} = \sum_{n=1}^{N} \lambda_n t_n \bar{\phi}(\bar{x}_n)$ to $L(\bar{w}, b, \bar{a})$:

(1) $\|\bar{w}\|^2 = \bar{w}^T \bar{w}$

$$= \left[\sum_{n=1}^{N} \lambda_n t_n \bar{\phi}(\bar{x}_n)\right]^T \left[\sum_{m=1}^{N} \lambda_m t_m \bar{\phi}(\bar{x}_m)\right]$$

$$= \sum_{n=1}^{N} \lambda_n t_n \bar{\phi}^T(\bar{x}_n) \sum_{m=1}^{N} \lambda_m t_m \bar{\phi}(\bar{x}_m)$$

$$= \sum_{n}^{N} \sum_{m}^{N} \left[\lambda_n t_n \bar{\phi}^T(\bar{x}_n)\right] \lambda_m t_m \bar{\phi}(\bar{x}_m)$$

$$= \sum_{n} \sum_{m} \lambda_n \lambda_m t_n t_m \bar{\phi}^T(\bar{x}_n) \bar{\phi}(\bar{x}_m)$$

We free ourselves from setting M! ↑ $k(\bar{x}_n, \bar{x}_m)$

(2) $\sum_{n} \lambda_n \left(t_n(\bar{w}^T \bar{\phi}(x_n) + b) - 1\right) = \sum_{n} \lambda_n t_n \bar{w}^T \bar{\phi}_n + \sum_{n} \lambda_n t_n b - \sum_{n} \lambda_n$

* $\sum_{n} \lambda_n t_n \bar{w}^T \bar{\phi}_n = \sum_{n} \lambda_n t_n \left(\sum_{m} \lambda_m t_m \bar{\phi}_m^T\right) \bar{\phi}_n$

$$= \sum_{n} \sum_{m} \lambda_n \lambda_m t_n t_m \bar{\phi}_m^T \bar{\phi}_n$$

* $\sum_{n} \lambda_n t_n b = b \sum_{n} \lambda_n t_n = 0$

$$y_n = \bar{w}^T \bar{\phi}(\bar{x}_n) + b = \sum_{m} \lambda_m t_m \bar{\phi}_m^T \bar{\phi}_n + b$$

$$= \sum_{m} \lambda_m t_m k(\bar{x}_m, \bar{x}_n) + b$$

Hence:

$$\tilde{L}(\lambda) = \sum_{n=1}^{N} \lambda_n - \frac{1}{2} \sum_{n=1}^{N} \sum_{m=1}^{N} \lambda_n \lambda_m t_n t_m k(x_n, x_m)$$

Q: What is $k(x_i, x_j)$?

Previously: $k(x_i, x_j) = \bar{\phi}^T(x_i)\,\bar{\phi}(x_j)$

($k(x_i, x_j)$: 1×1; $\bar{\phi}^T(x_i)$: 1×M; $\bar{\phi}(x_j)$: M×1)

where: $\bar{\phi}(x_i)$ is the basis function → e.g. Gaussian or Sigmoid basis.

Assuming $M=1$ and Gaussian basis functions:

$$\phi(x_i)\phi(x_j) = \exp\left(-\frac{(x_i-\mu_1)^2}{2s^2}\right)\exp\left(-\frac{(x_j-\mu_1)^2}{2s^2}\right)$$

$$= \exp\left[-\frac{(x_i-\mu_1)^2 + (x_j-\mu_1)^2}{2s^2}\right]$$

$$= \exp\left[-\frac{x_i^2 + x_j^2 - 2x_i\mu_1 - 2x_j\mu_1 + \mu_1^2}{2s^2}\right]$$

Thus, $\phi(x_i)\phi(x_j)$ depends on x_i & x_j as well as the parameters of the basis functions, which the number can be large.

Reading: Pattern Recognition and Machine Learning, Bishop

- Chapter 3, Sect. 3.1. (for Basis Functions)
- Chapter 6, Sect. 6.2 (for Kernels)

The illustration of $k(x_i, x_j)$ based on polynomials, Gaussians and sigmoids:

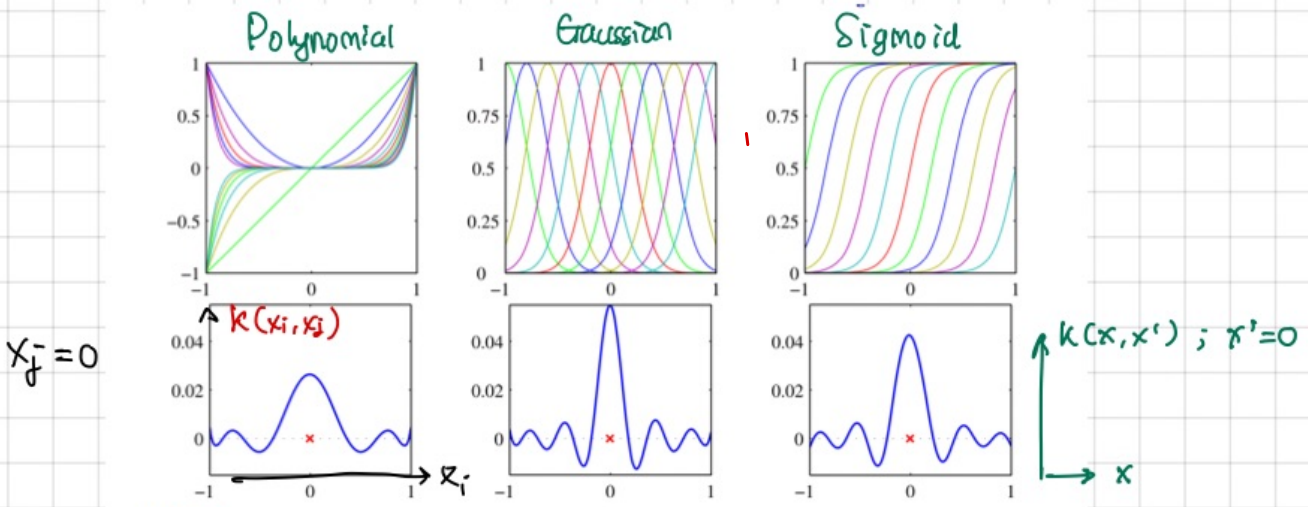


Figure 6.1 Illustration of the construction of kernel functions starting from a corresponding set of basis functions. In each column the lower plot shows the kernel function $k(x, x')$ defined by (6.10) plotted as a function of x for $x' = 0$, while the upper plot shows the corresponding basis functions given by polynomials (left column), 'Gaussians' (centre column), and logistic sigmoids (right column).

Instead of using basis functions, we can use a kernel function:

see textbook fig. 6.5 (page #308)

$$k(\bar{x}_i, \bar{x}_j) = \theta_0 \exp\left\{-\frac{\theta_1}{2}\|\bar{x}_i - \bar{x}_j\|^2\right\} + \theta_2 + \theta_3 \bar{x}_i^T \bar{x}_j$$

$\|\cdot\|$ → means: magnitude

Using this kernel, the number of hyperparameters is reduced to only 4.

Kernel visualization with different values of θ's:

There is no ϕ anymore

4 different values of x_j

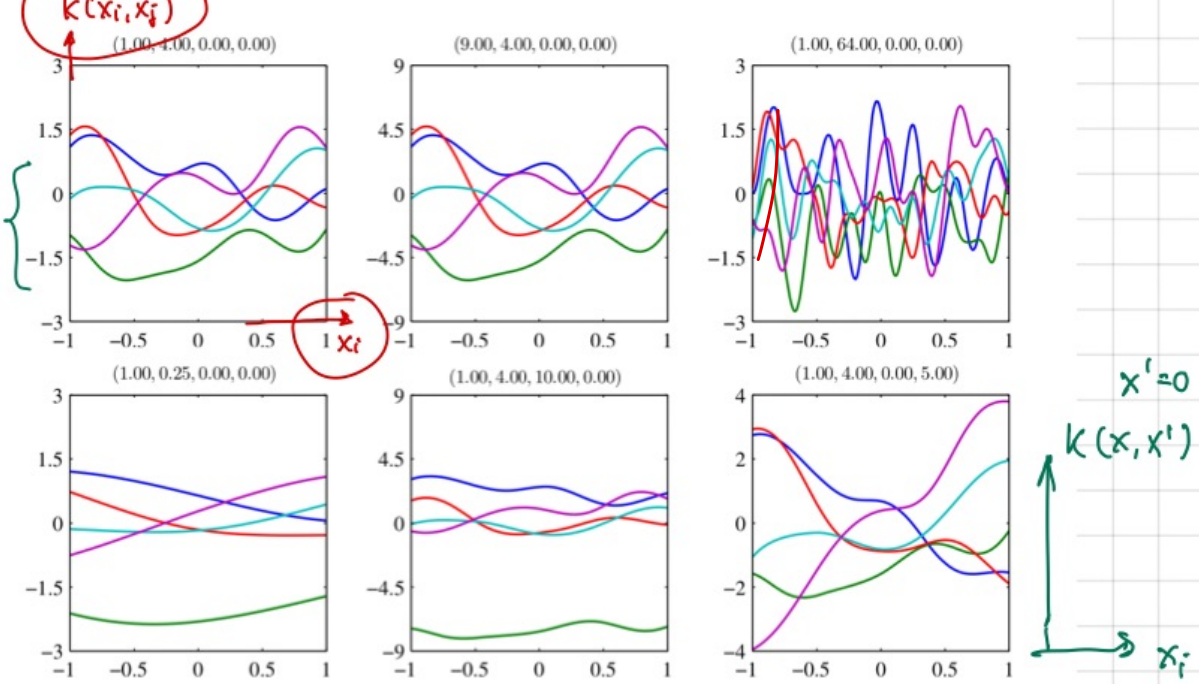


Figure 6.5 Samples from a Gaussian process prior defined by the covariance function (6.63). The title above each plot denotes $(\theta_0, \theta_1, \theta_2, \theta_3)$.

Overall:

$$\tilde{L}(\bar{\lambda}) = \sum_{n=1}^{N} \lambda_n - \frac{1}{2} \sum_{n=1}^{N} \sum_{m=1}^{N} \lambda_n \lambda_m t_n t_m k(x_n, x_m)$$

$$\text{s.t.}: \quad \lambda_n \geq 0$$

$$\sum_{n=1} \lambda_n t_n = 0$$

} Quadratic programming w.r.t. $\bar{\lambda}$

where: $k(x_n, x_m) = \phi^T(x_n)\, \phi(x_m)$

Note: we want to estimate $\bar{\lambda}$ from $\tilde{L}(\bar{\lambda})$. Having the values of $\bar{\lambda}$, we can obtain $\bar{w}$ & b.

The constrained equation satisfies the KKT (Karush-Kuhn-Tucker) conditions:

$$\lambda_n \geq 0$$
$$t_n y(\bar{x}_n) - 1 \geq 0$$
$$\lambda_n (t_n y(\bar{x}_n) - 1) = 0$$

} see notes on KKT (next page)

Thus: for every data point, $\bar{x}_n$, either $\lambda_n = 0$ or $t_n y(\bar{x}_n) = 1$.

For $\bar{x}_n$ which $\lambda_n = 0$ → this $\bar{x}_n$ doesn't affect the decision $\tilde{L}(\lambda)$

For $\bar{x}_n$ which $\lambda_n > 0$ → This $\bar{x}_n$ is called support vector, and because $t_n y(\bar{x}_n) = 1$, it corresponds to a point lying on the max. margin hyperplanes in feature space.

After solving the quadratic optimization equation, $\tilde{L}(\bar{\lambda})$: we can obtain $\bar{\lambda}$, and select λ_n that is not zero (= support vector points). From these non-zero λ's we can compute:

$$t_n\, y(\bar{x}_n) = 1 \quad ; \text{ for any point } n \text{ in the support set.}$$

$$t_n \left(\sum_{m \in S} \lambda_m t_m k(\bar{x}_m, \bar{x}_n) + b \right) = 1$$

see page #6

S → set of indices of the support vectors.

$$t_n \left[t_n \left(\sum_{m \in S} \lambda_m t_m k(\bar{x}_m, \bar{x}_n) + b \right) \right] = t_n \quad ; \quad t_n^2 = 1$$

$$b_n = t_n - \sum_{m \in S} \lambda_m t_m k(\bar{x}_m, \bar{x}_n)$$

more stable

$$b = \frac{1}{N_s} \sum_{n \in S} b_n$$

$$\downarrow\downarrow$$

$$y_n = \sum_{m \in S} \lambda_m t_m k(\bar{x}_m, \bar{x}_n) + b$$

$$b = \frac{1}{N_s} \sum_{n \in S} \left[t_n - \sum_{m \in S} \lambda_m t_m k(\bar{x}_m - \bar{x}_n) \right]$$

The decision boundary in the feature space.

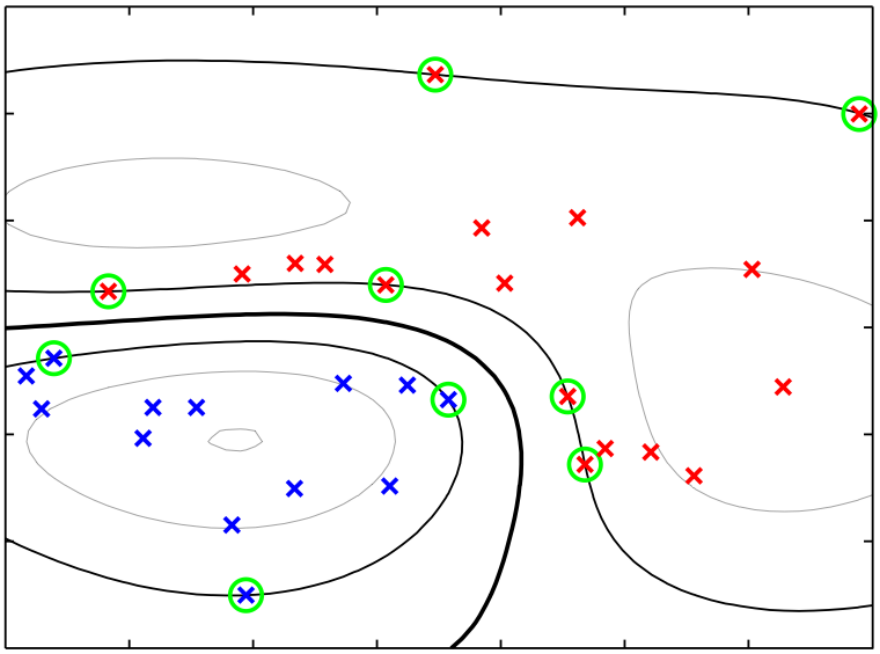
- Testing Stage: Given a new input $\bar{x}_*$, we can classify this using:

$$y(\bar{x}_*) = \sum_{m \in S} \lambda_m t_m k(\bar{x}_m, \bar{x}_*) + b$$

Where:

$$\text{sign}(y(x_*)) = \begin{cases} + & : \text{class 1} \\ - & : \text{class 2} \end{cases}$$

Example of synthetic data from two classes in two dimensions showing contours of constant $y(\mathbf{x})$ obtained from a support vector machine having a Gaussian kernel function. Also shown are the decision boundary, the margin boundaries, and the support vectors.



In the hyper-space the decision boundary/plane is linear.

Summary

$$\{\bar{w}, b\}^* = \underset{\{\bar{w}\}\{b\}}{\arg\max}\ \gamma(\bar{x}_n)$$

$$= \underset{\{\bar{w}\}\{b\}}{\arg\max}\ \frac{1}{\|\bar{w}\|} \min_n t_n y(\bar{x}_n)$$

$$\{\bar{w}, b\}^* = \arg\min \frac{1}{2}\|\bar{w}\|^2$$

$$\text{s.t. } t_n y_n \geq 1 \quad \forall n$$

} Quadratic programming

⇓

Lagrangian duality:

$$L(\bar{w}, b, \bar{\lambda}) = \frac{1}{2}\|w\|^2 - \sum_n \lambda_n (t_n y_n - 1)$$

⇓

$$\tilde{L}(\bar{\lambda}) = \sum_{n=1} \lambda_n - \frac{1}{2}\sum_n \sum_m \lambda_n \lambda_m t_n t_m k(\bar{x}_n, \bar{x}_m)$$

$$\text{s.t. } \lambda_n \geq 0$$

$$\sum_n \lambda_n t_n = 0$$

KKT conditions: $\lambda_n (t_n y_n - 1) = 0$

⇓

$$y_n = \sum_{m \in S} \lambda_m t_m k(\bar{x}_m, \bar{x}_n) + b$$

$$b = \frac{1}{N_s} \sum_{n \in S} \left[t_n - \sum_{m \in S} \lambda_m t_m k(\bar{x}_m - \bar{x}_n) \right]$$

} The decision boundary in the feature space.

[2] Overlapping Class Distributions

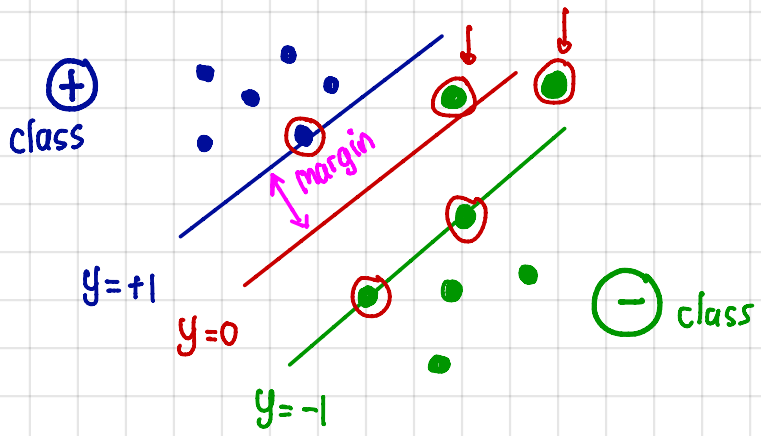
Problem: We assume the class distributions are linearly separable in feature space $\bar{\phi}(\bar{x})$. As a result, SVM will give exact separation of the training data in the original input space, $\bar{x}$.

⇓

This can lead to an overfitting problem (poor generalization for the testing data). Since, the training data can be affected by noise.

⇓

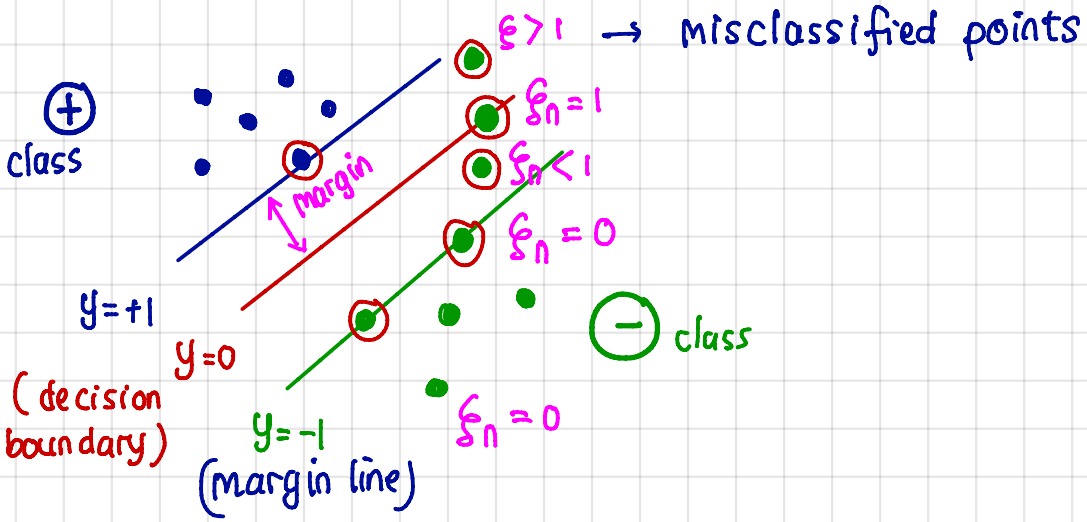
Idea: to allow some training data to be misclassified, but with a penalty that increases with the distance from the boundary.



⇓

To introduce: slack variables, $\xi_n \geq 0$.

$$\xi_n = \begin{cases} 0 & : \bar{x}_n \text{ is on or inside the correct margin line} \\ 1 & : \bar{x}_n \text{ is on the decision boundary} \\ 0 < \xi_n < 1 & : \bar{x}_n \text{ is inside the margin} \\ > 1 & : \bar{x}_n \text{ is on a wrong side} \end{cases}$$



Previous classification constraint: $t_n\, y(\bar{x}_n) \geq 1$,

is replaced by:

$$t_n\, y(\bar{x}_n) \geq 1 - \xi_n$$

(ξ_n: known)

where $\xi_n \geq 0$.

Goal: to maximize the margin while softly penalizing points lying on the wrong side:

$$\min_{\{\bar{w}\}} \; C \sum_{n=1}^{N} \xi_n + \frac{1}{2} \|\bar{w}\|^2$$

$$\text{s.t.} \quad t_n\, y(\bar{x}_n) \geq 1 - \xi_n \qquad \forall n$$

where: C is the weighting factor between the slack variable and the margin.

The corresponding Lagrangian:

$$L(\bar{w}, b, \xi, \bar{\lambda}, \bar{\mu}) = \frac{1}{2}\|\bar{w}\|^2 + C\sum_{n=1}^{N}\xi_n - \sum_{n=1}^{N}\lambda_n\{t_n y(\bar{x}_n) - 1 + \xi_n\} - \sum_{n=1}^{N}\mu_n\xi_n$$

$\bar{\lambda}, \bar{\mu}$: Lagrangian Multipliers

where : $\lambda_n \geq 0$ and $\mu_n \geq 0$

The corresponding set of KKT conditions:

$$\lambda_n \geq 0 \quad ; \quad \mu_n \geq 0$$
$$t_n y(\bar{x}_n) - 1 + \xi_n \geq 0 \quad ; \quad \xi_n \geq 0$$
$$\lambda_n(t_n y(\bar{x}_n) - 1 + \xi_n) = 0 \quad ; \quad \mu_n\xi_n = 0$$

Optimizing out $\bar{w}$, b, and $\{\xi_n\}$:

$$\frac{\partial L}{\partial \bar{w}} = 0 \rightarrow \bar{w} = \sum_{n=1}^{N}\lambda_n t_n \bar{\phi}(\bar{x}_n)$$

$$\frac{\partial L}{\partial b} = 0 \rightarrow \sum_{n=1}^{N}\lambda_n t_n = 0$$

$$\frac{\partial L}{\partial \xi_n} = 0 \rightarrow \lambda_n = C - \mu_n$$

Putting these $\bar{w}$, b and $\{\xi_n\}$ back to the equation:

$$\tilde{L}(\bar{\lambda}) = \sum_{n=1}^{N}\lambda_n - \frac{1}{2}\sum_{n=1}^{N}\sum_{m=1}^{N}\lambda_n\lambda_m t_n t_m k(x_n, x_m)$$

s.t. :

$$0 \leq \lambda_n \leq C$$
$$\sum_{n=1}^{N}\lambda_n t_n = 0$$

As before, points that have $\lambda_n > 0$ are the support vectors, which satisfies: $t_n y(\bar{x}_n) = 1 - \xi_n$.

If $\lambda_n < C \rightarrow \mu_n > 0$ which requires $\xi_n = 0$ and thus such points lie on the margin.

If $\lambda_n = C \rightarrow \xi_n \leq 1 \rightarrow$ lie on the margin & correctly classied

$\xi_n > 1 \rightarrow$ lie on the margin & incorrectly classified.

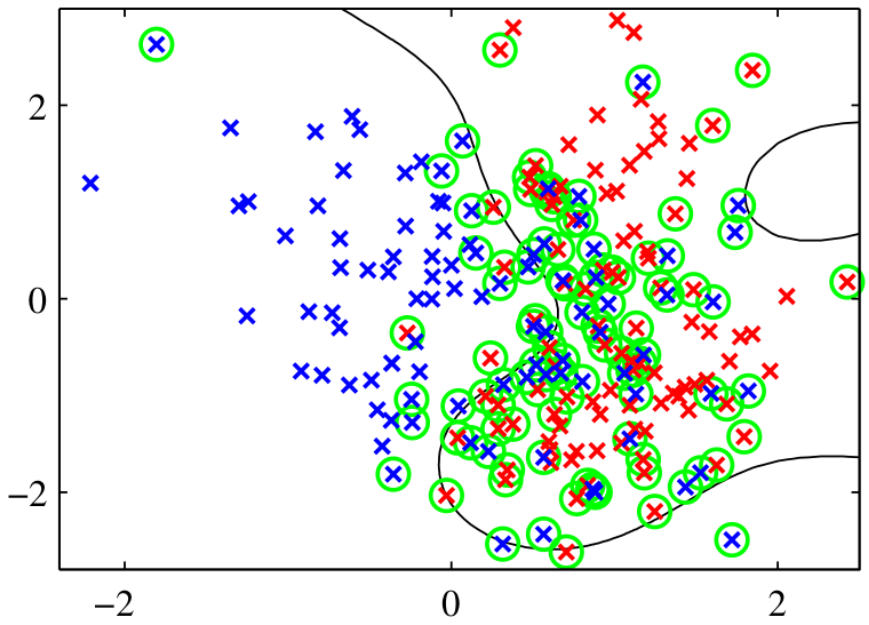
To determine b from the support vectors:

$$t_n \left(\sum_{m \in S} \lambda_m t_m k(\bar{x}_m, \bar{x}_n) + b \right) = 1$$

$$b = \frac{1}{N_M} \sum_{n \in M} \left(t_n - \sum_{m \in S} a_m t_m k(\bar{x}_n, \bar{x}_m) \right)$$

↳ set of indices of point having $0 < \lambda_n < C$.

Illustration of the ν-SVM applied to a nonseparable data set in two dimensions. The support vectors are indicated by circles.



MULTI-CLASSES SVM

Basic concept: One against all (thus basically only 2 classes)

- Assuming there are C classes, we run SVM (C-1) times, and then find the one that is classified uniquely.

  e.g. K=3 : 1. SVM ① against ②,③ → classified as ②,③
  2. SVM ② against ①,③ → classified as ①,③

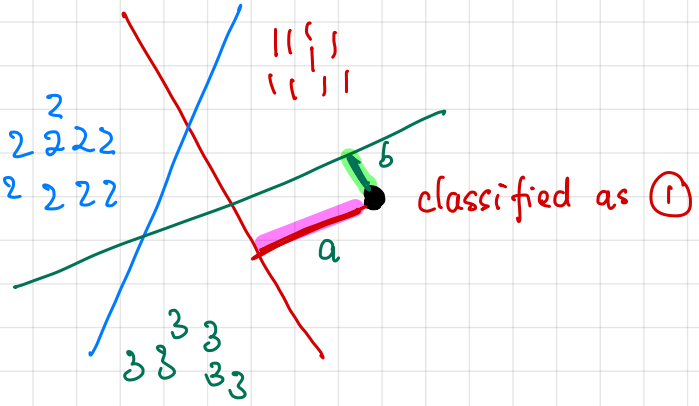
  Then the label should be ③

- Drawback: Multiple answers.

  e.g: 1. SVM ① against ②,③ → classified ①
  2. SVM ② against ①,③ → classied ②

- Heuristic solution: $y(\bar{x}_*) = \max_k y_k(\bar{x}_*)$

  » Choose the class that gives the highest margin to the boundary.



Reading: Pattern Recognition & Machine Learning, Bishop
Chapter 7: Sect. 7.1 (7.1.1 ~ 7.1.3)

Notes on Lagrange Multipliers & KKT conditions:

1. Lagrange multipliers is used to find stationary points of a multivariate function s.t. one or more constraints.

e.g.:

$$x^* = \underset{\{x\}}{\arg\min}\ f(x)$$
$$\text{s.t. } g(x) = 0 \quad \rightarrow \text{Equality constraint (boundary constraint)}$$

$$\Downarrow$$

$$L(x, \lambda) = f(x) + \lambda g(x)$$

↳ Lagrangian function

Optimizing $L(x, \lambda)$ w.r.t. x:

$$\frac{\partial L}{\partial x} = 0$$

w.r.t. λ: $\frac{\partial L}{\partial \lambda} = 0 \rightarrow \frac{\partial L}{\partial \lambda} = g(x) = 0$

2. Non-equality constraint: $g(x) \leq 0$ (area constraint)

see next page for illustration.

Conditions:

① $g(x) < 0 \rightarrow$ inactive constraint $\rightarrow$ $g(x)$ has no role $\lambda = 0$

② $g(x) = 0 \rightarrow$ active constraint $\rightarrow \lambda \neq 0$

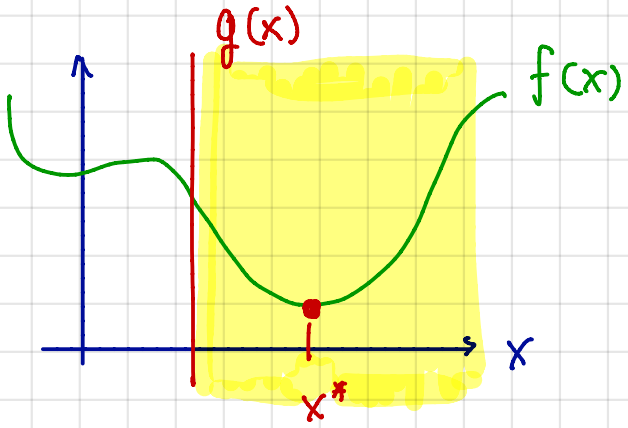
$$\Downarrow$$

$$\lambda g(x) = 0$$

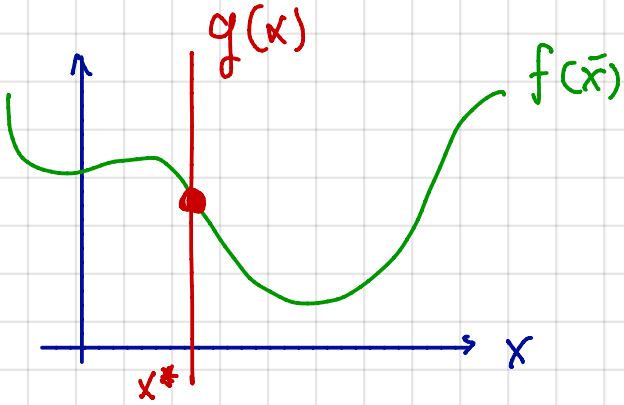
In SVM, $g(x) = (t_n y(\vec{x}_n) - 1)$

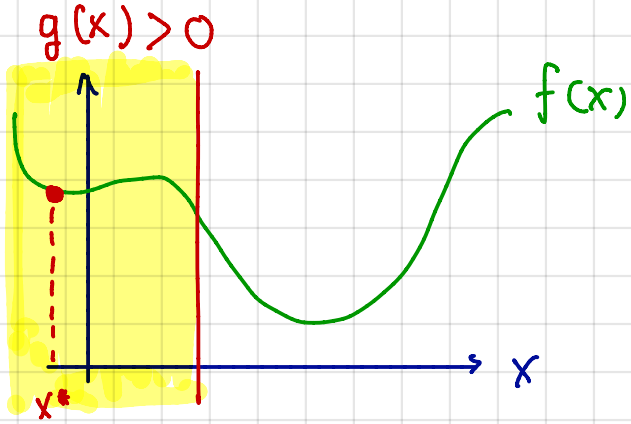
KKT conditions :

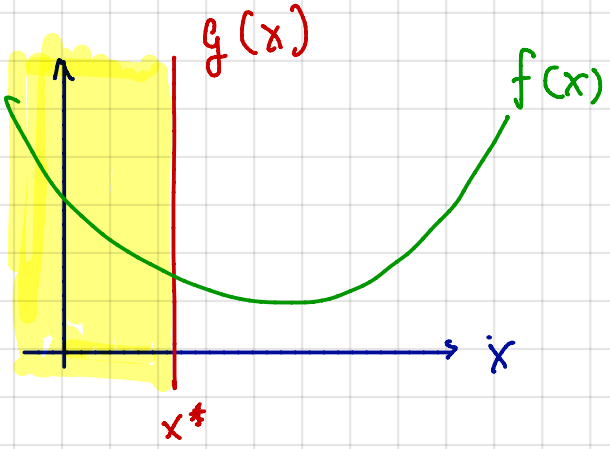
① $g(x) > 0$



② $g(x) = 0$







In above two cases, $g(x)$ doesn't have any role in deciding the stationary point. The stationary is decided by $f(\bar{x})$.

$\Downarrow$

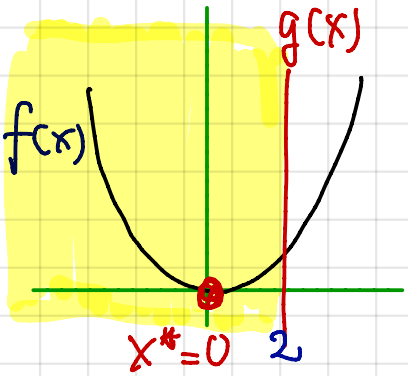
$\alpha = 0$

$L(x, \lambda) = f(x) + \lambda g(x)$

For these two cases, the stationary point intersects with $g(x)$ $\rightarrow$ $\alpha > 0$ because the min of $L(x, \lambda)$ is decided by $g(x)$.

KKT condition examples :

1. $\min \; x^2$
   $s.t. \; x \leq 2$



Dual: $L(x, \lambda) = x^2 + \lambda(x-2)$

$f(x) = x^2 \; ; \; g(x) = (x-2)$

Minimizing $L(x, \lambda)$: correct one

① $\frac{\partial L}{\partial x} = 2x + \lambda = 0 \Rightarrow 2x = -\lambda \rightarrow \lambda = 0 \rightarrow g(x^*) \neq 0$

$x^* = 0 \rightarrow g(x^*=0) = -2$

② $\frac{\partial L}{\partial \lambda} = x - 2 = 0 \Rightarrow x = 2 \rightarrow \lambda > 0 \rightarrow g(x^*) = 0$

$g(x^*=2) = 0$

---

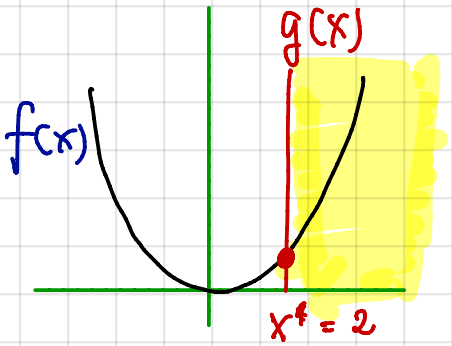
2. $\min \; x^2$
   $s.t. \; x \geq 2$

$\Rightarrow \min \; x^2$
$s.t \; -x \leq -2$

$\Rightarrow \min \; x^2$
$s.t \; -x + 2 \leq 0$

Dual: $L(x, \lambda) = x^2 + \lambda(-x+2)$

$f(x) = x^2 \; ; \; g(x) = 2 - x$



① $\frac{\partial L}{\partial x} = 2x - \lambda = 0 \Rightarrow x = \frac{\lambda}{2} \rightarrow \lambda = 0 \rightarrow g(x^*) \neq 0$

$g(x^*=0) = 2$

② $\frac{\partial L}{\partial \lambda} = 2 - x = 0 \Rightarrow x = 2 \rightarrow \lambda > 0 \rightarrow g(x^*) = 0$

$g(x^*=2) = 0$