InfoLab

IC612: Data Warehousing and Data Mining

Lecture 9:
Data Classification: Advanced Concepts (2)

Min-Soo Kim

DGIST 대구경북과학기술원
Daegu Gyeongbuk Institute of Science & Technology

InfoLab

Linear Decision Surfaces and Functions

DGIST 대구경북과학기술원
Daegu Gyeongbuk Institute of Science & Technology

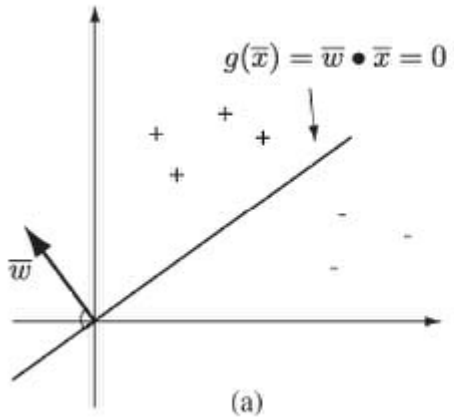
Binary classification problem

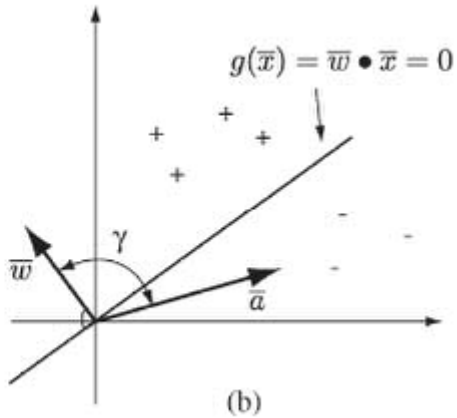
- **Linear decision surface : a line, plane, or hyperplane that separates the two groups of objects**
  - we use decision functions rather than decision surfaces to make all this mathematically more convenient
- **From data sets to decision functions**
  - an object x is a position vector in the n-dimensional dot product space $\mathbf{R^n}$, i.e. $\overline{x} \in \mathbf{R^n}$ (n : the number of attributes)
  - sample set $S \subset \mathbf{R^n}$
  - target function $f : \mathbf{R^n} \rightarrow \{+1, -1\}$
  - training set $D = \{(\overline{x}, y) \mid \overline{x} \in S \text{ and } y = f(\overline{x})\}$
- **Problem**
  - compute a function $\hat{f}: R^n \rightarrow \{+1, -1\}$ using D such that $\hat{f}(\overline{x}) \cong f(\overline{x})$ for all $\overline{x} \in \mathbf{R^n}$
  - use linear decision surface to construct the model $\hat{f}$

Linear decision surfaces through the origin

- **Assumption**
  - two-dimensional dot product space
  - linearly separable training data set
  - guarantee to find a line *g* that separates the two classes perfectly
  - guarantee that the line goes through the origin of dot product space
- **Decision surface**

$$\hat{f}(\overline{x}) = \begin{cases} +1 & \text{if } g(\overline{x}) \geq 0 \\ -1 & \text{if } g(\overline{x}) < 0 \end{cases}$$





$$g(\overline{a}) = \overline{w} \bullet \overline{a} = |\overline{w}||\overline{a}|\cos(\gamma) = k$$

Linear decision surfaces with an offset

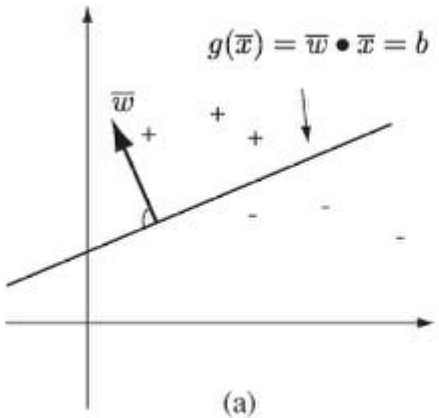
- **Relax assumption**
  - decision surface has not to go through the origin of dot product space
  - form of decision surface is $g(\overline{x}) = \overline{w} \bullet \overline{x} = b$

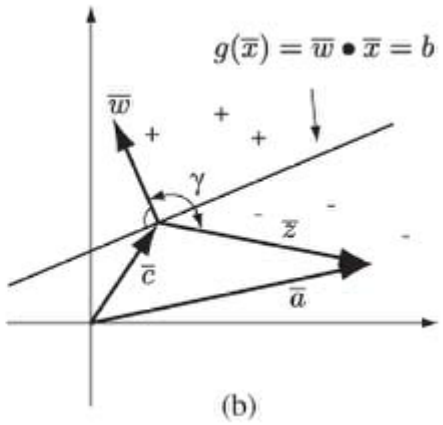
$$g(\overline{c}) = \overline{w} \bullet \overline{c} = b$$

$$\overline{a} = \overline{c} + \overline{z}$$

$$\overline{z} = \overline{a} - \overline{c}$$

- **Decision surface**
  - $\overline{c}$: some arbitrary point, say , on the decision surface itself
  - if we consider $\overline{z}$ to be the "position vector" of point $\overline{a}$ with respect to point $\overline{c}$, the point $\overline{c}$ offers almost the same perspective in p.4





$$\overline{w} \bullet \overline{z} = |\overline{w}||\overline{z}| \cos(\gamma) = k$$

$$\begin{aligned} \overline{w} \bullet \overline{z} &= \overline{w} \bullet (\overline{a} - \overline{c}) \\ &= \overline{w} \bullet \overline{a} - \overline{w} \bullet \overline{c} \\ &= \overline{w} \bullet \overline{a} - b \\ &= g(\overline{a}) - b \end{aligned}$$

$$\hat{f}(\overline{x}) = \begin{cases} +1 & \text{if } g(\overline{x}) - b \geq 0 \\ -1 & \text{if } g(\overline{x}) - b < 0 \end{cases}$$

$$\hat{f}(\overline{x}) = \text{sgn}(\overline{w} \bullet \overline{x} - b)$$

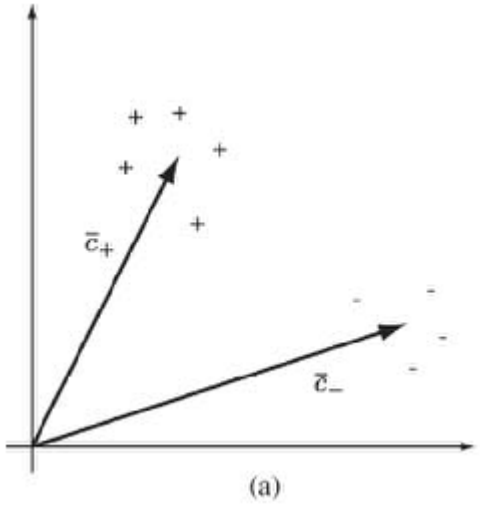
Simple learning algorithm

- **Linearly separable training set**
  - D = $\{(\bar{x}_1, y_1), (\bar{x}_2, y_2), \ldots, (\bar{x}_l, y_l)\}$
  - $y_i \in \{+1, -1\}$

- **Step1**
  - computing the average objects or means of the two respective classes in D
  - mean for the class labeled +1 : $\bar{c}_+$
  - mean for the class labeled −1 : $\bar{c}_-$

$$\bar{c}_+ = \frac{1}{l_+} \sum_{(\bar{x}_i, +1) \in D} \bar{x}_i \qquad l_+ = |\{(\bar{x}, y) \mid (\bar{x}, y) \in D \text{ and } y = +1\}|$$

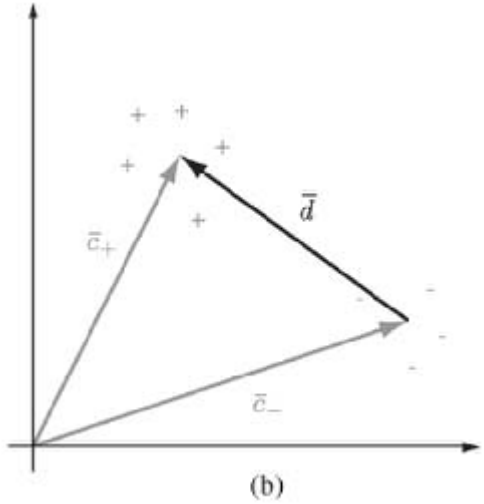
$$\bar{c}_- = \frac{1}{l_-} \sum_{(\bar{x}_i, -1) \in D} \bar{x}_i \qquad l_- = |\{(\bar{x}, y) \mid (\bar{x}, y) \in D \text{ and } y = -1\}|$$



(a)

Step2

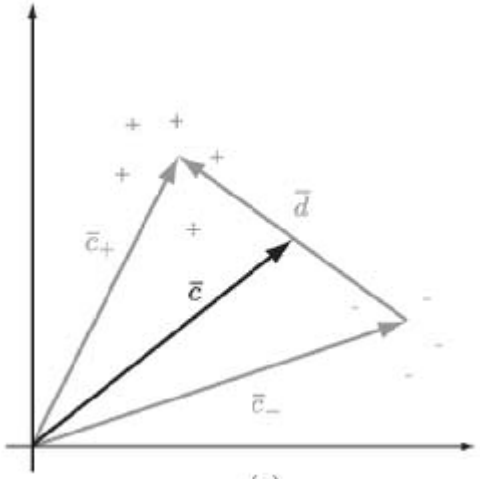
- construct the vector $\bar{d}$ such that $\bar{d} = \bar{c}_+ + \bar{c}_-$



(b)

Step3

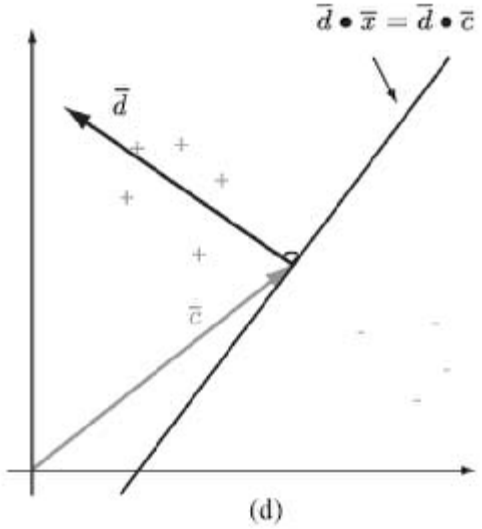
- compute the mean, call it $\bar{c}$, between the two class means $\bar{c}_+$ and $\bar{c}_-$



(c)

Step4

- translate the vector $\overline{d}$ so that it is rooted in the average object $\overline{c}$
- construct a line perpendicular to $\overline{d}$ through $\overline{c}$
- interpret this line as a decision surface
  - $\overline{w} = \overline{d}$
  - $b = \overline{d} \cdot \overline{c}$



Step5

- plugging equations $\overline{w} = \overline{d}$ and $b = \overline{d} \cdot \overline{c}$ into the equation for a linear decision function $\hat{f}(\overline{x}) = \text{sgn}(\overline{w} \bullet \overline{x} - b)$ gives us the decision function

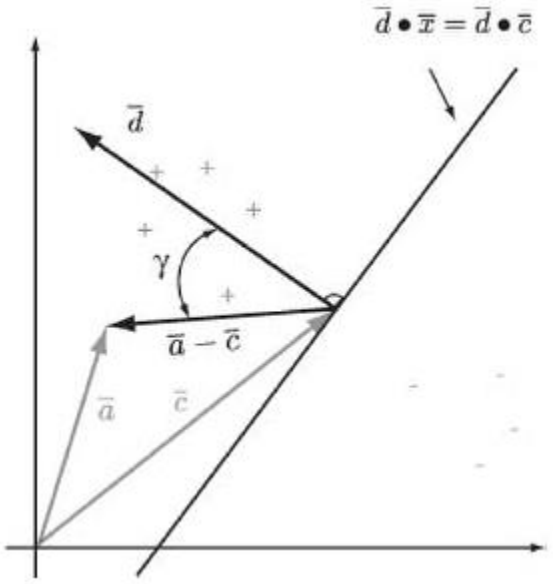
$$\hat{f}(\overline{x}) = \text{sgn}(\overline{d} \bullet \overline{x} - \overline{d} \bullet \overline{c})$$

$$\hat{f}(\overline{x}) = \text{sgn}((\overline{x} - \overline{c}) \bullet \overline{d})$$

$$= \text{sgn}(|\overline{x} - \overline{c}||\overline{d}| \cos \gamma),$$

Step5 (*cont.*)

- decision function for some point $\overline{x}$ is computed by taking the dot product between
  - the vector $\overline{x}$-$\overline{c}$, the "position vector" of the point $\overline{x}$ with respect to $\overline{c}$, and
  - the normal vector of the decision surface $\overline{d}$
- we derive a purely algebraic expression for the decision function in terms of the class means by plugging equations $\overline{d} = \overline{c}_+ - \overline{c}_-$ and $\overline{c} = \frac{1}{2}(\overline{c}_+ + \overline{c}_-)$ into $\hat{f}(\overline{x}) = \mathrm{sgn}((\overline{x} - \overline{c}) \bullet \overline{d})$



$$\begin{aligned}\hat{f}(\overline{x}) &= \mathrm{sgn}((\overline{x} - \overline{c}) \bullet \overline{d}) \\ &= \mathrm{sgn}\left(\left[\overline{x} - \tfrac{1}{2}(\overline{c}_+ + \overline{c}_-)\right] \bullet (\overline{c}_+ - \overline{c}_-)\right)\end{aligned}$$

InfoLab

Maximum-Margin Classifiers

DGIST 대구경북과학기술원
Daegu Gyeongbuk Institute of Science & Technology

Maximum-margin classifiers

- **New approach**
  - based on searching for a decision surface that is equidistant to the class boundaries where the two classes are closest to each other
  - maximizes the distances to these class boundaries
- **Reducing the probability of misclassification**
  - by placing the decision surface right in the middle between the two class boundaries
  - by maximizing the distances from the class boundaries
- **Constructing a maximum-margin classifier**
  - a convex optimization problem that can be solved via quadratic programming techniques
  - training set points: the heaviest constraints on the position of such an optimal decision surface, called support vectors

Optimization problems

- **Selecting the best solution from a number of possible or feasible solutions**
  - feasible solutions are ranked by an objective function
  - goal is to find the feasible solution that minimizes (or maximizes) the value of this function
- **Most optimization problems have a set of constraints that limit the solution space**
- **Problem :** $\min_{\overline{x}} \phi(\overline{x})$ **such that** $h_i(\overline{x}) \geq c_i$
  - $\phi(\overline{x})$: objective function
  - $h_i(\overline{x})$: constraint with bound c_i
  - optimization aim: find feasible solution $\overline{x}^*$ s.t. $\phi(\overline{x}^*) \leq \phi(\overline{q})$ for any other feasible solution $\overline{q}$

- **Turning a maximization problem into a minimization problem**

$$\max \phi(\overline{x}) = \min -\phi(\overline{x}),$$

$$\max \phi(\overline{x}) = \min \frac{1}{\phi(\overline{x})}$$

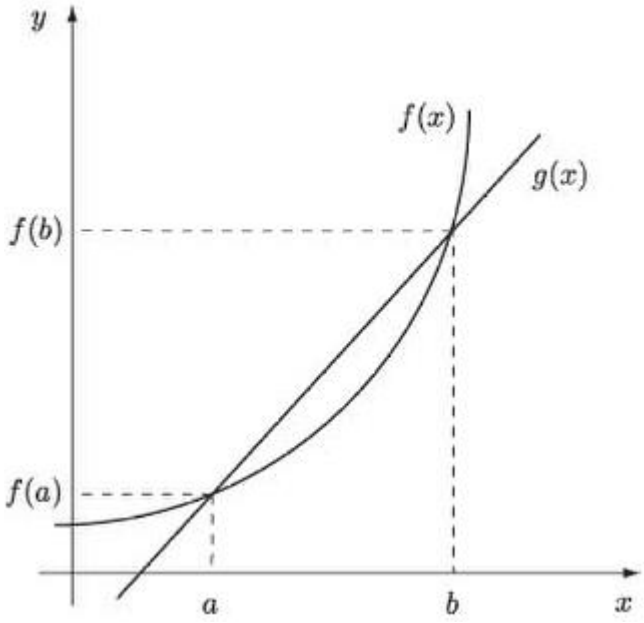
  - as long as $1/\phi(\overline{\boldsymbol{x}})$ is well-defined

- **Linear optimization problem**
  - both an objective function and constraints are linear
  - both represents lines, planes, or hyperplanes in the appropriate dot product spaces

- **Nonlinear optimization problem**
  - the objective function or the constraints are not linear

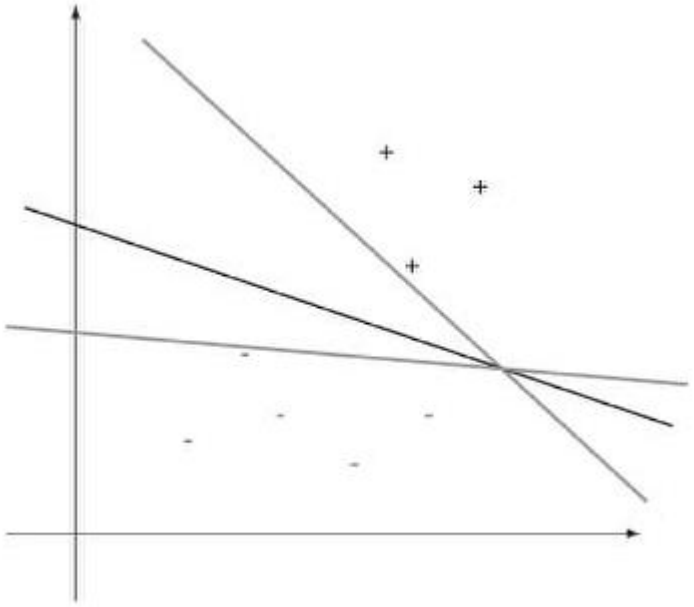
Convex optimization problem

- **A convex objective function and linear constraints**
- **Particularly well behaved in that**
  - the objective function has a global minimum
  - the function surface is smooth
    - we can draw a line from one point on the function surface to any other point on the surface without crossing the surface itself
- **Example**
  - f() is convex if f(x) ≤ g(x) in [a, b]
- **Quadratic programing technique**
  - take advantage of the convexity of an objective function in order to solve a convex optimization problem



Maximum margins

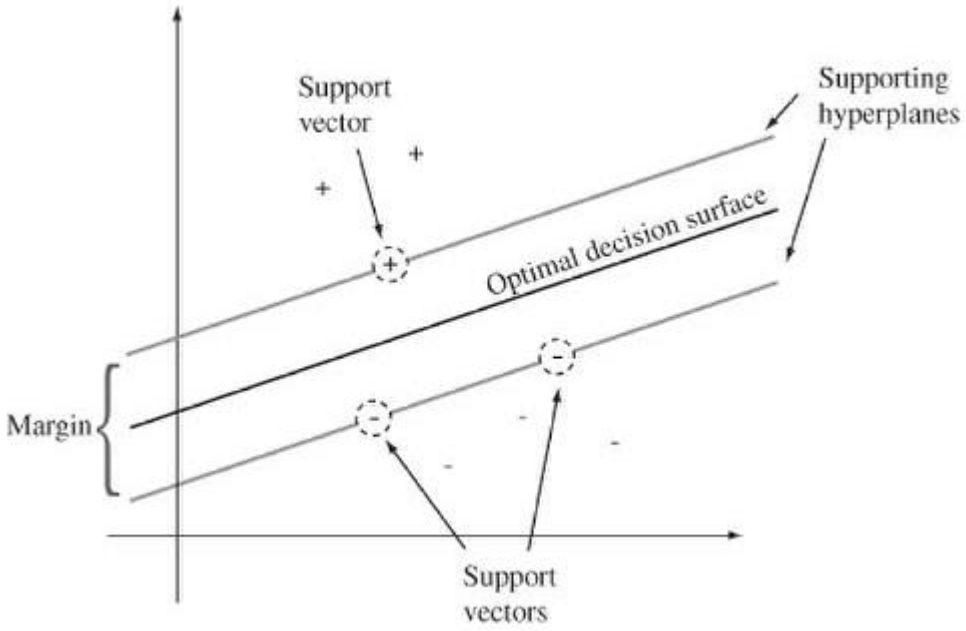
- **Intuition**
  - optimal decision surface is equidistant from the class boundaries given a linearly separable training set for a binary classification problem



- **Definition of supporting hyperplane**
  - a hyperplane supports a class if it is parallel to a (linear) decision surface and all points of its respective class are either above or below

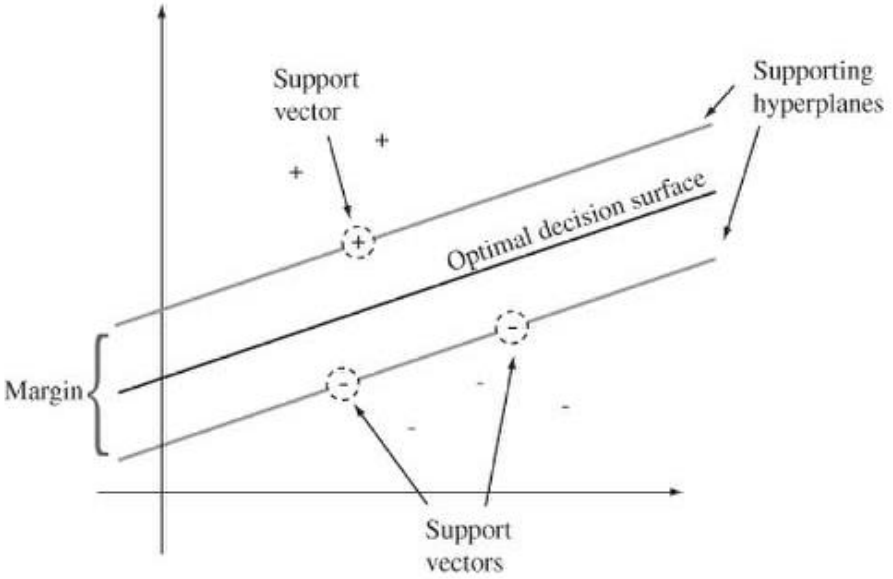
- **Two supporting hyperplanes**
  - one is translated in the direction of the class with the + 1 label
  - the other is translated in the direction of the class with the −1 label

- **Definition of margin**
  - the distance between the two supporting hyperplanes is called a margin in a binary classification problem



Definition of optimality

- a decision surface for a binary classification problem is optimal if it is equidistant from the two supporting hyperplanes and maximizes their margin



Properties

- we have the two supporting hyperplanes translated so they just touch their respective class boundaries
- the distance between the hyperplanes is the margin
- the optimal decision surface is located at the center of the margin
- the size of the margin is constrained by the circled points in each class, called support vectors
- any rotation or translation of the decision surface would result in a smaller margin

Optimizing the margin

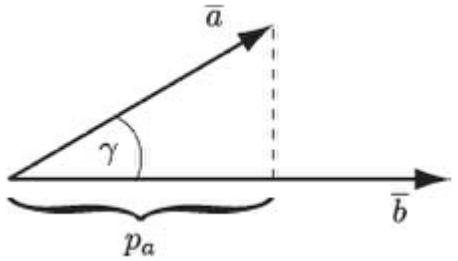
- **Optimization problem**
  - finding a decision surface that maximizes the margin between the two supporting hyperplanes
  - feasible solutions : all possible decision surfaces with their associated supporting hyperplanes
  - objective function : computes the size of the margin for each decision surface
  - constraints : the positions of the supporting hyperplanes, which are not allowed to cross their respective class boundaries

- **Formally maximum margin $m^* = \max \phi(\overline{w}, b)$**
  - objective function $\phi(\overline{w}, b)$: computes the margin of a given decision surface $\overline{w} \cdot \overline{x} = b$
  - maximum margin m^*: due to some optimal decision surface $\overline{w}^* \cdot \overline{x} = b^*$

- **To make this optimization problem computable**
  - we need to derive a suitable expression for the objective function $\boldsymbol{\phi}()$

- **Definition of projection**
  - let $\overline{\boldsymbol{a}}$ and $\overline{\boldsymbol{b}}$ be vectors in $\boldsymbol{R}^n$ that form an angle γ between them; then we say that p_a is the projection of $\overline{\boldsymbol{a}}$ in the direction of $\overline{\boldsymbol{b}}$ s.t.

$$p_a = |\overline{a}| \cos\gamma = \frac{\overline{a} \bullet \overline{b}}{|b|}$$

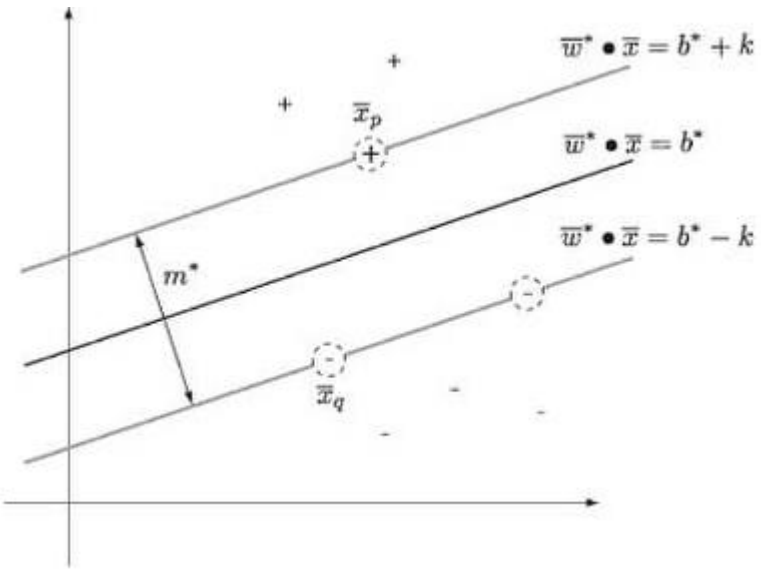


- **Our assumption**
  - linearly separable training set $D = \{(\overline{x}_1, y_1), (\overline{x}_2, y_2), \ldots, (\overline{x}_l, y_l)\} \subseteq \mathbb{R}^n \times \{+1, -1\}$
  - optimal decision surface $\overline{w}^* \bullet \overline{x} = b^*$

- **The following identities hold:** $m^* = \phi(\overline{w}^*, b^*) = \max \phi(\overline{w}, b)$
- **Maximum margin m***
  - can be computed by the objective function $\phi()$ given the parameters $\overline{w}^*$ and b* of the optimal decision surface
- **Two supporting hyperplanes equidistant from the optimal decision surface** $\overline{w}^* \bullet \overline{x} = b^*$

$$\overline{w}^* \bullet \overline{x} = b^* + k$$

$$\overline{w}^* \bullet \overline{x} = b^* - k$$



- **Support vectors lie on the supporting hyperplane**

$$\overline{w}^* \bullet \overline{x}_p = b^* + k$$

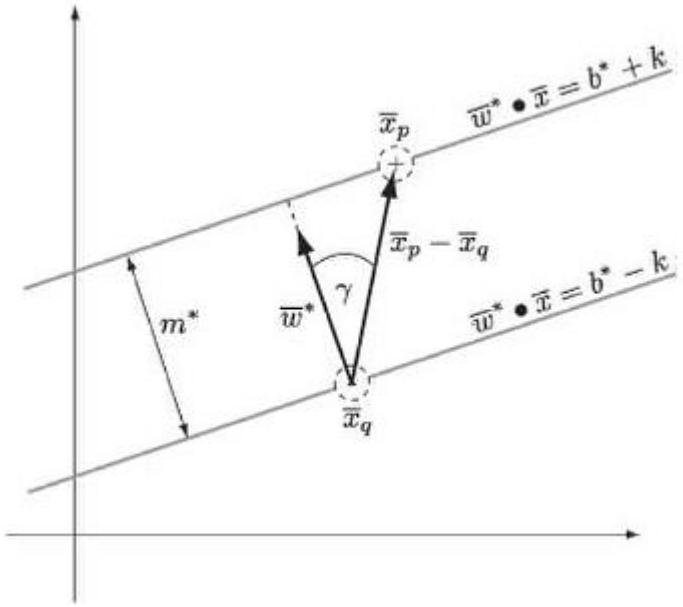
$$\overline{w}^* \bullet \overline{x}_q = b^* - k$$

Distance between two supporting planes

- the distance is equal to the margin m*
- it can be computed as the projection of the vector $\overline{x}_p - \overline{x}_q$ in the direction of $\overline{w}^*$
  - i.e., we can compute the margin as the projection of the difference between the two support vectors in the direction of the normal vector of the decision surface

$$
\begin{aligned}
m^* &= |\overline{x}_p - \overline{x}_q| \cos\gamma \\
&= \frac{\overline{w}^* \bullet (\overline{x}_p - \overline{x}_q)}{|\overline{w}^*|} \\
&= \frac{\overline{w}^* \bullet \overline{x}_p - \overline{w}^* \bullet \overline{x}_q}{|\overline{w}^*|} \\
&= \frac{(b^* + k) - (b^* - k)}{|\overline{w}^*|} \\
&= \frac{2k}{|\overline{w}^*|}.
\end{aligned}
$$

$$
m^* = \max \frac{2k}{|\overline{w}|}
$$



- **We want to express the maximization problem as a minimization problem**

$$
\begin{aligned}
m^* &= \max \frac{2k}{|\overline{w}|} \\
&= \min \frac{|\overline{w}|}{2k} \\
&= \min \frac{|\overline{w}|^2}{2k} \\
&= \min \tfrac{1}{2k} \overline{w} \bullet \overline{w} \\
&= \min \tfrac{1}{2} \overline{w} \bullet \overline{w}.
\end{aligned}
$$

  - the second step is justified since optimization over the positive values $|\overline{w}|$ is invariant under the transformation with the square function
    - square function preserves the order-theoretic properties of its domain
  - the last step is justified since optimization is invariant under scaling with a constant
    - free to pick a convenient value for k, and in our case we chose k = 1

- **Our objective function is now** $\phi(\overline{w}, b) = \frac{1}{2}\overline{w} \bullet \overline{w}$
  - The function does not have a term *b* to be optimized
  - The offset term *b* plays a role in the constraints

- **Shifting the focus to the constraints**
  - supporting hyperplanes are not allowed to cross their respective class boundaries

$$\overline{w}^* \bullet \overline{x}_i \geq b^* + k \qquad \text{for all } (\overline{x}_i, y_i) \in D \text{ s.t. } y_i = +1$$

$$\overline{w}^* \bullet \overline{x}_i \leq b^* - k \qquad \text{for all } (\overline{x}_i, y_i) \in D \text{ s.t. } y_i = -1$$

  - generalizing and taking to our choice of k = 1 into account gives us

$$\overline{w} \bullet \overline{x}_i \geq 1 + b \qquad \text{for all } (\overline{x}_i, y_i) \in D \text{ s.t. } y_i = +1,$$

$$\overline{w} \bullet (-\overline{x}_i) \geq 1 - b \qquad \text{for all } (\overline{x}_i, y_i) \in D \text{ s.t. } y_i = -1.$$

  - more compact form is

$$\overline{w} \bullet (y_i \overline{x}_i) \geq 1 + y_i b \qquad \text{for all } (\overline{x}_i, y_i) \in D.$$

Proposition : Maximum-Margin Classifier

- given a linearly separable training set

$$D = \{(\overline{x}_1, y_1), (\overline{x}_2, y_2), \ldots, (\overline{x}_l, y_l)\} \subseteq \mathbb{R}^n \times \{+1, -1\},$$

- we can compute a maximum-margin decision surface with an optimization

$$\overline{w}^* \bullet \overline{x} = b^*$$

$$\min \phi(\overline{w}, b) = \min_{\overline{w}, b} \frac{1}{2} \overline{w} \bullet \overline{w}$$

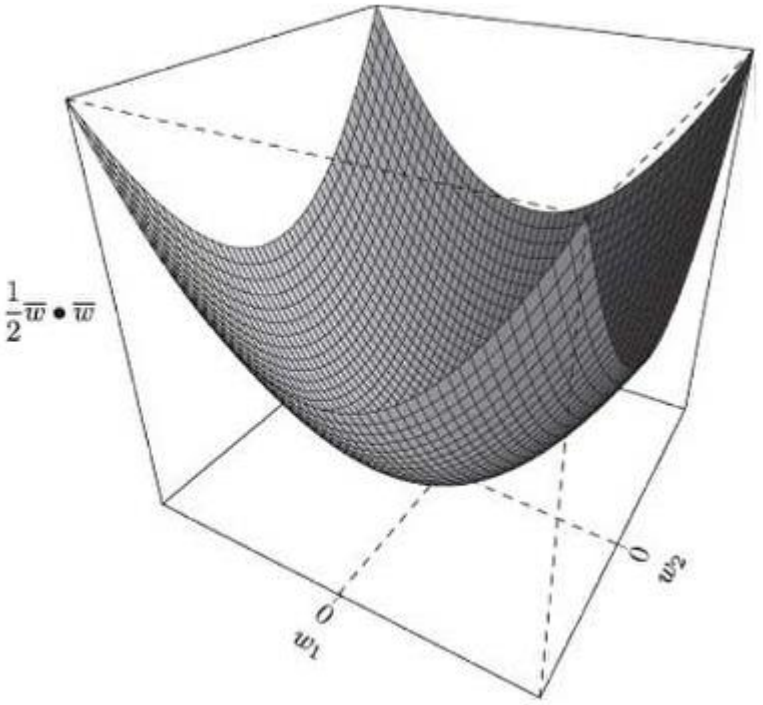
- subject to the constraints

$$\overline{w} \bullet (y_i \overline{x}_i) \geq 1 + y_i b \quad \text{for all } (\overline{x}_i, y_i) \in D$$

Quadratic programming

- **Our objective function is a convex function**

$$\phi(\overline{w}, b) = \frac{1}{2}\overline{w} \bullet \overline{w} = \frac{1}{2}(w_1^2 + \ldots + w_n^2)$$



  - convexity implies that we are able to find the global minimum of our objective function
  - an efficient way to solve convex optimization problems of the form given is via quadratic programming

$$\overline{w}^* = \text{solve}(\mathbf{Q}, \overline{q}, \mathbf{X}, \overline{c})$$

$$\overline{w}^* = \underset{\overline{w}}{\text{argmin}} \left(\frac{1}{2}\overline{w}^{\mathrm{T}}\mathbf{Q}\,\overline{w} - \overline{q} \bullet \overline{w}\right)$$ subject to the constraints $\mathbf{X}^{\mathrm{T}}\overline{w} \geq \overline{c}$

$$\overline{w}^* = \underset{\overline{w}}{\operatorname{argmin}} \left(\frac{1}{2} \overline{w}^{\mathrm{T}} \mathbf{Q} \, \overline{w} - \overline{q} \bullet \overline{w} \right)$$ subject to the constraints $\mathbf{X}^{\mathrm{T}} \overline{w} \geq \overline{c}$

- **Parameters**
  - Q : $n \times n$ matrix
  - X: $l \times n$ matrix
  - $\overline{w}^*$, $\overline{w}$, and $\overline{q}$: n-dimensional vectors
  - $\overline{c}$: l-dimensional vector

- **If we let Q be the identity matrix I and $\overline{q} = \overline{0}$**

$$\overline{w}^* = \underset{\overline{w}}{\operatorname{argmin}} \left(\frac{1}{2} \overline{w}^{\mathrm{T}} \mathbf{I} \, \overline{w} - \overline{0} \bullet \overline{w} \right) = \underset{\overline{w}}{\operatorname{argmin}} \left(\frac{1}{2} \overline{w} \bullet \overline{w} \right)$$

$$\overline{w}^{\mathrm{T}} \mathbf{I} \, \overline{w} = \overline{w} \bullet \overline{w}$$

- **Big difference between margin optimization and quadratic program solver**
  - quadratic program solvers return the argument that minimizes the objective function
    - rather than the minimized value of the objective function

Constraints in quadratic program solvers are expressed in matrix form

$$\mathbf{X}^{\mathrm{T}}\overline{w} \geq \overline{c}$$

$$\overline{w} \bullet (y_i \overline{x}_i) \geq 1 + y_i b \quad \text{for all } (\overline{x}_i, y_i) \in D$$

$$(y_i \overline{x}_i) \bullet \overline{w} \geq 1 + y_i b$$

$$\mathbf{X} = \begin{pmatrix} y_1 x_1^1 & \cdots & y_i x_i^1 & \cdots & y_l x_l^1 \\ \vdots & & \vdots & & \vdots \\ y_1 x_1^n & \cdots & y_i x_i^n & \cdots & y_l x_l^n \end{pmatrix}$$

$$\overline{c} = \begin{pmatrix} 1 + y_1 b \\ 1 + y_2 b \\ \vdots \\ 1 + y_l b \end{pmatrix}$$

- notice : free variable b in the quadratic program
  - need to take this free variable into account in our optimization problem
- optimization has to minimize the objective function over both $\overline{w}$ and b

Proposition

- given the linearly separable training set

$$D = \{(\overline{x}_1, y_1), (\overline{x}_2, y_2), \ldots, (\overline{x}_l, y_l)\} \subseteq \mathbb{R}^n \times \{+1, -1\}.$$

- we can compute a maximum-margin decision surface with a quadratic programming approach that solves the generalized optimization problem

$$(\overline{w}^*, b^*) = \underset{\overline{w}, b}{\operatorname{argmin}} \left(\tfrac{1}{2} \overline{w}^{\mathrm{T}} \mathbf{Q}\, \overline{w} - \overline{q} \bullet \overline{w} \right)$$

- subject to the constraints $\mathbf{X}^{\mathrm{T}} \overline{w} \geq \overline{c}$

- with Q = I, $\overline{\boldsymbol{q}} = \overline{\mathbf{0}}$, and

$$\mathbf{X} = \begin{pmatrix} y_1 x_1^1 & \cdots & y_i x_i^1 & \cdots & y_l x_l^1 \\ \vdots & & \vdots & & \vdots \\ y_1 x_1^n & \cdots & y_i x_i^n & \cdots & y_l x_l^n \end{pmatrix} \qquad \overline{c} = \begin{pmatrix} 1 + y_1 b \\ 1 + y_2 b \\ \vdots \\ 1 + y_l b \end{pmatrix}$$

Algorithm

- **Computing a decision surface with a maximum margin using a quadratic program solver**
  - repeated tests in the algorithm as to whether or not the solver was successful in finding a solution
  - r : radius of the training set D
  - q : size of the search interval for offset term values (we set it to 1000)
    - precise value is highly data dependent
    - needs to be determined experimentally

let $D = \{(\overline{x}_1, y_1), (\overline{x}_2, y_2), \ldots, (\overline{x}_l, y_l)\} \subset \mathbb{R}^n \times \{+1, -1\}$
$r \leftarrow \max\{|\overline{x}| \mid (\overline{x}, y) \in D\}$
$q \leftarrow 1000$
let $\overline{w}^*$ and b^* be undefined
Construct $\mathbf{X}$ according to (6.32) using D.
for each $b \in [-q, q]$ **do**
  Construct $\overline{c}$ according to (6.33) using b.
  $\overline{w} \leftarrow solve(\mathbf{I}, \overline{0}, \mathbf{X}, \overline{c})$
  if ($\overline{w}$ is defined **and** $\overline{w}^*$ is undefined) **or**
    ($\overline{w}$ is defined **and** $|\overline{w}| < |\overline{w}^*|$) **then**
    $\overline{w}^* \leftarrow \overline{w}$
    $b^* \leftarrow b$
  end if
end for
if $\overline{w}^*$ is undefined **then**
  stop constraints not satisfiable
else if $|\overline{w}|^* > q/r$ **then**
  stop bounding assumption of $|\overline{w}|$ violated
end if
return $(\overline{w}^*, b^*)$

- **To determine a reasonable interval of values for b during optimization**

$$b = \overline{w} \bullet \overline{x}.$$

$$b = |\overline{w}||\overline{x}| \cos\gamma$$

$$-|\overline{w}||\overline{x}| \le b \le |\overline{w}||\overline{x}|$$

$$-|\overline{w}|r \le b \le |\overline{w}|r$$

$$\begin{aligned} m^* &= |\overline{x}_p - \overline{x}_q| \cos\gamma \\ &= \frac{\overline{w}^* \bullet (\overline{x}_p - \overline{x}_q)}{|\overline{w}^*|} \\ &= \frac{\overline{w}^* \bullet \overline{x}_p - \overline{w}^* \bullet \overline{x}_q}{|\overline{w}^*|} \\ &= \frac{(b^* + k) - (b^* - k)}{|\overline{w}^*|} \\ &= \frac{2k}{|\overline{w}^*|}. \end{aligned}$$

- largest possible margin

$$\frac{2}{|\overline{w}|} \le 2r.$$

$$\frac{1}{r} \le |\overline{w}|$$

- we have $|\overline{w}| \to \infty$ for infinitesimally small margins
- q : a bounding constant that bounds the value of $|\overline{w}|$ to a multiple of the maximum margin 1/r

$$\frac{1}{r} \le |\overline{w}| \le \frac{q}{r}$$

$$-q \le b \le q$$

InfoLab

Support Vector Machines

DGIST 대구경북과학기술원
Daegu Gyeongbuk Institute of Science & Technology

Duality

- **Support vector machines can be viewed as the dual to the maximum-margin classifiers**
  - the dual is obtained by applying *Lagrangian optimization theory* to the maximum-margin classifier optimization problem

- **Linear classifiers based on support vector machines can easily be extended to nonlinear classifiers**
  - broadening the applicability of support vector machines tremendously
  - by applying what is referred to as the *kernel trick* to a linear support vector machine
  - retaining the efficiency of finding linear decision surfaces for training sets that are not linearly separable

Slack variables

- **Generalizing support vector machines by allowing the underlying maximum-margin classifier to make mistakes on the training set**
  - real-world training sets are not perfect and contain noise
  - noise might give rise to an extremely complicated boundary (decision surface) between the classes of a classification problem
  - allowing the classifier to ignore these points can compute a much simpler decision surface
  - simple decision surfaces have a much higher probability of classifying points correctly

- **Soft-margin classifiers**
  - maximum-margin classifiers that incorporate slack variables
  - c.f., hard-margin classifiers

Lagrangian dual

- **Convenient technique to derive the dual of an optimization problem**

- **Primal optimization problem** ***(similar with p.12 in Part I)***
  - optimization problem of the form $\min_{\bar{x}} \boldsymbol{\phi}(\bar{\boldsymbol{x}})$ such that $\boldsymbol{g_i}(\bar{\boldsymbol{x}}) \geq \mathbf{0}$
  - $\boldsymbol{\phi}(\bar{\boldsymbol{x}})$ is a convex objective function
  - constraints $\boldsymbol{g_i}(\bar{\boldsymbol{x}})$ are linear

- **Lagrangian optimization problem (new problem)**

$$\max_{\bar{\alpha}} \min_{\bar{x}} L(\bar{\alpha}, \bar{x}) = \max_{\bar{\alpha}} \min_{\bar{x}} \left(\phi(\bar{x}) - \sum_{i=1}^{l} \alpha_i g_i(\bar{x}) \right) \qquad \alpha_i \geq 0$$

$$\max_{\overline{\alpha}} \min_{\overline{x}} L(\overline{\alpha}, \overline{x}) = \max_{\overline{\alpha}} \min_{\overline{x}} \left(\phi(\overline{x}) - \sum_{i=1}^{l} \alpha_i g_i(\overline{x}) \right) \qquad \alpha_i \geq 0$$

- **New objective function $L(\overline{\alpha}, \overline{x})$ is called Lagrangian**
  - it incorporates the original objective function $\phi()$ together with a linear combination of the constraints $g_i()$
  - the values $\alpha_1, \ldots, \alpha_l$ are called the Lagrangian multipliers

$$\overline{\alpha} = (\alpha_1, \alpha_2, \ldots, \alpha_l)$$

  - we have exactly one Lagrangian multiplier α_i for each constraint g_i
  - we call $\overline{x}$ the primal variable and $\overline{\alpha}$ the dual variable

- **Two nested optimization operators with opposing optimization objectives**
  - each optimization operator returns a partially evaluated function to be optimized by the other optimization operator
  - if we fix the vector $\overline{x}$ to the value $\overline{x}^*$, then the optimization problem becomes the maximization problem

$$\max_{\overline{\alpha}} L(\overline{\alpha}, \overline{x}^*) = \max_{\overline{\alpha}} \left(\phi(\overline{x}^*) - \sum_{i=1}^{l} \alpha_i g_i(\overline{x}^*) \right)$$

  - if we fix the vector $\overline{\alpha}$ to the value $\overline{\alpha}^*$, then the optimization problem becomes the minimization problem

$$\min_{\overline{x}} L(\overline{\alpha}^*, \overline{x}) = \min_{\overline{x}} \left(\phi(\overline{x}) - \sum_{i=1}^{l} \alpha_i^* g_i(\overline{x}) \right)$$

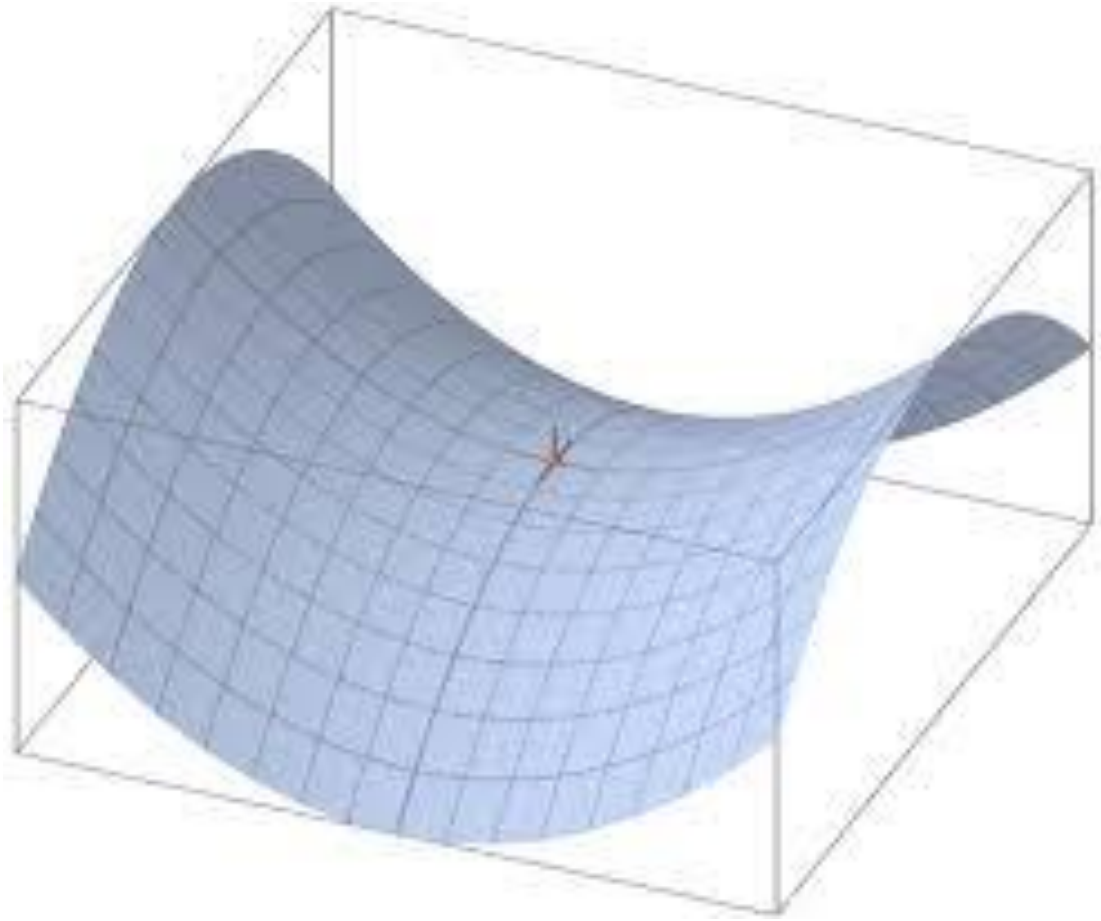
Solutions to the Lagrangian optimization

- points that both maximize the function $L(\bar{\alpha}, \bar{x})$ with respect to the dual variable $\bar{\alpha}$ and minimize it with respect to the primal variable $\bar{x}$
- the solutions are saddle points on the graph of the function $L(\bar{\alpha}, \bar{x})$
- we have a unique saddle point
  - since we assume that the primal objective function $\phi(\bar{x})$ is convex and the constraints $g_i(\bar{x})$ are linear
- the partial derivative of $L(\bar{\alpha}, \bar{x})$ with respect to $\bar{x}$ at the saddle point has to be zero

$$\frac{\partial L}{\partial \bar{x}} = \bar{0} \qquad \frac{\partial L}{\partial \bar{x}}(\bar{\alpha}, \bar{x}^*) = \bar{0},$$

- the point $\bar{x}^*$ represents an optimum of $L(\bar{\alpha}, \bar{x})$ with respect to $\bar{x}$

Saddle point



- **Let $\overline{\alpha}^*$ and $\overline{x}^*$ be a solution to the Lagrangian such that**

$$\max_{\overline{\alpha}} \min_{\overline{x}} L(\overline{\alpha}, \overline{x}) = L(\overline{\alpha}^*, \overline{x}^*) = \phi(\overline{x}^*) - \sum_{i=1}^{l} \alpha_i^* g_i(\overline{x}^*)$$

- **Then, $\overline{x}^*$ is a solution to the primal objective function if and only if the following conditions hold:**

$$\frac{\partial L}{\partial \overline{x}}(\overline{\alpha}^*, \overline{x}^*) = \overline{0},$$

$$\alpha_i^* g_i(\overline{x}^*) = 0,$$

$$g_i(\overline{x}^*) \geq 0,$$

$$\alpha_i^* \geq 0$$

  - each constraint $g_i()$ evaluated at $\overline{x}^*$ and multiplied by its corresponding Lagrangian multiplier α_i^* has to result in a value zero
  - the term $\sum_{i=1}^{l} \alpha_i^* g_i(\overline{x}^*)$ has to be vanish so that $L(\overline{\alpha}^*, \overline{x}^*) = \phi(\overline{x}^*)$

Karush-Kuhn-Tucker(KKT) conditions

- condition1: the value $\overline{x}^*$ lies on the saddle point

$$\frac{\partial L}{\partial \overline{x}}(\overline{\alpha}^*, \overline{x}^*) = \overline{0}$$

- condition2, 3: original constraints of the primal and Lagrangian optimization problems
  - ensure that the points $\overline{\alpha}^*$ and $\overline{x}^*$ lie in the respective feasible regions

$$g_i(\overline{x}^*) \geq 0$$

$$\alpha_i^* \geq 0$$

KKT complementarity condition

$$\alpha_i^* g_i(\overline{x}^*) = 0$$

- **Solving equation** $\frac{\partial L}{\partial \overline{x}}(\overline{\alpha}, \overline{x}^*) = \overline{0}$ **for** $\overline{x}^*$
  - reformulate the original optimization problem in terms of its dual variable only

$$L(\overline{\alpha}, \overline{x}^*) = \phi'(\overline{\alpha})$$

  - find the optimum with respect to the dual variable as the Lagrangian optimization

$$\max_{\overline{\alpha}} \phi'(\overline{\alpha}) \qquad \alpha_i \geq 0$$

- **Lagrangian dual function** ϕ'
  - we can solve our primal optimization problem using the Lagrangian dual

$$\max_{\overline{\alpha}} \phi'(\overline{\alpha}) = \phi'(\overline{\alpha}^*) = L(\overline{\alpha}^*, \overline{x}^*) = \phi(\overline{x}^*)$$

  - where $\overline{\alpha}^*$ and $\overline{x}^*$ have to satisfy the KKT conditions

Example

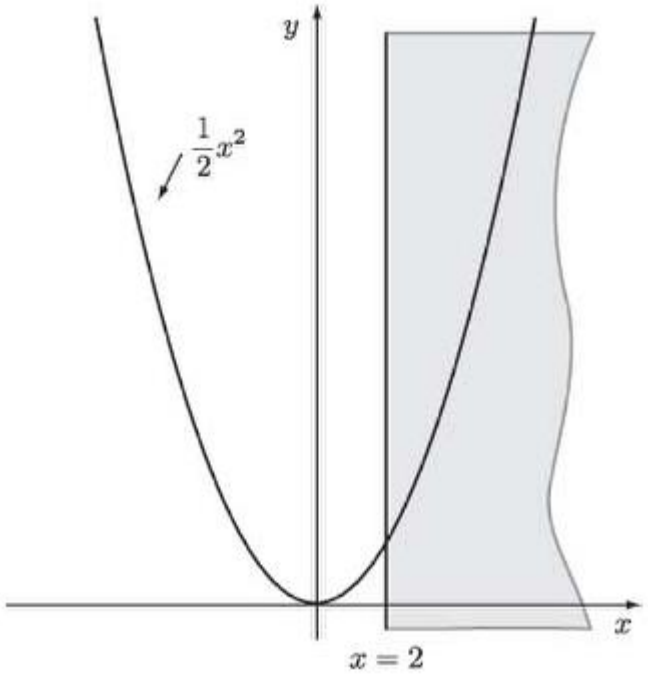
- **Consider the convex optimization problem** $\min \phi(x) = \min \frac{1}{2}x^2$
  - subject to the linear constraint

    $g(x) = x - 2 \geq 0$

- **Standard technique of finding the minimum**
  - by taking the derivative of ϕ with respect to x and setting it to zero

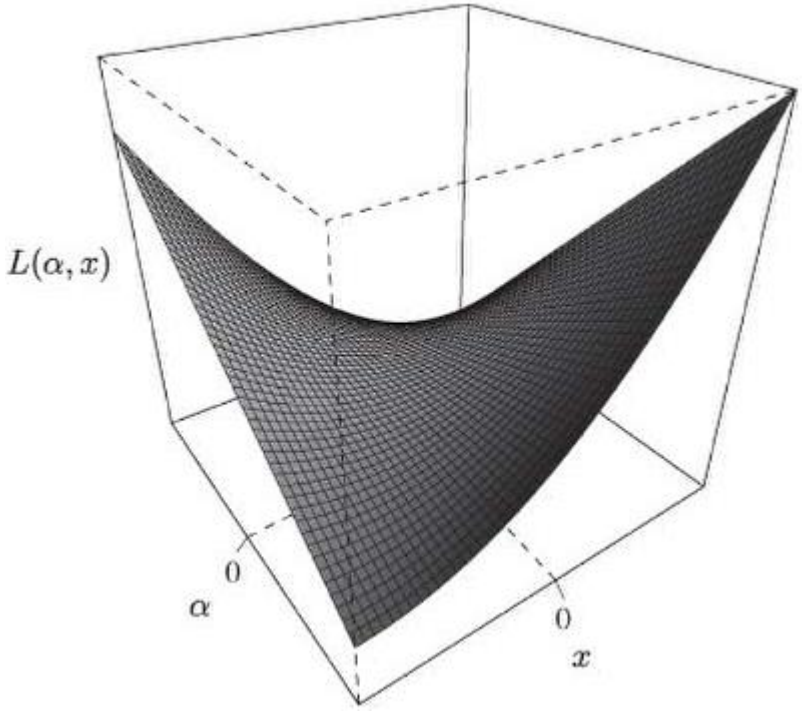
    $\frac{d\phi}{dx} = 0$

    fails because the value x = 0 that gives rise to the minimum is not part of the feasible region



Lagrangian

$$L(\alpha, x) = \tfrac{1}{2}x^2 - \alpha(x-2)$$



- saddle point has to occur where the gradient of the Lagrangian with respect to the variable x is equal to zero

$$\frac{\partial L}{\partial x}(\alpha, x^*) = x^* - \alpha = 0$$

- solving for x* gives us

$$x^* = \alpha$$

$$L(\alpha, x^*) = \tfrac{1}{2}\alpha^2 - \alpha^2 + 2\alpha = 2\alpha - \tfrac{1}{2}\alpha^2$$

- Lagrangian dual optimization with $\phi'(\alpha) = L(\alpha, x^*)$

$$\max_{\alpha} \phi'(\alpha) = \max_{\alpha} \left(2\alpha - \tfrac{1}{2}\alpha^2\right) \qquad \alpha \geq 0$$

- compute the value α* $\quad \frac{d\phi'}{d\alpha}(\alpha^*) = 2 - \alpha^* = 0 \quad$ α* = 2 $\quad \alpha^* g(x^*) = \alpha^*(x^* - 2) = 2(2-2) = 0$

Dual Maximum-margin optimization

- **Support vector machines can be seen as the dual to maximum-margin classifiers**
  - derive this dual by applying the technique of the Lagrangian dual to maximum-margin classifiers
  - rewrite maximum-margin optimization problem in a form appropriate for Lagrangian optimization

$$\min_{\overline{w},b} \phi(\overline{w}, b) = \min_{\overline{w},b} \tfrac{1}{2}\overline{w} \bullet \overline{w}$$

$$g_i(\overline{w}, b) = y_i(\overline{w} \bullet \overline{x}_i - b) - 1 \geq 0$$

$$\begin{aligned} L(\overline{\alpha}, \overline{w}, b) &= \phi(\overline{w}, b) - \sum_{i=1}^{l} \alpha_i g_i(\overline{w}, b) \\ &= \frac{1}{2}\overline{w} \bullet \overline{w} - \sum_{i=1}^{l} \alpha_i (y_i(\overline{w} \bullet \overline{x}_i - b) - 1) \\ &= \frac{1}{2}\overline{w} \bullet \overline{w} - \sum_{i=1}^{l} \alpha_i y_i \overline{w} \bullet \overline{x}_i + b\sum_{i=1}^{l} \alpha_i y_i + \sum_{i=1}^{l} \alpha_i \end{aligned}$$

Lagrangian optimization problem for maximum-margin classifiers

$$\max_{\overline{\alpha}} \min_{\overline{w},b} L(\overline{\alpha}, \overline{w}, b) \qquad \alpha_i \geq 0$$

- let $\overline{\alpha}^*$, $\overline{w}^*$, and b* be a solution to the Lagrangian optimization problem such that

$$\max_{\overline{\alpha}} \min_{\overline{w},b} L(\overline{\alpha}, \overline{w}, b) = L(\overline{\alpha}^*, \overline{w}^*, b^*)$$

- since $\boldsymbol{\phi}$ is convex and the constraints $\boldsymbol{g_i}$ are linear, the solution $\overline{\alpha}^*$, $\overline{w}^*$, and b* satisfy the KKT conditions

$$\frac{\partial L}{\partial \overline{w}}(\overline{\alpha}^*, \overline{w}^*, b^*) = \overline{0},$$

$$\frac{\partial L}{\partial b}(\overline{\alpha}^*, \overline{w}^*, b^*) = 0,$$

assure that $\overline{w}^*$ and b* lie on the saddle point of the Lagrangian

$$\alpha_i^*(y_i(\overline{w}^* \bullet \overline{x}_i - b^*) - 1) = 0,$$

complementarity condition

$$y_i(\overline{w}^* \bullet \overline{x}_i - b^*) - 1 \geq 0,$$

$$\alpha_i^* \geq 0$$

- **Complementarity condition implies that $\overline{w}^*$ and b* are also solutions for the primal optimization problem**

$$\max_{\overline{\alpha}} \min_{\overline{w},b} L(\overline{\alpha}, \overline{w}, b) = L(\overline{\alpha}^*, \overline{w}^*, b^*) = \phi(\overline{w}^*, b^*)$$

- **To solve the Lagrangian optimization, we construct the Lagrangian dual**
  - applying the KKT condition; taking the partial derivative of Lagrangian *L* with respect to the primal variable $\overline{w}$; evaluating it at the saddle point $\overline{w}^*$, and setting it to zero:

$$\frac{\partial L}{\partial \overline{w}}(\overline{\alpha}, \overline{w}^*, b) = \overline{w}^* - \sum_{i=1}^{l} \alpha_i y_i \overline{x}_i = \overline{0}$$

$$\frac{\partial L}{\partial \overline{w}}(\overline{\alpha}^*, \overline{w}^*, b^*) = \overline{0}$$

$$\overline{w}^* = \sum_{i=1}^{l} \alpha_i y_i \overline{x}_i$$

  - for *b*

$$\frac{\partial L}{\partial b}(\overline{\alpha}, \overline{w}, b^*) = \sum_{i=1}^{l} \alpha_i y_i = 0$$

$$\frac{\partial L}{\partial b}(\overline{\alpha}^*, \overline{w}^*, b^*) = 0$$

In a maximum-margin classifier

- the supporting hyperplane for the +1 class has to go through some point $\overline{x}_p$ +1 closest to the class boundary
- this allows us to compute the offset b^+ of the supporting hyperplane for class +1 as

$$b^+ = \overline{w}^* \bullet \overline{x}_p$$

- for a given rotation $\overline{w}^*$, the point $\overline{x}_p$ closest to the class boundary will produce the smallest offset
  - that is, we can compute b^+ as an optimization as follows:

$$b^+ = \min\{\overline{w}^* \bullet \overline{x} \mid (\overline{x}, y) \in D \text{ with } y = +1\}$$

- applying similar reasoning to the class −1

$$b^- = \max\{\overline{w}^* \bullet \overline{x} \mid (\overline{x}, y) \in D \text{ with } y = -1\}$$

- decision surface is located right between the two supporting hyperplanes

$$b^* = \frac{b^+ + b^-}{2}$$

- **Both $\overline{w}^*$ and b* can be expressed in terms of the dual variable by repeated use of** $\overline{w}^* = \sum_{i=1}^{l} \alpha_i y_i \overline{x}_i$
  - optimal decision surface $\overline{w}^* \bullet \overline{x} = b^*$ is completely determined by the value of $\overline{\alpha}$
  - finding a solution $\overline{\alpha}^*$ will give us our decision surface
    - we can find a solution $\overline{\alpha}^*$ by solving the Lagrangian dual

- **Constructing Lagrangian dual**
  - substituting $\overline{w}^* = \sum_{i=1}^{l} \alpha_i y_i \overline{x}_i$ into

$$
\begin{aligned}
L(\overline{\alpha}, \overline{w}, b) &= \phi(\overline{w}, b) - \sum_{i=1}^{l} \alpha_i g_i(\overline{w}, b) \\
&= \frac{1}{2} \overline{w} \bullet \overline{w} - \sum_{i=1}^{l} \alpha_i (y_i (\overline{w} \bullet \overline{x}_i - b) - 1) \\
&= \frac{1}{2} \overline{w} \bullet \overline{w} - \sum_{i=1}^{l} \alpha_i y_i \overline{w} \bullet \overline{x}_i + b \sum_{i=1}^{l} \alpha_i y_i + \sum_{i=1}^{l} \alpha_i
\end{aligned}
$$

  - applying the constraint $\frac{\partial L}{\partial b}(\overline{\alpha}, \overline{w}, b^*) = \sum_{i=1}^{l} \alpha_i y_i = 0$

$$
\phi'(\overline{\alpha}) = L(\overline{\alpha}, \overline{w}^*, b^*) = \sum_{i=1}^{l} \alpha_i - \frac{1}{2} \sum_{i=1}^{l} \sum_{j=1}^{l} \alpha_i \alpha_j y_i y_j \overline{x}_i \bullet \overline{x}_j
$$

Proposition (Maximum-Margin Lagrangian Dual)

- given the maximum-margin optimization as in the previous Proposition, the Lagrangian dual optimization for maximum-margin classifiers is

$$\max_{\overline{\alpha}} \phi'(\overline{\alpha}) = \max_{\overline{\alpha}} \left(\sum_{i=1}^{l} \alpha_i - \frac{1}{2} \sum_{i=1}^{l} \sum_{j=1}^{l} \alpha_i \alpha_j y_i y_j \overline{x}_i \bullet \overline{x}_j \right)$$

- subject to the constraints

$$\sum_{i=1}^{l} \alpha_i y_i = 0, \qquad \alpha_i \geq 0$$

- **Case1: $\alpha_j^* > 0$ for some point $(\overline{x}_j, y_j) \in$ D**
  - to satisfy the complementarity condition we have $y_j(\overline{w}^* \bullet \overline{x}_j - b^*) - 1 = 0$
  - or
$$\overline{w}^* \bullet \overline{x}_j = b^* + 1 \quad \text{if } y_j = +1$$
$$\overline{w}^* \bullet \overline{x}_j = b^* - 1 \quad \text{if } y_j = -1$$

- **Training set point $(\overline{x}_j, y_j)$ with a nonzero Lagrangian multiplier $\alpha_j^* > 0$ lies on one of the two supporting hyperplanes**
  - it represents a constraint on the margin in that the supporting hyperplanes cannot be moved beyond it
  - we call points with nonzero Lagrangian multipliers support vectors
  - only support vectors contribute to the solution of the dual maximum-margin optimization from $\overline{w}^* = \sum_{i=1}^{l} \alpha_i y_i \overline{x}_i$ and $b^* = \frac{b^+ + b^-}{2}$

- **Case2: $\alpha_j^* = 0$ for some point $(\overline{x}_j, y_j) \in$ D**
  - the point $\overline{x}_j$ is a point that does not lie in the vicinity of the class boundary because we have $y_j(\overline{w}^* \bullet \overline{x}_j - b^*) - 1 > 0$

    or $\overline{w}^* \bullet \overline{x}_j > b^* + 1 \quad \text{if } y_j = +1$

    $\overline{w}^* \bullet \overline{x}_j < b^* - 1 \quad \text{if } y_j = -1$
  - this implies that points with zero-valued Lagrangian multipliers do not constrain the size of the margin

- **Recall**
  - the primal maximum-margin optimization problem finds the respective supporting hyperplanes that are farthest apart
    - i.e., that create the maximum margin between them
  - the points in the training set that limit the size of the margin were called support vectors

Statement

- the primal maximum-margin optimization computes the supporting hyperplanes whose margin is limited by support vectors
- the dual maximum-margin optimization computes the support vectors that limit the size of the margin of the supporting hyperplanes

Insight

- only support vectors contribute to our dual solution allows us to express the value for b* in a more elegant way
- we already know which training set points constitute the constraints on the supporting hyper-planes
  - the points with non-zero Lagrangian multipliers
- we can compute b^* as $$b^* = \overline{w}^* \bullet \overline{x}_{sv^+} - 1 = \sum_{i=1}^{l} \alpha_i^* y_i \overline{x}_i \bullet \overline{x}_{sv^+} - 1$$

Dual decision function

- **Recall**
  - decision functions in linear classifiers are based on linear decision surfaces
  - decision function itself returns a +1(*or -1*) label for a point that lies above(*or below*) the decision surface

- **Optimal decision surface** $\overline{w}^* \bullet \overline{x} = b^*$
  - maximum-margin decision function $\hat{f}(\overline{x}) = \text{sgn}(\overline{w}^* \bullet \overline{x} - b^*)$

$$\hat{f}(\overline{x}) = \text{sgn}\left(\sum_{i=1}^{l} \alpha_i^* y_i \overline{x}_i \bullet \overline{x} - \sum_{i=1}^{l} \alpha_i^* y_i \overline{x}_i \bullet \overline{x}_{sv^+} + 1\right)$$

  - **support vector machine**: dual maximum-margin classifier is determined completely by the support vectors

Linear support vector machine

- **Given**
  - A dot product space $\mathbb{R}^n$ as our data universe with points $\overline{x} \in \mathbb{R}^n$
  - Some target function $f : \mathbb{R}^n \rightarrow \{+1, -1\}$
  - A labeled, linearly separable training set

$$D = \{(\overline{x}_1, y_1), (\overline{x}_2, y_2), \ldots, (\overline{x}_l, y_l)\} \subseteq \mathbb{R}^n \times \{+1, -1\},$$

where $y_i = f(\overline{x}_i)$

- **Compute a model $\hat{f}$: $R^n \rightarrow \{+1, -1\}$ using D such that $\hat{f}(\overline{x}) \cong f(\overline{x})$**

$$\hat{f}(\overline{x}) = \operatorname{sgn}\left(\sum_{i=1}^{l} \alpha_i^* y_i \overline{x}_i \bullet \overline{x} - \sum_{i=1}^{l} \alpha_i^* y_i \overline{x}_i \bullet \overline{x}_{sv^+} + 1\right)$$

  - ➢ pick one support vector from the set of available support vectors

$$(\overline{x}_{sv^+}, +1) \in \{(\overline{x}_i, +1) \mid (\overline{x}_i, +1) \in D \text{ and } \alpha_i^* > 0\}$$

- **Train our support vector models with the Lagrangian dual optimization for maximum-margin classifiers**

$$\overline{\alpha}^* = \underset{\overline{\alpha}}{\operatorname{argmax}} \left(\sum_{i=1}^{l} \alpha_i - \frac{1}{2} \sum_{i=1}^{l} \sum_{j=1}^{l} \alpha_i \alpha_j y_i y_j \overline{x}_i \bullet \overline{x}_j \right)$$

  - subject to the constraints

$$\sum_{i=1}^{l} \alpha_i y_i = 0, \qquad \alpha_i \geq 0$$

- **We can solve our classification problem using linear support vector machines as long as the training data are linearly separable**

Nonlinear support vector machines

- **Only very few real data sets are linearly separable**
- **Support vector machines**
  - basic linear framework is easily extended to the case where the data set is not linearly separable
  - fundamental idea is to transform the input space into a higher-dimensional space called a feature space
    - input space: the data set is not linearly separable
    - feature space: the data are linearly separable
  - all the computations associated with the feature space can be performed in the input space

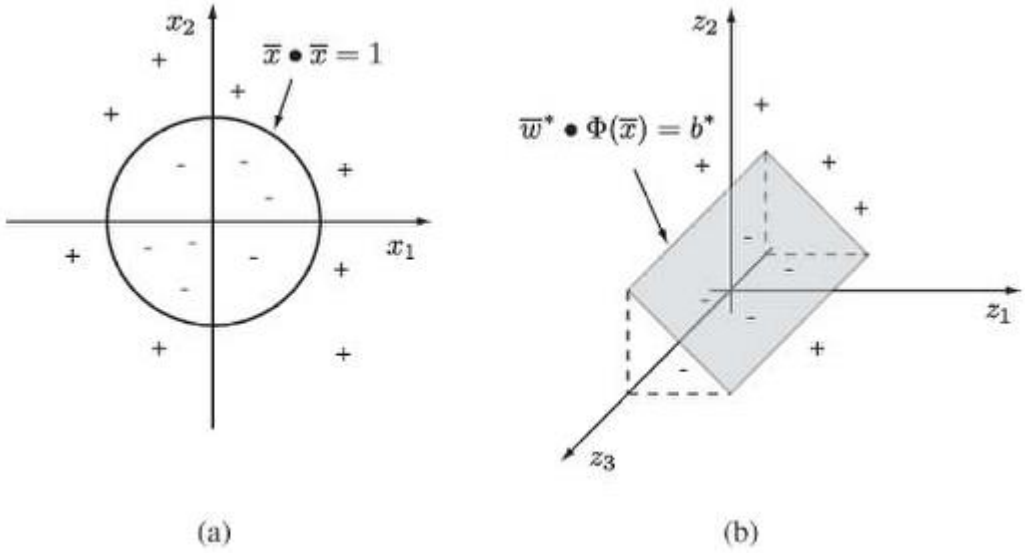
  - **kernel function**: the functions associated with transformations
  - **kernel trick**: the process of using kernel functions to move from a linear to a nonlinear support vector machine

Example

- There is no linear decision surface of the form $\overline{w} \bullet \overline{x} = b$
- In contrast, the nonlinear decision surface $\overline{x} \bullet \overline{x} = 1$
- Decision surface in a higher-dimensional space

$$\hat{f}(\overline{x}) = \text{sgn}(\overline{w} \bullet \Phi(\overline{x}) - b)$$

- Kernel function: $\Phi(\overline{x}) = \Phi(x_1, x_2) = (x_1^2, x_2^2, \sqrt{2}x_1x_2) = (z_1, z_2, z_3) = \overline{z}$



$\overline{w}^* \bullet \Phi(\overline{x}) = b^*$ *plane*

$\overline{w}^* = (w_1^*, w_2^*, w_3^*) = (1, 1, 0)$

$b^* = 1$

- **Point $\overline{q}$ = (1, 0) in input space**
  - clearly lies on the nonlinear decision surface $\overline{q} \bullet \overline{q} = (1, 0) \bullet (1, 0) = 1^2 + 0^2 = 1$
  - also lies on the plane in feature space:

$$\begin{aligned}\overline{w}^* \bullet \Phi(\overline{q}) &= (1, 1, 0) \bullet (1^2, 0^2, \sqrt{2} \times 1 \times 0) \\ &= (1, 1, 0) \bullet (1, 0, 0) \\ &= 1^2 + 1 \times 0 + 0 \times 0 \\ &= 1 \\ &= b^*.\end{aligned}$$

- **Decision surface** $\hat{f}(\overline{x}) = \operatorname{sgn}\left(\overline{w}^* \bullet \Phi(\overline{x}) - b^*\right)$

$$\begin{aligned}\hat{f}(\overline{x}) &= \operatorname{sgn}\left(\overline{w}^* \bullet \Phi(\overline{x}) - b^*\right) \\ &= \operatorname{sgn}\left(w_1^* x_1^2 + w_2^* x_2^2 + w_3^* \sqrt{2} x_1 x_2 - b^*\right) \\ &= \operatorname{sgn}\left(\overline{w}^* \bullet \overline{z} - b^*\right) \\ &= \operatorname{sgn}\left(\sum_{i=1}^{3} w_i^* z_i - b^*\right).\end{aligned}$$

$$\Phi(\overline{x}) = \Phi(x_1, x_2) = (x_1^2, x_2^2, \sqrt{2} x_1 x_2) = (z_1, z_2, z_3) = \overline{z}$$

  - this shows that the complexity of the decision function is directly related to the number of dimensions in the feature space

$$\hat{f}(\overline{x}) = \operatorname{sgn}\left(\sum_{i=1}^{d} w_i z_i - b\right)$$

Kernel trick

- **Dual representation of the normal vector of our decision function** $\hat{f}(\overline{x}) = \operatorname{sgn}(\overline{w}^* \bullet \Phi(\overline{x}) - b^*)$

$$\overline{w}^* = \sum_{i=1}^{l} \alpha_i^* y_i \Phi(\overline{x}_i) \qquad \Phi : \mathbb{R}^2 \to \mathbb{R}^3$$

  - assume that the values α_i^* represent the appropriate Lagrangian multipliers for this dual representation
  - notice that the transformation of the training points $\Phi(\overline{x}_i)$ is necessary since $\overline{w}^*$ is a normal vector in the feature space

- **Plugging dual representation into decision function**

$$\hat{f}(\overline{x}) = \operatorname{sgn}\left(\overline{w}^* \bullet \Phi(\overline{x}) - b^*\right)$$
$$= \operatorname{sgn}\left(\sum_{i=1}^{l} \alpha_i^* y_i \Phi(\overline{x}_i) \bullet \Phi(\overline{x}) - b^*\right)$$

- **Simplifying it using the calculations in the table**

$$\hat{f}(\overline{x}) = \operatorname{sgn}\left(\sum_{i=1}^{l} \alpha_i^* y_i \Phi(\overline{x}_i) \bullet \Phi(\overline{x}) - b^*\right)$$
$$= \operatorname{sgn}\left(\sum_{i=1}^{l} \alpha_i^* y_i (\overline{x}_i \bullet \overline{x})^2 - b^*\right).$$

Given the mapping $\Phi : \mathbb{R}^2 \to \mathbb{R}^3$ defined as $\Phi(\overline{x}) = (x_1^2, x_2^2, \sqrt{2}x_1x_2)$, the value of the dot product $\Phi(\overline{x}) \bullet \Phi(\overline{y})$ with $\overline{x}, \overline{y} \in \mathbb{R}^2$ can be computed in the input space $\mathbb{R}^2$:

$$\Phi(\overline{x}) \bullet \Phi(\overline{y}) = (x_1^2, x_2^2, \sqrt{2}x_1x_2) \bullet (y_1^2, y_2^2, \sqrt{2}y_1y_2)$$
$$= x_1^2y_1^2 + x_2^2y_2^2 + 2x_1x_2y_1y_2$$
$$= (x_1y_1 + x_2y_2)(x_1y_1 + x_2y_2)$$
$$= (\overline{x} \bullet \overline{y})(\overline{x} \bullet \overline{y})$$
$$= (\overline{x} \bullet \overline{y})^2.$$

- **Obtaining an expression whose complexity is proportional to the number of support vectors**
  - instead of obtaining a function whose complexity is proportional to the dimensions of the feature space

Given an appropriate mapping $\Phi : R^n \rightarrow R^m$ $(m \geq n)$

- functions of the form $k(\bar{x}, \bar{y}) = \Phi(\bar{x}) \bullet \Phi(\bar{y})$ where $\bar{x}, \bar{y} \in \mathbb{R}^n$ are called kernels or kernel functions

- kernel functions evaluate a dot product in feature space
- characteristic of a kernel : the value of this dot product is actually computed in the input space

Rewriting decision function in terms of kernel functions

$$\hat{f}(\bar{x}) = \operatorname{sgn}\left(\bar{w}^* \bullet \Phi(\bar{x}) - b^*\right) = \operatorname{sgn}\left(\sum_{i=1}^{l} \alpha_i^* y_i \Phi(\bar{x}_i) \bullet \Phi(\bar{x}) - b^*\right)$$

kernel trick

$$\hat{f}(\bar{x}) = \operatorname{sgn}\left(\sum_{i=1}^{n} \alpha_i^* y_i k(\bar{x}_i, \bar{x}) - b^*\right)$$

Appropriate kernel function

- we can take advantages of mappings into feature spaces without having to pay the price of actually having to compute the explicit mappings
  - the computations in the feature space always simplify to computations in the input space
- we can control the complexity of this model by selecting the kernel function judiciously
- trick lies in finding the appropriate kernel in order to construct a model for a particular data set

Types of kernels

Kernel Name	Kernel Function[a]	Free Parameters
Linear kernel	$k(\overline{x}, \overline{y}) = \overline{x} \bullet \overline{y}$	none
Homogeneous polynomial kernel	$k(\overline{x}, \overline{y}) = (\overline{x} \bullet \overline{y})^d$	$d \geq 2$
Nonhomogeneous polynomial kernel	$k(\overline{x}, \overline{y}) = (\overline{x} \bullet \overline{y} + c)^d$	$d \geq 2$, $c > 0$
Gaussian kernel	$k(\overline{x}, \overline{y}) = e^{-(\lvert\overline{x}-\overline{y}\rvert^2/2\sigma^2)}$	$\sigma > 0$

- **Linear kernel** $k(\overline{x}, \overline{y}) = \Phi(\overline{x}) \bullet \Phi(\overline{y}) = \overline{x} \bullet \overline{y}$
  - feature space is simply the same as the input space
  - this kernel is useful for high-dimensional data sets in conjunction with soft-margin classifiers

- **Homogeneous polynomial kernel (of degree 2)**

$$k(\overline{x}, \overline{y}) = \Phi(\overline{x}) \bullet \Phi(\overline{y}) = (\overline{x} \bullet \overline{y})^2$$

$$\Phi(x_1, x_2) = (x_1^2, x_2^2, \sqrt{2}\, x_1 x_2)$$

- **Dual structure of the offset term b* can also be represented with a kernel**

$$\begin{aligned} b^* &= \overline{w}^* \bullet \Phi(\overline{x}_{sv^+}) - 1 \\ &= \sum_{i=1}^{l} \alpha_i^* y_i \Phi(\overline{x}_i) \bullet \Phi(\overline{x}_{sv^+}) - 1 \\ &= \sum_{i=1}^{l} \alpha_i^* y_i k(\overline{x}_i, \overline{x}_{sv^+}) - 1, \end{aligned}$$

- **To actually find the support vectors in feature space**
  - we must also apply the kernel trick to our training algorithm for support vector machine models:

$$\overline{\alpha}^* = \underset{\overline{\alpha}}{\operatorname{argmax}} \left(\sum_{i=1}^{l} \alpha_i - \frac{1}{2} \sum_{i=1}^{l} \sum_{j=1}^{l} \alpha_i \alpha_j y_i y_j k(\overline{x}_i, \overline{x}_j) \right)$$

  - subject to constraints $\sum_{i=1}^{l} \alpha_i y_i = 0,$ $\alpha_i \geq 0, \quad i = 1, \ldots, l$

- **To find our optimal Lagrangian multipliers $\overline{\alpha}^*$**
  - replace the expression $\kappa(\overline{x}_i, \overline{x}_j)$ with any appropriate kernel

- **Constraints** $\sum_{i=1}^{n} y_i \alpha_i = 0$ **and** $\alpha_j \geq 0$ for i = 1,…,/ **are not affected by the kernel trick**

Feature search

- We can rewrite our support vector machine model in terms of a kernel function

$$\hat{f}(\overline{x}) = \operatorname{sgn}\left(\sum_{i=1}^{l} \alpha_i^* y_i k(\overline{x}_i, \overline{x}) - b^*\right)$$

- Similarly, we can write our training algorithm in terms of a kernel function

$$\overline{\alpha}^* = \underset{\overline{\alpha}}{\operatorname{argmax}} \left(\sum_{j=1}^{l} \alpha_i - \frac{1}{2} \sum_{i=1}^{l} \sum_{j=1}^{l} \alpha_i \alpha_j y_i y_j k(\overline{x}_i, \overline{x}_j)\right)$$

- **We are free to change kernels according to the requirements of the classification problem**
  - if our classification problem involves quadratic decision surfaces in input space
    - choose a polynomial kernel of degree 2 $k(\overline{x}, \overline{y}) = (\overline{x} \bullet \overline{y})^2$
    - it maps quadratic decision surfaces in the input space to linear decision surfaces in the feature space
  - for more complex decision surfaces in the input space
    - try polynomial kernels of higher degrees or even more complex kernels (e.g., Gaussian kernel)

- **Feature search**
  - process of selecting a kernel and the associated values of its free parameters, such as the degree *d* for the polynomial kernel
  - in general, non-trivial and requiring some trade-offs in model complexity and model accuracy
  - e.g., grid search (provided by many packages)

Soft-margin classifiers

- **Generalization of maximum-margin classifiers**
  - to deal with noisy training data by allowing the classifiers to make mistakes
- **Recall**
  - maximum-margin classifiers are models of the form

$$\hat{f}(\overline{x}) = \operatorname{sgn}(\overline{w} \bullet \overline{x} - b)$$

  - where the normal vector $\overline{x}$ and the offset term *b* of the decision surface are computed via the primal optimization problem

$$\min \phi(\overline{w}, b) = \min \frac{1}{2} \overline{w} \bullet \overline{w}$$

  - subject to the constraints $y_i(\overline{w} \bullet \overline{x}_i - b) - 1 \geq 0$
  - given the training set $(\overline{x}_1, y_1), \ldots, (\overline{x}_l, y_l) \in \mathbb{R}^n \times \{+1, -1\}$

Construction of optimized maximum-margin classifier

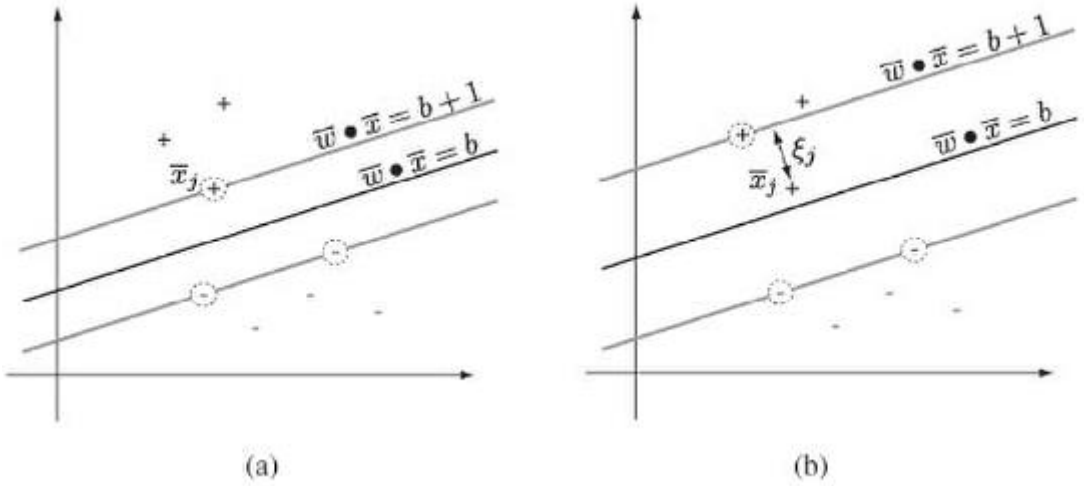
- positioning the supporting hyperplanes as far away from the decision surface as possible
  - so that they just touch their respective class boundaries
- only partially successful in the case of noisy training data where the size of the margin is limited by a few noisy training points
- reducing the impact that these points have on the size of the margin
  - by allowing them to lie on the “wrong” side of their respective supporting hyperplanes with the introduction of slack variables

Slack variable

- error terms that measure how far a particular point lies on the wrong side of its respective supporting hyperplane
- measuring how much of an error is committed by allowing the supporting hyperplane to be unconstrained by that point

Example

- (a) maximum-margin classifier with its margin limited by the (perhaps noisy) point ($\overline{x}_j$, +1)
- (b) the supporting hyperplane $\overline{\boldsymbol{w}} \cdot \overline{\boldsymbol{x}} = b + 1$ is unconstrained by the training point ($\overline{x}_j$, +1)
- the training point is allowed to lie on the wrong side of the supporting hyperplane
- the amount of the error is measured by the corresponding slack variable ξ_j



- **Constraint $\overline{w} \cdot \overline{x}_j - b \ - 1 \geq 0$ of optimization problem is violated**

- **We can recover a sensible constraint by taking the slack variable into account $\overline{w} \cdot \overline{x}_j - b + \xi_j - 1 \geq 0$**
  - it creates the illusion that the point appears to be located right on the supporting hyperplane
  - it satisfies the original constraint
  - $\xi_j \geq 0$: error is always measured as a positive quantity

- **For each training point $(\overline{x}_i, y_i)$**
  - constraints are $y_i(\overline{w} \bullet \overline{x}_i - b) + \xi_i - 1 \geq 0$, with $\xi_i \geq 0$

Proposition (Soft-Margin Optimization)

- given a training set

$$D = \{(\overline{x}_1, y_1), (\overline{x}_2, y_2), \ldots, (\overline{x}_l, y_l)\} \subseteq \mathbb{R}^n \times \{+1, -1\}$$

- we can compute a soft-margin decision surface $\overline{w}^* \bullet \overline{x} = b^*$

- with an optimization

$$\min_{\overline{w},\overline{\xi},b} \phi(\overline{w}, \overline{\xi}, b) = \min_{\overline{w},\overline{\xi},b} \left(\frac{1}{2} \overline{w} \bullet \overline{w} + C \sum_{i=1}^{l} \xi_i \right)$$

- subject to the constraints

$$y_i(\overline{w} \bullet \overline{x}_i - b) + \xi_i - 1 \geq 0$$

$$\xi_i \geq 0,$$

with $i = 1, \ldots, l$, $\overline{\xi} = (\xi_1, \ldots, \xi_l)$, *and* $C > 0$

Important fact

- still a convex optimization because all the error terms are constrained to be positive values

Optimizing the function $\phi(\overline{w}, \overline{\xi}, b)$

- it is a trade-off between the size of the margin and the size of the error
- the error is the sum of the values of the slack variables

- the larger we make the margin, the more training points will be on the wrong side of their respective supporting hyperplanes
  - therefore, the larger the error

- if we make the margin large, this will probably introduce a large number of nonzero slack variables
- If we make the margin small, we can reduce the number of nonzero slack variables
  - but, we are also back to where we started in the sense that noisy points will dictate the position of the decision surface

Controlling the trade-off between margin size and error

- constant C (called the cost) allows us to do
- we cannot simply ignore the slack variables by setting $C = 0$
- with a large value for C, the optimization will try to find a solution with as small a number of nonzero slack variables as possible
  - because errors are costly, due to the large C
- with a small value for C, the introduction of nonzero slack variables is much more forgiving and we can find solutions with a larger margin
  - ignoring some of the noisier points near the decision surface

large $C \sim$ small margin,
small $C \sim$ large margin.

- a solution $\overline{w}^*$, $\overline{\xi}^*$, and b* to the optimization problem

$$\min_{\overline{w},\overline{\xi},b} \phi(\overline{w},\overline{\xi},b) = \frac{1}{2}\overline{w}^* \bullet \overline{w}^* + C\sum_{i=1}^{l}\xi_i^* = m^*$$

is then a trade-off between the size of the margin m* and the size of the error $\sum_{i=1}^{l}\xi_i^*$.

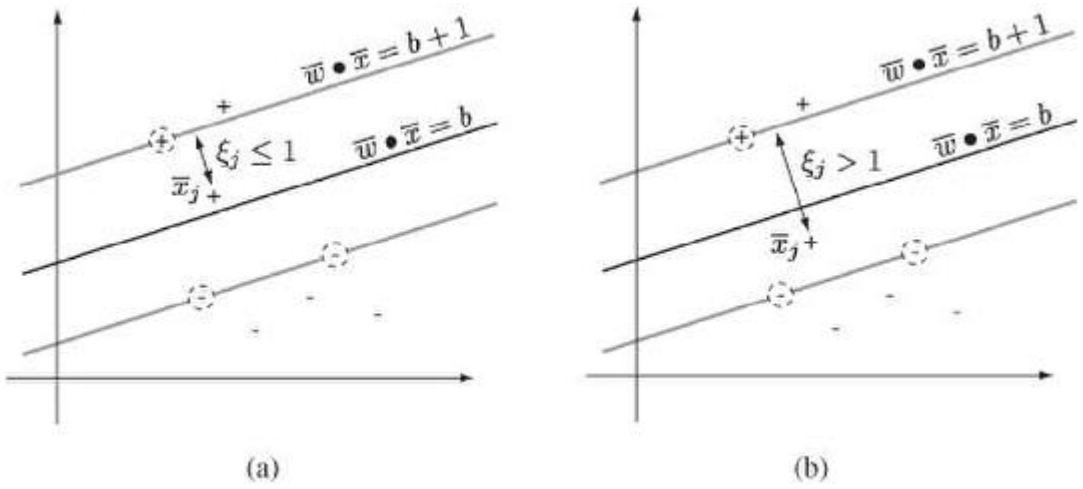
- **Slack variables only appear as part of the training algorithm**
  - maximum-margin classifier model itself remains unchanged

$$\hat{f}(\overline{x}) = \text{sgn}(\overline{w}^* \bullet \overline{x} - b^*)$$

  - only difference between the model in a hard-margin setting and in a soft-margin setting : allowing our model to make a certain number of classification errors governed by the cost constant C

- **If a particular point $(\overline{x}_j, y_j)$ has a slack variable with valueless than 1, $\xi_j \leq 1$**
  - the point will be classified correctly by the decision function even though the point lies in the margin

- **If the point has a slack variable such that $\xi_j > 1$**
  - the point will be misclassified by the decision function



We skip

- **Dual Setting for Soft-Margin Classifiers**

- **Implementation**
  - gradient ascent
  - quadratic programming
  - sequential minimal optimization (SMO)

InfoLab

Thank you!

DGIST 대구경북과학기술원
Daegu Gyeongbuk Institute of Science & Technology