Aplikace teorie neuronových sítí

Doc. RNDr. Iveta Mrázová, CSc.

Katedra teoretické informatiky
Matematicko-fyzikální fakulta
Univerzity Karlovy v Praze

Aplikace teorie neuronových sítí

– Podpůrné vektorové stroje –
– Support vector machines –

Doc. RNDr. Iveta Mrázová, CSc.

Katedra teoretické informatiky
Matematicko-fyzikální fakulta
Univerzity Karlovy v Praze

Optimální dělící nadrovina

- **Předp.:** Trénovací vzory $\left(\vec{x}_1, y_1\right), \ldots, \left(\vec{x}_l, y_l\right); \vec{x}_i \in \mathbf{R}^n; y_i \in \{+1, -1\}$
  lze separovat pomocí nadroviny $\left(\vec{w} \cdot \vec{x}\right) + b = 0$

- $\Rightarrow$ **optimální nadrovina**
  - Separace probíhá bez chyb
  - Vzdálenost vektorů nejbližších k nadrovině je maximální

$$\left(\vec{w} \cdot \vec{x}\right) + b \geq 1 \quad \text{jestliže} \quad y_i = 1$$
$$\left(\vec{w} \cdot \vec{x}\right) + b \leq -1 \quad \text{jestliže} \quad y_i = -1$$

  - $\Rightarrow \quad y_i \left[\left(\vec{w} \cdot \vec{x}\right) + b\right] \geq 1 \; ; \quad j = 1, \ldots, l$

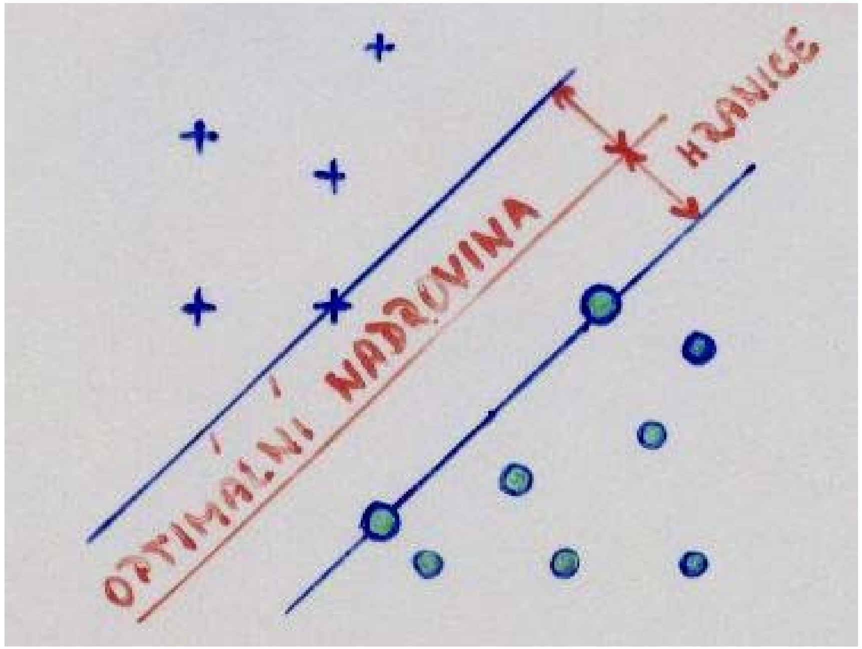
    $+$ minimalizace $\Phi\left(\vec{w}\right) = \left\|\vec{w}\right\|^2$ (zahrnuto $\vec{w}$ i b)

Optimální dělicí nadrovina (2)

- **Kanonická nadrovina**:

$$\min_{\vec{x}_i \in X^*} \left| \left(\vec{w} \cdot \vec{x}_i \right) + b \right| = 1 , \quad \text{kde} \;\; X^* = \left\{ \vec{x}_1 , \ldots , \vec{x}_l \right\}$$

- omezení
  - Parametry nadroviny
  - VC-dimenze



VC-dimenze

- **Věta: (V. N. Vapnik – 1995)**

  Podmnožina kanonických nadrovin $f\left(\vec{x}, \vec{w}, b\right) = \operatorname{sgn}\left\{\left(\vec{w} \cdot \vec{x}\right) + b\right\}$

  definovaná na množině vektorů

  $$X^* = \left\{ \vec{x}_1, \ldots, \vec{x}_l \mid \vec{x}_i \in \mathbf{R}^n;\ 1 \le i \le l \right\}$$

  omezených pomocí ***R***: $\left\| \vec{x}_i - \vec{a} \right\| \le R, \quad \vec{x}_i \in X^*$

  (***a*** odpovídá středu koule o poloměru ***R***)

  a splňujících podmínku $\left\| \vec{w} \right\| \le A$ má VC-dimenzi ***h*** omezenou

  pomocí $h \le \min\left(\left[R^2 A^2 \right], n \right) + 1$

Konstrukce optimální nadroviny

- Minimalizace: $\Phi\left(\vec{w}\right) = \frac{1}{2}\left(\vec{w} \cdot \vec{w}\right)$
- za podmínky: $y_i\left[\left(\vec{x}_i \cdot \vec{w}\right) + b\right] \geq 1; \quad i = 1, \dots, l$
- ⇢ sedlový bod funkcionálu

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

α_i … Lagrangeovy multiplikátory

Minimalizace vzhledem k $\vec{w}$ a $\boldsymbol{b}$

Maximalizace vzhledem k $\alpha_i > 0$

Konstrukce optimální nadroviny (2)

V sedlovém bodě by mělo řešení splňovat:

$$\frac{\partial L\left(\overrightarrow{w_0}, b_0, \overrightarrow{\alpha^0}\right)}{\partial b} = 0 \qquad \frac{\partial L\left(\overrightarrow{w_0}, b_0, \overrightarrow{\alpha^0}\right)}{\partial \vec{w}} = 0$$

⇒ **vlastnosti optimální nadroviny:**

1. Koeficienty optimální nadroviny by měly splňovat:

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

2. Optimální nadrovina (vektor $\overrightarrow{w_0}$) je lineární kombinací vektorů z trénovací množiny:

$$\overrightarrow{w_0} = \sum_{i=1}^{l} y_i \alpha_i^0 \overrightarrow{x_i} \; ; \qquad \alpha_i^0 \geq 0 \; ; \quad i = 1, \ldots , l$$

Konstrukce optimální nadroviny (3)

3. Pouze tzv. podpůrné vektory mohou mít nenulové koeficienty α_i^0 (v rozvoji $\overrightarrow{w_0}$): $\overrightarrow{w_0} = \sum_{\text{podpurné vektory}} y_i \alpha_i^0 \overrightarrow{x_i}\ ;\ \alpha_i^0 \geq 0$

$\Rightarrow$ Dělicí nadrovina splňuje podmínku:

$$\alpha_i^0 = \left\{ \left[\left(\overrightarrow{x_i} \cdot \overrightarrow{w_0} \right) + b_0 \right] y_i - 1 \right\} = 0\ ; \quad i = 1, \dots, l$$

$\Rightarrow$ **Maximalizace funkcionálu:**

$$W(\alpha) = \sum_{i=1}^{l} \alpha_i - \frac{1}{2} \sum_{i,j}^{l} \alpha_i \alpha_j y_i y_j \left(\overrightarrow{x_i} \cdot \overrightarrow{x_j} \right)$$

v nezáporném kvadrantu $\alpha_i \geq 0\ ; \quad i = 1, \dots, l$

za podmínky $\sum_{i=1}^{l} \alpha_i y_i = 0$

Klasifikace

- Nechť $\alpha^0 = \left(\alpha_1^0, \ldots, \alpha_l^0\right)$ je řešení $\rightarrow$ $W_0 = \sum_{\text{podpurné vektory}} y_i \, \alpha_i^0 \, \vec{x}_i$
- $\Rightarrow$ **klasifikace:**

$$f\left(\vec{x}\right) = \operatorname{sgn}\left(\sum_{\text{podpurné vektory}} y_i \, \alpha_i^0 \left(\vec{x}_i \cdot \vec{x}\right) - b_0 \right)$$

$\vec{x}_i \ldots$ podpůrný vektor $\alpha_{\mathrm{i}}^0 \ldots$ přísl. Lagrangeův koeficient

$b_0 \ldots$konstanta (práh) $b_0 = \frac{1}{2}\left[\left(\vec{w}_0 \cdot \vec{x}^*(1)\right) + \left(\vec{w}_0 \cdot \vec{x}^*(-1)\right)\right]$

$\vec{x}^*(1) \ldots$ podpůrný vektor patřící do 1. třídy

$\vec{x}^*(-1) \ldots$ podpůrný vektor patřící do 2. třídy

Příklad

- **Konstrukce dělicí nadroviny odpovídající polynomu 2. stupně:**

$\Rightarrow$ vytvořit příznakový prostor $\boldsymbol{Z}$, který bude mít $N = \dfrac{n\,(n+3)}{2}$ souřadnic

$z_1 = x_1, \ldots, z_n = x_n$ $\quad n$ souřadnic

$z_{n+1} = (x_1)^2, \ldots, z_{2n} = (x_n)^2$ $\quad n$ souřadnic

$z_{2n+1} = x_1 x_2, \ldots, z_N = x_n x_{n-1}$ $\quad \dfrac{n(n-1)}{2}$ souřadnic

$\vec{x} = (x_1, \ldots, x_n)$

Dělicí nadrovina konstruovaná v novém (příznakovém) prostoru (**Z**) odpovídá polynomu 2. stupně ve vstupním prostoru

Problémy

- Dělicí nadplocha nemusí nutně správně generalizovat
  - *Pokud lze optimální nadrovinu zkonstruovat na základě malého počtu podpůrných vektorů (vzhledem k trénovací množině), bude systém vykazovat vysoký stupeň generalizace*
- Vysoká dimenze příznakového prostoru
  - i v případě,   že optimální nadrovina generalizuje správně a lze ji teoreticky najít
  - *Konvoluce skalárního součinu*

Poznámky

- Při konstrukci optimální dělicí nadroviny v příznakovém prostoru Z není nutné uvažovat příznakový prostor v explicitní formě

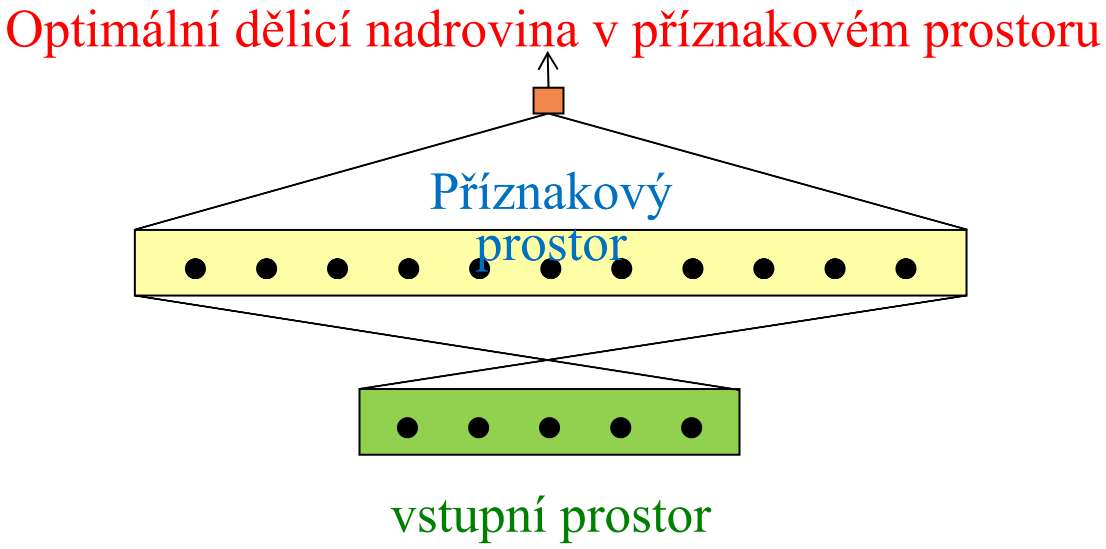
X je nutná možnost vyjádřit skalání součin pro podpůrné vektory v příznakovém prostoru

$$\left(\vec{z}_i \cdot \vec{z}\right) = K\left(\vec{x}, \vec{x}_i\right)$$

- $\vec{z}$ … obraz (v příznakovém prostoru) vektoru $\vec{x}$ (ze vstupního prostoru)

Podpůrné vektorové stroje (Suppor Vector (SV-) Machines)

- **Idea:**
  - Vstupní vzor (vektor $\vec{x}$) se zobrazí do vícerozměrného příznakového prostoru $\boldsymbol{Z}$ pomocí nelineárního, apriorně zvoleného zobrazení; V tomto prostoru pak probíhá konstrukce optimální dělicí nadroviny



Konstrukce SV-strojů

- Konvoluce skalárního součinu umožňuje konstrukci diskriminačních funkcí (nelineárních ve vstupním prostoru)

$$f\left(\vec{x}\right) = \operatorname{sgn}\left(\sum_{\text{podpurné vektory}} y_i\, \alpha_i\, K\left(\vec{x}_i, \vec{x}\right) - b_0 \right)$$

- (ekvivalentní lineárním diskriminačním funkcím v příznakovém prostoru; $K\left(\vec{x}_i, \vec{x}\right)$… konvoluce skalárního součinu)

Konstrukce SV-strojů (2)

- K nalezení koeficientů α_i (v separabilním případě) postačí nalézt maximum funkcionálu:

$$W(\vec{\alpha}) = \sum_{i=1}^{l} \alpha_i - \frac{1}{2} \sum_{i,j}^{l} \alpha_i \alpha_j y_i y_j K(\vec{x_i}, \vec{x_j})$$

za podmínky $\sum_{i=1}^{l} \alpha_i y_i = 0\,; \quad \alpha_i \geq 0\,;\ i = 1, \ldots, l$

(nelineární programovaní – např.: *J. J. More, G. Toraldo (1991): „On the solution of large quadratic programming problems with bound constrains,“ SIAM Optimization, 1, (1), pp. 93-113*) $\Rightarrow$ **učení**

Složitost konstrukce závisí na počtu podpůrných vektorů (spíš než na dimenzi příznakového prostoru)

Konstrukce SV-strojů (3)

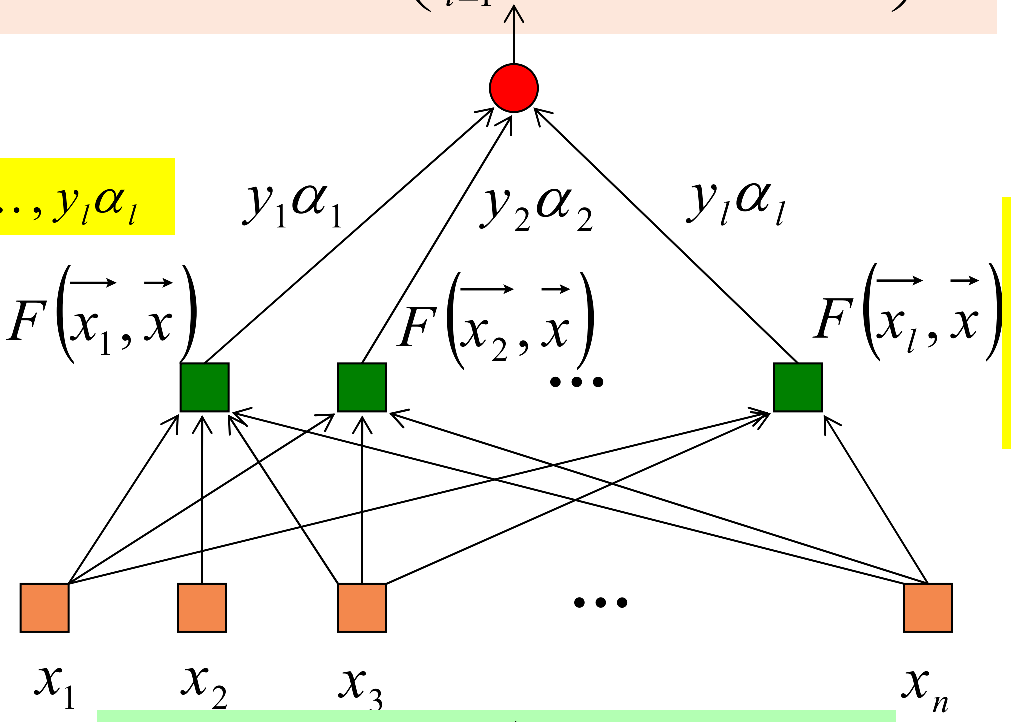
- Použitím různých funkcí pro konvoluci skalárního součinu lze zkonstruovat učící se stroje s různými typy nelineárních dělicích nadploch ve vstupním prostoru
  - Polynomiální učící se stroje
  - RBF-stroje
  - Dvouvrstvé neuronové sítě

Konstrukce SV-strojů (4)

rozhodovací pravidlo $$y = \operatorname{sgn}\left(\sum_{i=1}^{l} y_i \alpha_i F\left(\vec{x}_i, \vec{x}\right) - b\right)$$

váhy $y_1\alpha_1, \dots, y_l\alpha_l$

nelineární transf. (vych. z podp. vekt.) $\vec{x}_1, \dots, \vec{x}_l$



vstupní vektor $\vec{x} = (x_1, \dots, x_n)$

Dvouvrstvé neuronové sítě

- **Volba jádra:** $K\left(\vec{x}, \vec{x}_i\right) = S\,[\,v\left(\vec{x}, \vec{x}_i\right) + c\,]$
- ***S(u)*** … sigmoidální funkce $\tanh\left(vu + c\right), \quad |u| \leq 1$
- **Konstrukce SV-strojů:**

$$f\left(\vec{x}, \vec{\alpha}\right) = \operatorname{sgn}\left\{\sum_{i=1}^{l} \alpha_i S\left(v\left(\vec{x}, \vec{x}_i\right) + c\right) + b\right\}$$

Dvouvrstvé neuronové sítě

- **Automatické zjištění následujících údajů:**
  1. **Architektura dvouvrstvého stroje**
     - Počet ***l*** skrytých neuronů ~ počet podpůrných vektorů
  2. **Váhové vektory** $\vec{w}_i = \vec{x}_i$ neuronů první (skryté) vrstvy (~ podpůrné vektory)
  3. **Váhový vektor pro druhou vrstvu** (hodnoty $\vec{\alpha}$)

- **Příklad:** $K\left(\vec{x}, \vec{x}_i\right) = \tanh\left(\frac{b\left(\vec{x} \cdot \vec{x}_i\right)}{256} - c\right)$

$$b = 2$$
$$c = 1$$

Aplikace teorie neuronových sítí

– SVM- stroje –

– rozšiřující materiály –

Bing Liu: http://www.cs.uic.edu/~liub/WebMiningBook.html

Introduction to SVM

- Support vector machines were invented by V. Vapnik and his co-workers in 1970s in Russia and became known to the West in 1992.
- SVMs are linear classifiers that find a hyperplane to separate two class of data, positive and negative.
- Kernel functions are used for nonlinear separation.
- SVM not only has a rigorous theoretical foundation, but also performs classification more accurately than most other methods in applications, especially for high dimensional data.
- It is perhaps the best classifier for text classification.

Basic concepts

- Let the set of training examples D be

  $\{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \ldots, (\mathbf{x}_r, y_r)\}$,

  where $\mathbf{x}_i = (x_1, x_2, \ldots, x_n)$ is an **input vector** in a real-valued space $X \subseteq R^n$ and y_i is its **class label** (output value), $y_i \in \{1, -1\}$.

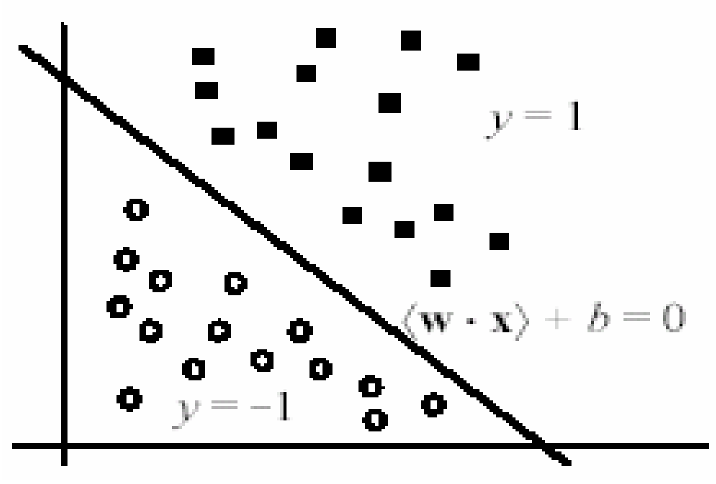
  1: positive class and -1: negative class.
- SVM finds a linear function of the form (**w**: weight vector)

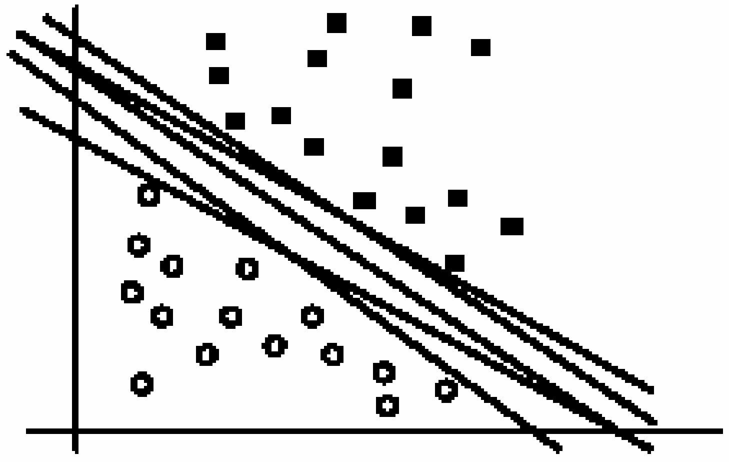
$$f(\mathbf{x}) = \langle \mathbf{w} \cdot \mathbf{x} \rangle + b$$

$$y_i = \begin{cases} 1 & if \langle \mathbf{w} \cdot \mathbf{x}_i \rangle + b \geq 0 \\ -1 & if \langle \mathbf{w} \cdot \mathbf{x}_i \rangle + b < 0 \end{cases}$$

The hyperplane

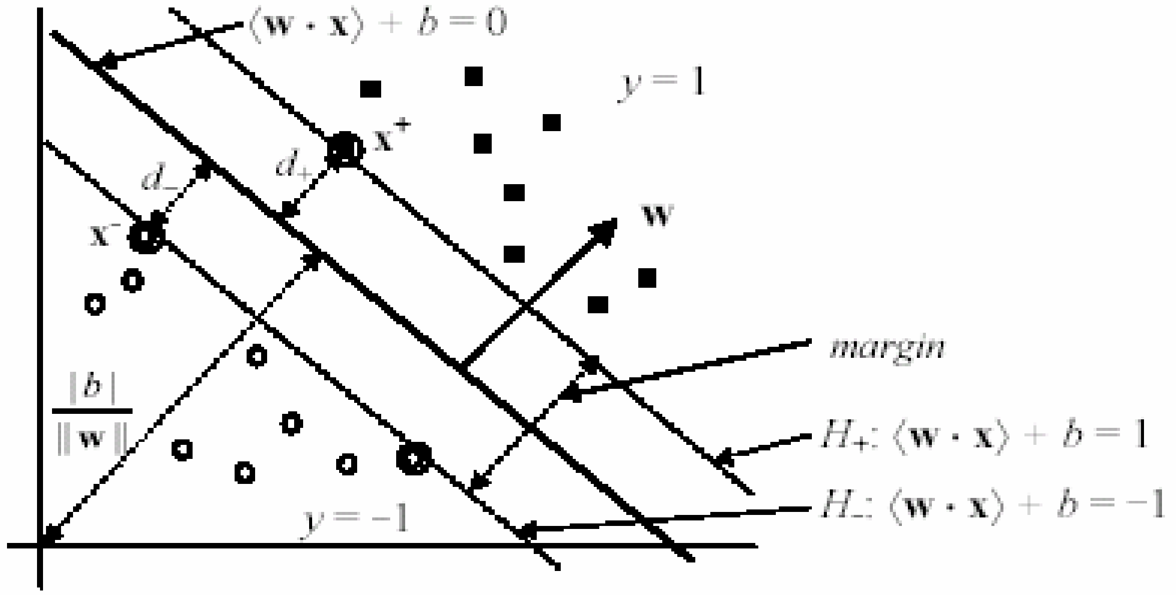
- The hyperplane that separates positive and negative training data is $\langle \mathbf{w} \cdot \mathbf{x} \rangle + b = 0$
- It is also called the decision boundary (surface).
- So many possible hyperplanes, which one to choose?





Maximal margin hyperplane

- SVM looks for the separating hyperplane with the largest margin.
- Machine learning theory says this hyperplane minimizes the error bound


$\langle \mathbf{w} \cdot \mathbf{x} \rangle + b = 0$
$y = 1$
$\mathbf{x}^+$
d_+
d_-
$\mathbf{x}^-$
$\mathbf{w}$
margin
$\frac{|b|}{\|\mathbf{w}\|}$
H_+: $\langle \mathbf{w} \cdot \mathbf{x} \rangle + b = 1$
H_-: $\langle \mathbf{w} \cdot \mathbf{x} \rangle + b = -1$
$y = -1$

Linear SVM: separable case

- Assume the data are linearly separable.
- Consider a positive data point ($\mathbf{x}^+$, 1) and a negative ($\mathbf{x}^-$, -1) that are closest to the hyperplane

  $$<\mathbf{w} \cdot \mathbf{x}> + b = 0.$$

- We define two parallel hyperplanes, H_+ and H_-, that pass through $\mathbf{x}^+$ and $\mathbf{x}^-$ respectively. H_+ and H_- are also parallel to $<\mathbf{w} \cdot \mathbf{x}> + b = 0.$

$$H_{+}: \quad \langle \mathbf{w} \cdot \mathbf{x}^+ \rangle + b = 1$$

$$H_{-}: \quad \langle \mathbf{w} \cdot \mathbf{x}^- \rangle + b = -1$$

such that

$$\begin{aligned} \langle \mathbf{w} \cdot \mathbf{x}_i \rangle + b &\geq 1 && \text{if } y_i = 1 \\ \langle \mathbf{w} \cdot \mathbf{x}_i \rangle + b &\leq -1 && \text{if } y_i = -1, \end{aligned}$$

Compute the margin

- Now let us compute the distance between the two margin hyperplanes H_+ and H_-. Their distance is the **margin** ($d_+ + d_-$ in the figure).
- Recall from vector space in algebra that the (perpendicular) **distance** from a point $\mathbf{x}_i$ to the hyperplane $\langle \mathbf{w} \cdot \mathbf{x} \rangle + b = 0$ is:

$$\frac{|\langle \mathbf{w} \cdot \mathbf{x}_i \rangle + b|}{\|\mathbf{w}\|} \qquad (36)$$

where $\|\mathbf{w}\|$ is the norm of $\mathbf{w}$,

$$\|\mathbf{w}\| = \sqrt{<\mathbf{w} \cdot \mathbf{w}>} = \sqrt{w_1^{\ 2} + w_2^{\ 2} + \ldots + w_n^{\ 2}} \qquad (37)$$

Compute the margin (cont …)

- Let us compute d_+.
- Instead of computing the distance from $\mathbf{x}^+$ to the separating hyperplane $\langle \mathbf{w} \cdot \mathbf{x} \rangle + b = 0$, we pick up any point $\mathbf{x}_s$ on $\langle \mathbf{w} \cdot \mathbf{x} \rangle + b = 0$ and compute the distance from $\mathbf{x}_s$ to $\langle \mathbf{w} \cdot \mathbf{x}^+ \rangle + b = 1$ by applying the distance Eq. (36) and noticing $\langle \mathbf{w} \cdot \mathbf{x}_s \rangle + b = 0$,

$$d_+ = \frac{|\langle \mathbf{w} \cdot \mathbf{x}_s \rangle + b - 1|}{\|\mathbf{w}\|} = \frac{1}{\|\mathbf{w}\|} \qquad (38)$$

$$margin = d_+ + d_- = \frac{2}{\|\mathbf{w}\|} \qquad (39)$$

A optimization problem!

Definition (Linear SVM: separable case): Given a set of linearly separable training examples,

$$D = \{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \ldots, (\mathbf{x}_r, y_r)\}$$

Learning is to solve the following constrained minimization problem,

$$\text{Minimize}: \frac{\langle \mathbf{w} \cdot \mathbf{w} \rangle}{2} \qquad (40)$$

$$\text{Subject to}: y_i(\langle \mathbf{w} \cdot \mathbf{x}_i \rangle + b) \geq 1, \quad i = 1, 2, \ldots, r$$

$y_i(\langle \mathbf{w} \cdot \mathbf{x}_i \rangle + b \geq 1, \; i = 1, 2, \ldots, r$ summarizes

$\langle \mathbf{w} \cdot \mathbf{x}_i \rangle + b \geq 1$ for $y_i = 1$

$\langle \mathbf{w} \cdot \mathbf{x}_i \rangle + b \leq -1$ for $y_i = -1$.

Solve the constrained minimization

- Standard Lagrangian method

$$L_P = \frac{1}{2}\langle \mathbf{w}\cdot\mathbf{w}\rangle - \sum_{i=1}^{r}\alpha_i[y_i(\langle \mathbf{w}\cdot\mathbf{x}_i\rangle + b) - 1] \qquad (41)$$

where $\alpha_i \geq 0$ are the **Lagrange multipliers**.

- Optimization theory says that an optimal solution to (41) must satisfy certain conditions, called **Kuhn-Tucker conditions**, which are necessary (but not sufficient)
- Kuhn-Tucker conditions play a central role in constrained optimization.

Kuhn-Tucker conditions

$$\frac{\partial L_P}{\partial w_j} = w_j - \sum_{i=1}^{r} y_i \alpha_i \mathbf{x}_i = 0, \quad j = 1, 2, \ldots, m \qquad (48)$$

$$\frac{\partial L_P}{\partial b} = -\sum_{i=1}^{r} y_i \alpha_i = 0 \qquad (49)$$

$$y_i(\langle \mathbf{w} \cdot \mathbf{x}_i \rangle + b) - 1 \geq 0, \quad i = 1, 2, \ldots, r \qquad (50)$$

$$\alpha_i \geq 0, \quad i = 1, 2, \ldots, r \qquad (51)$$

$$\alpha_i (y_i(\langle \mathbf{w} \cdot \mathbf{x}_i \rangle + b) - 1) = 0, \quad i = 1, 2, \ldots, r \qquad (52)$$

- Eq. (50) is the original set of constraints.
- The complementarity condition (52) shows that only those data points on the margin hyperplanes (i.e., H_+ and H_-) can have $\alpha_i > 0$ since for them $y_i(\langle \mathbf{w} \cdot \mathbf{x}_i \rangle + b) - 1 = 0$.
- These points are called the **support vectors**, All the other parameters $\alpha_i = 0$.

Solve the problem

- In general, Kuhn-Tucker conditions are necessary for an optimal solution, but not sufficient.
- However, for our minimization problem with a convex objective function and linear constraints, the Kuhn-Tucker conditions are both **necessary** and **sufficient** for an optimal solution.
- Solving the optimization problem is still a difficult task due to the inequality constraints.
- However, the Lagrangian treatment of the convex optimization problem leads to an alternative **dual** formulation of the problem, which is easier to solve than the original problem (called the **primal**).

Dual formulation

- From primal to a dual: Setting to zero the partial derivatives of the Lagrangian (41) with respect to the **primal variables** (i.e., $\mathbf{w}$ and b), and substituting the resulting relations back into the Lagrangian.
  - I.e., substitute (48) and (49), into the original Lagrangian (41) to eliminate the primal variables

$$L_D = \sum_{i=1}^{r} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{r} y_i y_j \alpha_i \alpha_j \langle \mathbf{x}_i \cdot \mathbf{x}_j \rangle, \qquad (55)$$

Dual optimization prolem

$$\text{Maximize: } L_D = \sum_{i=1}^{r} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{r} y_i y_j \alpha_i \alpha_j \langle \mathbf{x}_i \cdot \mathbf{x}_j \rangle. \tag{56}$$

$$\text{Subject to: } \sum_{i=1}^{r} y_i \alpha_i = 0$$

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

- This dual formulation is called the **Wolfe dual**.
- For the convex objective function and linear constraints of the primal, it has the property that the maximum of L_D occurs at the same values of **w**, *b* and α_i, as the minimum of L_P (the primal).
- Solving (56) requires numerical techniques and clever strategies, which are beyond our scope.

The final decision boundary

- After solving (56), we obtain the values for α_i, which are used to compute the weight vector **w** and the bias b using Equations (48) and (52) respectively.
- **The decision boundary**

$$\langle \mathbf{w} \cdot \mathbf{x} \rangle + b = \sum_{i \in sv} y_i \alpha_i \langle \mathbf{x}_i \cdot \mathbf{x} \rangle + b = 0 \qquad (57)$$

- **Testing**: Use (57). Given a test instance **z**,

$$sign(\langle \mathbf{w} \cdot \mathbf{z} \rangle + b) = sign\left(\sum_{i \in sv} \alpha_i y_i \langle \mathbf{x}_i \cdot \mathbf{z} \rangle + b \right) \qquad (58)$$

- If (58) returns 1, then the test instance **z** is classified as positive; otherwise, it is classified as negative.

Linear SVM: Non-separable case

- Linear separable case is the ideal situation.
- Real-life data may have noise or errors.
  - Class label incorrect or randomness in the application domain.
- Recall in the separable case, the problem was

$$\text{Minimize} \quad : \quad \frac{\langle \mathbf{w} \cdot \mathbf{w} \rangle}{2}$$

$$\text{Subject to} \quad : \quad y_i(\langle \mathbf{w} \cdot \mathbf{x}_i \rangle + b) \geq 1, \quad i = 1, 2, ..., r$$

- With noisy data, the constraints may not be satisfied. Then, no solution!

Relax the constraints

- To allow errors in data, we relax the margin constraints by introducing **slack** variables, ξ_i (≥ 0) as follows:

$$\langle \mathbf{w} \cdot \mathbf{x}_i \rangle + b \geq 1 - \xi_i \quad \text{for } y_i = 1$$
$$\langle \mathbf{w} \cdot \mathbf{x}_i \rangle + b \leq -1 + \xi_i \quad \text{for } y_i = -1.$$

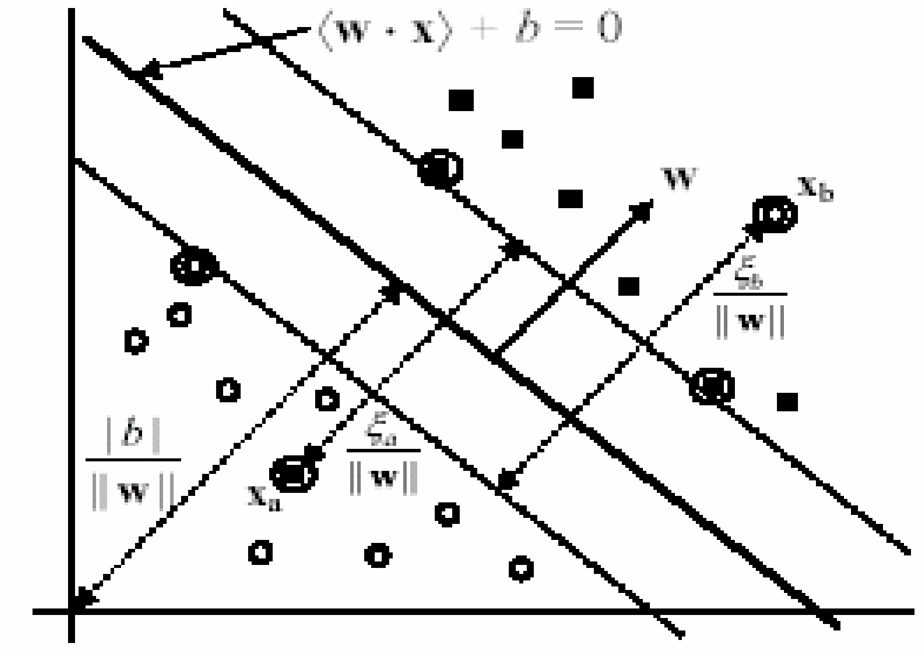
- The new constraints:

Subject to: $y_i(\langle \mathbf{w} \cdot \mathbf{x}_i \rangle + b) \geq 1 - \xi_i,\ i = 1, \ldots, r,$

$$\xi_i \geq 0,\ \ i = 1, 2, \ldots, r.$$

Geometric interpretation

- Two error data points $\mathbf{x}_a$ and $\mathbf{x}_b$ (circled) in wrong regions



$\langle \mathbf{w} \cdot \mathbf{x} \rangle + b = 0$
$\mathbf{w}$
$\mathbf{x}_b$
$\frac{\xi_b}{\|\mathbf{w}\|}$
$\frac{|b|}{\|\mathbf{w}\|}$
$\mathbf{x}_a$
$\frac{\xi_a}{\|\mathbf{w}\|}$

Penalize errors in objective function

- We need to penalize the errors in the objective function.
- A natural way of doing it is to assign an extra cost for errors to change the objective function to

$$\text{Minimize}: \frac{\langle \mathbf{w} \cdot \mathbf{w} \rangle}{2} + C\left(\sum_{i=1}^{r} \xi_i\right)^k \qquad (60)$$

- $k = 1$ is commonly used, which has the advantage that neither ξ_i nor its Lagrangian multipliers appear in the dual formulation.

New optimization problem

$$\text{Minimize} \quad : \quad \frac{\langle \mathbf{w} \cdot \mathbf{w} \rangle}{2} + C \sum_{i=1}^{r} \xi_i \qquad (61)$$

$$\text{Subject to} \quad : \quad y_i(\langle \mathbf{w} \cdot \mathbf{x}_i \rangle + b) \geq 1 - \xi_i, \quad i = 1, 2, \ldots, r$$

$$\xi_i \geq 0, \quad i = 1, 2, \ldots, r$$

- This formulation is called the **soft-margin SVM**. The primal Lagrangian is

$$L_P = \frac{1}{2}\langle \mathbf{w} \cdot \mathbf{w} \rangle + C\sum_{i=1}^{r} \xi_i - \sum_{i=1}^{r} \alpha_i [y_i(\langle \mathbf{w} \cdot \mathbf{x}_i \rangle + b) - 1 + \xi_i] - \sum_{i=1}^{r} \mu_i \xi_i \qquad (62)$$

where α_i, $\mu_i \geq 0$ are the **Lagrange multipliers**

Kuhn-Tucker conditions

$$\frac{\partial L_P}{\partial w_j} = w_j - \sum_{i=1}^{r} y_i \alpha_i \mathbf{x}_i = 0, \quad j = 1, 2, \ldots, m \tag{63}$$

$$\frac{\partial L_P}{\partial b} = -\sum_{i=1}^{r} y_i \alpha_i = 0 \tag{64}$$

$$\frac{\partial L_P}{\partial \xi_i} = C - \alpha_i - \mu_i = 0, \quad i = 1, 2, \ldots, r \tag{65}$$

$$y_i(\langle \mathbf{w} \cdot \mathbf{x}_i \rangle + b) - 1 + \xi_i \geq 0, \quad i = 1, 2, \ldots, r \tag{66}$$

$$\xi_i \geq 0, \quad i = 1, 2, \ldots, r \tag{67}$$

$$\alpha_i \geq 0, \quad i = 1, 2, \ldots, r \tag{68}$$

$$\mu_i \geq 0, \quad i = 1, 2, \ldots, r \tag{69}$$

$$\alpha_i(y_i(\langle \mathbf{w} \cdot \mathbf{x}_i \rangle + b) - 1 + \xi_i) = 0, \quad i = 1, 2, \ldots, r \tag{70}$$

$$\mu_i \xi_i = 0, \quad i = 1, 2, \ldots, r \tag{71}$$

From primal to dual

- As the linear separable case, we transform the primal to a dual by setting to zero the partial derivatives of the Lagrangian (62) with respect to the **primal variables** (i.e., **w**, b and ξ_i), and substituting the resulting relations back into the Lagrangian.
- Ie.., we substitute Equations (63), (64) and (65) into the primal Lagrangian (62).
- From Equation (65), $C - \alpha_i - \mu_i = 0$, we can deduce that $\alpha_i \leq C$ because $\mu_i \geq 0$.

Dual

- The dual of (61) is

$$\text{Maximize:}\quad L_D(\boldsymbol{\alpha}) = \sum_{i=1}^{r} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{r} y_i y_j \alpha_i \alpha_j \langle \mathbf{x}_i \cdot \mathbf{x}_j \rangle. \tag{72}$$

$$\text{Subject to:}\quad \sum_{i=1}^{r} y_i \alpha_i = 0$$

$$0 \le \alpha_i \le C, \quad i = 1, 2, \ldots, r.$$

- Interestingly, ξ_i and its Lagrange multipliers μ_i are not in the dual. The objective function is identical to that for the separable case.
- The only difference is the constraint $\alpha_i \le C$.

Find primal variable values

- The dual problem (72) can be solved numerically.
- The resulting α_i values are then used to compute **w** and b. **w** is computed using Equation (63) and b is computed using the Kuhn-Tucker complementarity conditions (70) and (71).
- Since no values for ξ_i, we need to get around it.
  - From Equations (65), (70) and (71), we observe that if $0 < \alpha_i < C$ then both $\xi_i = 0$ and $y_i\langle \mathbf{w} \cdot \mathbf{x}_i\rangle + b - 1 + \xi_i = 0$. Thus, we can use any training data point for which $0 < \alpha_i < C$ and Equation (69) (with $\xi_i = 0$) to compute b: $b = \frac{1}{y_i} - \sum_{i=1}^{r} y_i \alpha_i \langle \mathbf{x}_i \cdot \mathbf{x}_j \rangle = 0.$

(73)

(65), (70) and (71) in fact tell us more

$$
\begin{array}{lcl}
\alpha_i = 0 & \Rightarrow & y_i(\langle \mathbf{w} \cdot \mathbf{x}_i \rangle + b) \geq 1 \text{ and } \xi_i = 0 \\
0 < \alpha_i < C & \Rightarrow & y_i(\langle \mathbf{w} \cdot \mathbf{x}_i \rangle + b) = 1 \text{ and } \xi_i = 0 \\
\alpha_i = C & \Rightarrow & y_i(\langle \mathbf{w} \cdot \mathbf{x}_i \rangle + b) \leq 1 \text{ and } \xi_i \geq 0
\end{array} \qquad (74)
$$

- (74) shows a very important property of SVM.
  - The solution is **sparse** in α_i. Many training data points are outside the margin area and their α_i's in the solution are 0.
  - Only those data points that are on the margin (i.e., $y_i(\langle \mathbf{w} \cdot \mathbf{x}_i \rangle + b) = 1$, which are support vectors in the separable case), inside the margin (i.e., $\alpha_i = C$ and $y_i(\langle \mathbf{w} \cdot \mathbf{x}_i \rangle + b) < 1$), or errors are non-zero.
  - Without this sparsity property, SVM would not be practical for large data sets.

The final decision boundary

- The final decision boundary is (we note that many α_i's are 0)

$$\langle \mathbf{w} \cdot \mathbf{x} \rangle + b = \sum_{i=1}^{r} y_i \alpha_i \langle \mathbf{x}_i \cdot \mathbf{x} \rangle + b = 0 \qquad (75)$$

- The decision rule for classification (testing) is the same as the separable case, i.e.,

$$sign(\langle \mathbf{w} \cdot \mathbf{x} \rangle + b).$$

- Finally, we also need to determine the parameter C in the objective function. It is normally chosen through the use of a validation set or cross-validation.

How to deal with nonlinear separation?

- The SVM formulations require linear separation.
- Real-life data sets may need nonlinear separation.
- To deal with nonlinear separation, the same formulation and techniques as for the linear case are still used.
- We only transform the input data into another space (usually of a much higher dimension) so that
  - a linear decision boundary can separate positive and negative examples in the transformed space,
- The transformed space is called the **feature space**. The original data space is called the **input space**.

Space transformation

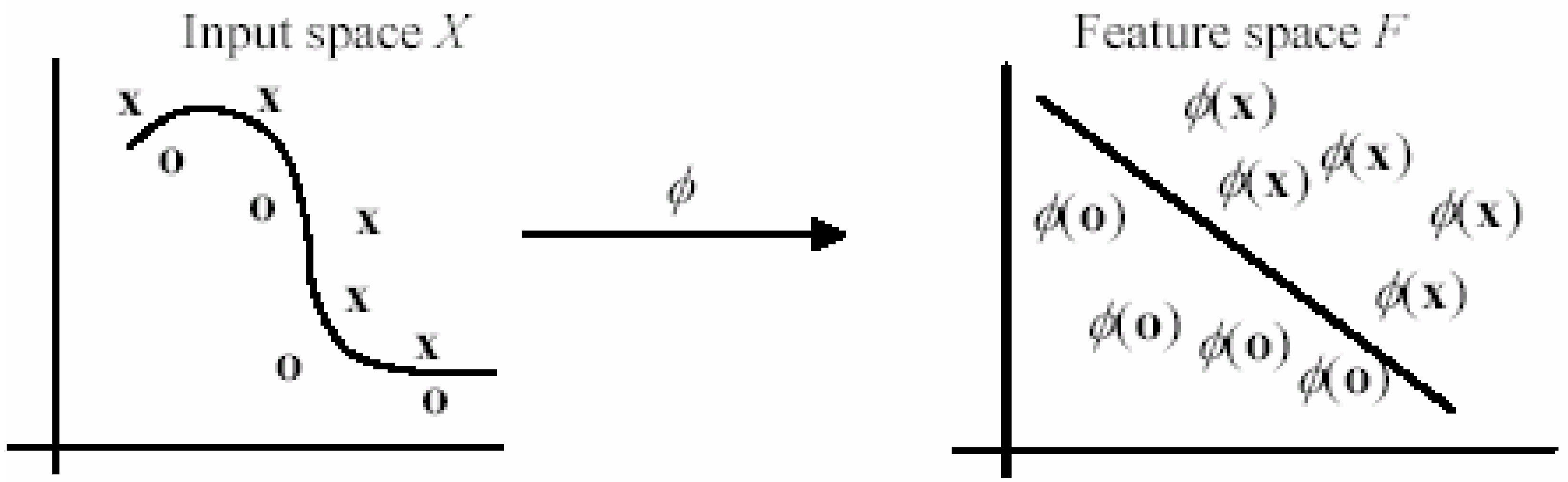
- The basic idea is to map the data in the input space X to a feature space F via a nonlinear mapping ϕ,

$$\begin{aligned} \phi &: X \to F \\ \mathbf{x} &\mapsto \phi(\mathbf{x}) \end{aligned} \qquad (76)$$

- After the mapping, the original training data set $\{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \ldots, (\mathbf{x}_r, y_r)\}$ becomes:

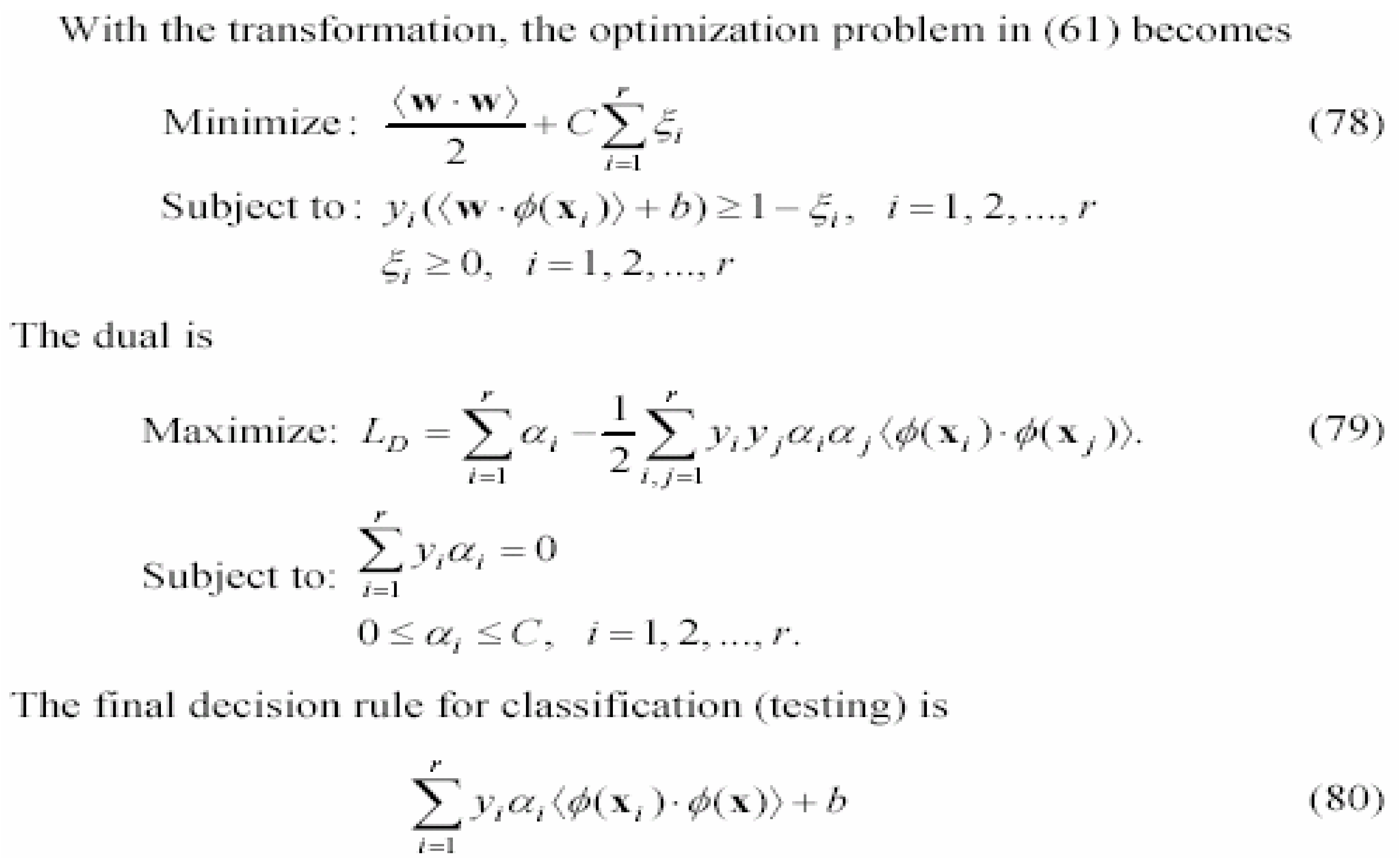
$$\{(\phi(\mathbf{x}_1), y_1), (\phi(\mathbf{x}_2), y_2), \ldots, (\phi(\mathbf{x}_r), y_r)\} \qquad (77)$$

Geometric interpretation



- In this example, the transformed space is also 2-D. But usually, the number of dimensions in the feature space is much higher than that in the input space

Optimization problem in (61) becomes



With the transformation, the optimization problem in (61) becomes

$$\text{Minimize:}\quad \frac{\langle \mathbf{w}\cdot\mathbf{w}\rangle}{2}+C\sum_{i=1}^{r}\xi_i \tag{78}$$

$$\text{Subject to:}\quad y_i(\langle \mathbf{w}\cdot\phi(\mathbf{x}_i)\rangle+b)\geq 1-\xi_i,\quad i=1,2,\dots,r$$

$$\xi_i\geq 0,\quad i=1,2,\dots,r$$

The dual is

$$\text{Maximize:}\quad L_D=\sum_{i=1}^{r}\alpha_i-\frac{1}{2}\sum_{i,j=1}^{r}y_iy_j\alpha_i\alpha_j\langle\phi(\mathbf{x}_i)\cdot\phi(\mathbf{x}_j)\rangle. \tag{79}$$

$$\text{Subject to:}\quad \sum_{i=1}^{r}y_i\alpha_i=0$$

$$0\leq\alpha_i\leq C,\quad i=1,2,\dots,r.$$

The final decision rule for classification (testing) is

$$\sum_{i=1}^{r}y_i\alpha_i\langle\phi(\mathbf{x}_i)\cdot\phi(\mathbf{x})\rangle+b \tag{80}$$

An example space transformation

- Suppose our input space is 2-dimensional, and we choose the following transformation (mapping) from 2-D to 3-D:

$$(x_1, x_2) \mapsto (x_1{}^2, x_2{}^2, \sqrt{2}x_1x_2)$$

- The training example ((2, 3), -1) in the input space is transformed to the following in the feature space:

  ((4, 9, 8.5), -1)

Problem with explicit transformation

- The potential problem with this explicit data trans-formation and then applying the linear SVM is that it may suffer from the curse of dimensionality.
- The number of dimensions in the feature space can be huge with some useful transformations even with reasonable numbers of attributes in the input space.
- This makes it computationally infeasible to handle.
- Fortunately, explicit transformation is not needed.

Kernel functions

- We notice that in the dual formulation both
  - the construction of the optimal hyperplane (79) in F and
  - the evaluation of the corresponding decision function (80)

  only require dot products $\langle \phi(\mathbf{x}) \cdot \phi(\mathbf{z}) \rangle$ and never the mapped vector $\phi(\mathbf{x})$ in its explicit form. This is a crucial point.
- Thus, if we have a way to compute the dot product $\langle \phi(\mathbf{x}) \cdot \phi(\mathbf{z}) \rangle$ using the input vectors $\mathbf{x}$ and $\mathbf{z}$ directly,
  - no need to know the feature vector $\phi(\mathbf{x})$ or even ϕ itself.
- In SVM, this is done through the use of **kernel functions**, denoted by K : $\quad K(\mathbf{x}, \mathbf{z}) = \langle \phi(\mathbf{x}) \cdot \phi(\mathbf{z}) \rangle \qquad (82)$

An example kernel function

- Polynomial kernel: $K(\mathbf{x}, \mathbf{z}) = \langle \mathbf{x} \cdot \mathbf{z} \rangle^d$ (83)
- Let us compute the kernel with degree $d = 2$ in a 2-dimensional space: $\mathbf{x} = (x_1, x_2)$ and $\mathbf{z} = (z_1, z_2)$.

$$\begin{aligned} \langle \mathbf{x} \cdot \mathbf{z} \rangle^2 &= (x_1 z_1 + x_2 z_2)^2 \qquad (84) \\ &= x_1^2 z_1^2 + 2 x_1 z_1 x_2 z_2 + x_2^2 z_2^2 \\ &= \langle (x_1^2, x_2^2, \sqrt{2} x_1 x_2) \cdot (z_1^2, z_2^2, \sqrt{2} z_1 z_2) \rangle \\ &= \langle \phi(\mathbf{x}) \cdot \phi(\mathbf{z}) \rangle, \end{aligned}$$

- This shows that the kernel $\langle \mathbf{x} \cdot \mathbf{z} \rangle^2$ is a dot product in a transformed feature space

Kernel trick

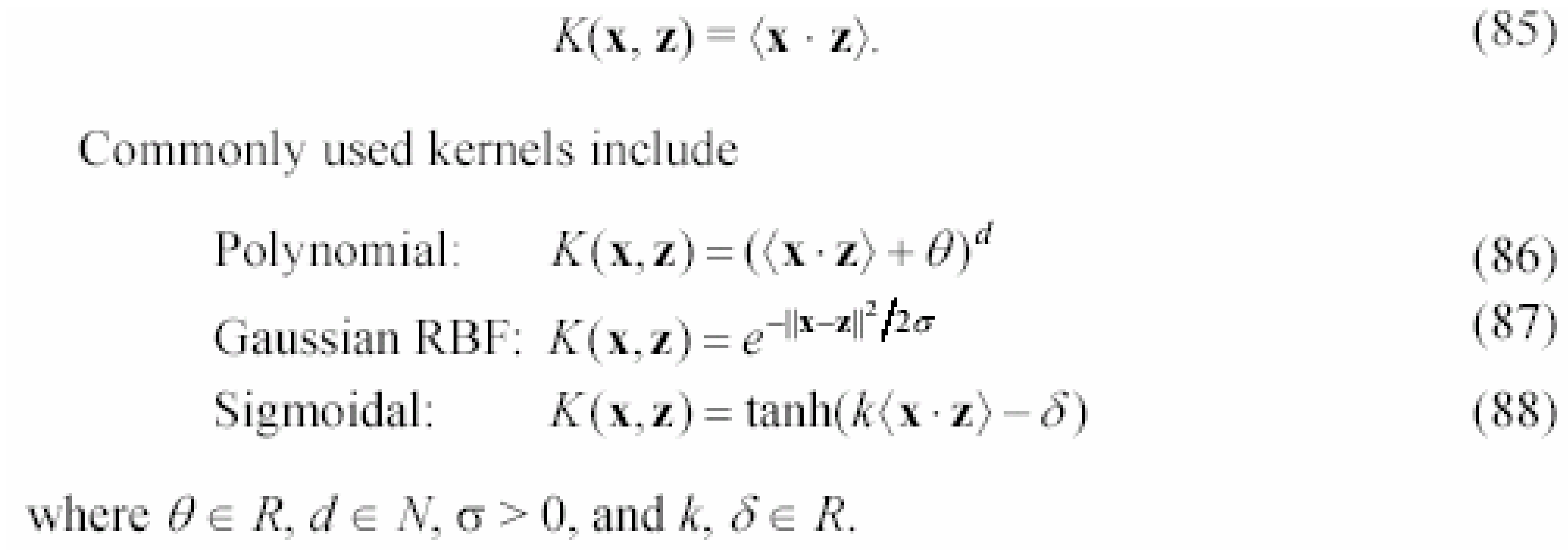
- The derivation in (84) is only for illustration purposes.
- We do not need to find the mapping function.
- We can apply the kernel function directly by
  - replace all the dot products $\langle \phi(\mathbf{x}) \cdot \phi(\mathbf{z}) \rangle$ in (79) and (80) with the kernel function $K(\mathbf{x}, \mathbf{z})$ (e.g., the polynomial kernel $\langle \mathbf{x} \cdot \mathbf{z} \rangle^d$ in (83)).
- This strategy is called the **kernel trick**.

Is it a kernel function?

- The question is: how do we know whether a function is a kernel without performing the derivation such as that in (84)? I.e,
  - How do we know that a kernel function is indeed a dot product in some feature space?
- This question is answered by a theorem called the Mercer's theorem, which we will not discuss here.

Commonly used kernels

- It is clear that the idea of kernel generalizes the dot product in the input space. This dot product is also a kernel with the feature map being the identity



$$K(\mathbf{x}, \mathbf{z}) = \langle \mathbf{x} \cdot \mathbf{z} \rangle. \qquad (85)$$

Commonly used kernels include

Polynomial: $K(\mathbf{x}, \mathbf{z}) = (\langle \mathbf{x} \cdot \mathbf{z} \rangle + \theta)^d$ (86)

Gaussian RBF: $K(\mathbf{x}, \mathbf{z}) = e^{-\|\mathbf{x}-\mathbf{z}\|^2 / 2\sigma}$ (87)

Sigmoidal: $K(\mathbf{x}, \mathbf{z}) = \tanh(k\langle \mathbf{x} \cdot \mathbf{z} \rangle - \delta)$ (88)

where $\theta \in R$, $d \in N$, $\sigma > 0$, and k, $\delta \in R$.

Some other issues in SVM

- SVM works only in a real-valued space. For a categorical attribute, we need to convert its categorical values to numeric values.
- SVM does only two-class classification. For multi-class problems, some strategies can be applied, e.g., one-against-rest, and error-correcting output coding.
- The hyperplane produced by SVM is hard to understand by human users. The matter is made worse by kernels. Thus, SVM is commonly used in applications that do not required human understanding.