Optimální rozdělující nadplocha.
Support vector machine. Adaboost.

Petr Pošík

Czech Technical University in Prague
Faculty of Electrical Engineering
Dept. of Cybernetics

Opakování

Binární klasifikace objektů $\mathbf{x}$ (klasifikace do 2 tříd, dichotomie):

- ✔ Stačí 1 diskriminační funkce
- ✔ Pravidlo:

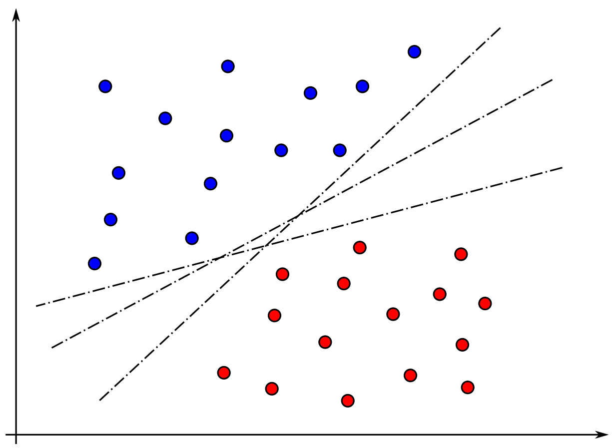
$$f(\mathbf{x}_i) > 0 \iff y_i = +1$$
$$f(\mathbf{x}_i) < 0 \iff y_i = -1$$

neboli

$$y_i = \operatorname{sign}(f(\mathbf{x}))$$

Učení lin. diskr. funkce: např. perceptronový algoritmus:

- ✔ váhový vektor je váženým součtem trénovacích bodů $\mathbf{x}_i$
- ✔ najde jakoukoli separující nadplochu, pokud taková existuje



Která z nekonečného množství rozdělujících nadploch je optimální?

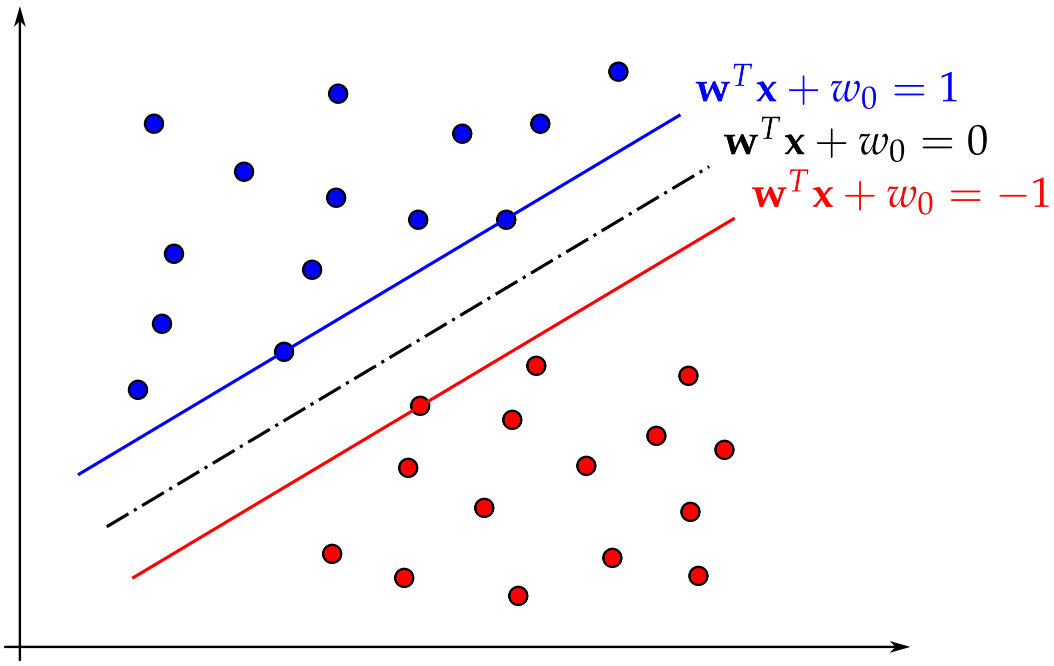
Optimální rozdělující nadplocha

Optimální rozdělující nadplocha

Margin (odstup):

- ✔ šířka pásma, v němž se hranice mezi třídami může pohybovat (kolmo ke své směrnici), aniž by se dotkla některého bodu

Lineární klasifikátor s maximálním marginem



Plus-rovina: $\{x : \mathbf{w}^T\mathbf{x} + w_0 = 1\}$
Minus-rovina: $\{x : \mathbf{w}^T\mathbf{x} + w_0 = -1\}$
Hranice mezi třídami: $\{x : \mathbf{w}^T\mathbf{x} + w_0 = 0\}$

Support vectors:

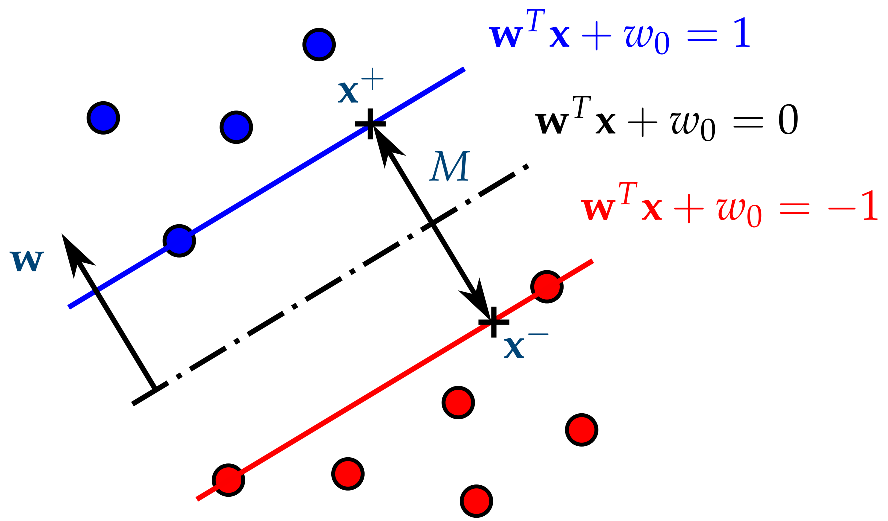
- ✔ datové vektory ležící v plus-rovině nebo minus-rovině
- ✔ pouze ty mají vliv na hranici mezi třídami

Proč maximalizovat margin?

- ✔ Intuitivně se to zdá být nejbezpečnější
- ✔ Pokud bychom udělali malou chybu při stanovování hranice, máme nejmenší šanci, že způsobíme špatnou klasifikaci
- ✔ Model je stabilní vůči změně trénovací množiny, pokud nedojde ke změně některého ze support vectors
- ✔ Existují teoretické výsledky (založené na VC dimenzi), že hledat klasifikátor s maximálním marginem je dobré
- ✔ Maximální margin v praxi dobře funguje

Velikost marginu

Jak vypočítat velikost marginu M ze známých $\mathbf{w}$ a w_0?



Víme, že:

$$\mathbf{w}^T\mathbf{x}^+ + w_0 = 1$$
$$\mathbf{w}^T\mathbf{x}^- + w_0 = -1$$
$$\mathbf{x}^- + \lambda\mathbf{w} = \mathbf{x}^+$$

Můžeme odvodit:

$$\mathbf{w}^T(\mathbf{x}^+ - \mathbf{x}^-) = 2$$
$$\mathbf{w}^T(\mathbf{x}^- + \lambda\mathbf{w} - \mathbf{x}^-) = 2$$
$$\lambda\mathbf{w}^T\mathbf{w} = 2$$
$$\lambda = \frac{2}{\mathbf{w}^T\mathbf{w}} = \frac{2}{\|\mathbf{w}\|^2}$$

Velikost marginu:

$$M = \|\mathbf{x}^+ - \mathbf{x}^-\| = \|\lambda\mathbf{w}\| = \lambda\|\mathbf{w}\| = \frac{2}{\|\mathbf{w}\|^2}\|\mathbf{w}| = \frac{2}{\|\mathbf{w}\|}$$

Učení optimální rozdělující nadplochy

Chceme maximalizovat margin $M = \frac{2}{\|\mathbf{w}\|}$ při splnění omezení na správnou klasifikaci.
To lze formulovat jako úlohu *kvadratického programování* (QP).

Učení optimální rozdělující nadplochy

Chceme maximalizovat margin $M = \frac{2}{\|\mathbf{w}\|}$ při splnění omezení na správnou klasifikaci. To lze formulovat jako úlohu *kvadratického programování* (QP).

- ✔ Primární úloha QP:

$$\begin{aligned} &\text{minimalizuj } \mathbf{w}^T\mathbf{w} \text{ vzhledem k } w_1, \dots, w_D \\ &\quad\text{tak, aby } y_i(\mathbf{w}^T\mathbf{x}_i + w_0) \geq 1 \end{aligned}$$

Učení optimální rozdělující nadplochy

Chceme maximalizovat margin $M = \frac{2}{\|\mathbf{w}\|}$ při splnění omezení na správnou klasifikaci. To lze formulovat jako úlohu *kvadratického programování* (QP).

- ✔ Primární úloha QP:

$$\text{minimalizuj } \mathbf{w}^T\mathbf{w} \text{ vzhledem k } w_1, \ldots, w_D$$
$$\text{tak, aby } y_i(\mathbf{w}^T\mathbf{x}_i + w_0) \geq 1$$

- ✔ Duální úloha QP:

$$\text{maximalizuj } \sum_{i=1}^{N} \alpha_i - \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N} \alpha_i\alpha_j y_i y_j \mathbf{x}_i^T\mathbf{x}_j \text{ vzhledem k } \alpha_1, \ldots, \alpha_m$$
$$\text{tak, aby } \alpha_i \geq 0$$
$$\text{a } \sum_{i=1}^{N} \alpha_i y_i = 0.$$

Učení optimální rozdělující nadplochy

Chceme maximalizovat margin $M = \frac{2}{\|\mathbf{w}\|}$ při splnění omezení na správnou klasifikaci. To lze formulovat jako úlohu *kvadratického programování* (QP).

- ✔ Primární úloha QP:

$$\begin{aligned} &\text{minimalizuj } \mathbf{w}^T\mathbf{w} \text{ vzhledem k } w_1, \dots, w_D \\ &\quad \text{tak, aby } y_i(\mathbf{w}^T\mathbf{x}_i + w_0) \geq 1 \end{aligned}$$

- ✔ Duální úloha QP:

$$\begin{aligned} &\text{maximalizuj } \sum_{i=1}^{N} \alpha_i - \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N} \alpha_i\alpha_j y_i y_j \mathbf{x}_i^T\mathbf{x}_j \text{ vzhledem k } \alpha_1, \dots, \alpha_m \\ &\quad \text{tak, aby } \alpha_i \geq 0 \\ &\qquad \text{a } \sum_{i=1}^{N} \alpha_i y_i = 0. \end{aligned}$$

- ✔ Výpočet řešení primární úlohy z řešení duální úlohy:

$$\mathbf{w} = \sum_{i=1}^{m} \alpha_i y_i \mathbf{x}_i, \qquad w_0 = y_k - \mathbf{w}^T\mathbf{x}_k,$$

kde $(\mathbf{x}_k, y_k)$ je libovolný *support vector*.

Optimální rozdělující nadplocha: závěrečné poznámky

Význam duální formulace:

- ✔ Úloha QP v duální formě je pro QP solvery snáze řešitelná než QP v primární formě.
- ✔ Klasifikovat nové body pak lze pomocí funkce

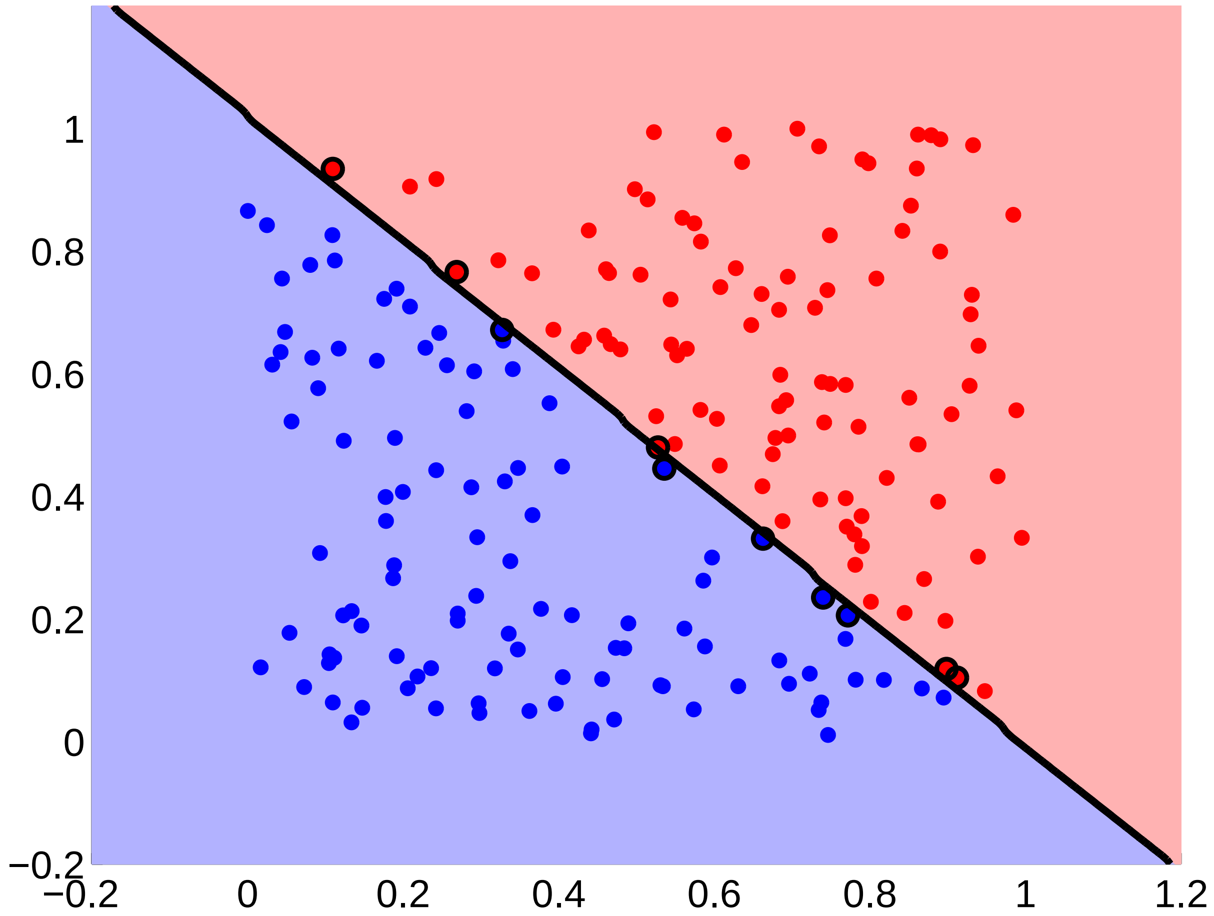
$$f(\mathbf{x}, \mathbf{w}, w_0) = \operatorname{sign}(\mathbf{w}^T\mathbf{x} + w_0) = \operatorname{sign}(\sum_{i=1}^{N} \alpha_i y_i \mathbf{x}_i^T \mathbf{x} + w_0),$$

  t.j. data se v diskriminační funkci vyskytují jen ve formě skalárních součinů.
- ✔ Data s $\alpha_i > 0$ jsou support vectors, takže součty mohou být jen přes *support vectors*.
- ✔ Duální formulace umožňuje využít dalších triků, o kterých se teprve dozvíte.

Když data nejsou lineárně separabilní:

- ✔ I v tomto případě existuje formulace úlohy QP (*soft margin*).
- ✔ Primární úloha má dvojnásobný počet omezení, úloha tedy je složitější.
- ✔ Výsledky úlohy QP se *soft marginem* jsou stejného typu jako bez něj.

Demo: optimální rozdělující nadplocha



Když lineární hranice nestačí. . .

Rozšíření báze

a.k.a. *vyrovnání příznakového porstoru*.

a.k.a. *vyrovnání příznakového porstoru*.

Proč?

- ✔ Lineární hranice nemusí být dostatečně flexibilní
- ✔ Využijeme algoritmy pro učení lineární diskriminační funkce i pro učení nelineárních modelů

Rozšíření báze

a.k.a. *vyrovnání příznakového porstoru*.

Proč?

- ✔ Lineární hranice nemusí být dostatečně flexibilní
- ✔ Využijeme algoritmy pro učení lineární diskriminační funkce i pro učení nelineárních modelů

Jak?

- ✔ Zavedeme nový vícerozměrný obrazový prostor F
- ✔ Do tohoto prostoru přetransformujeme data (odvodíme nové příznaky):

$$\begin{aligned} \mathbf{x} \quad &\rightarrow \quad \mathbf{z} = \Phi(\mathbf{x}), \\ \mathbf{x} = (x_1, x_2, \ldots, x_D) \quad &\rightarrow \quad \mathbf{z} = (\Phi_1(\mathbf{x}), \Phi_2(\mathbf{x}), \ldots, \Phi_G(\mathbf{x})), \end{aligned}$$

  přičemž obvykle $D \ll G$.
- ✔ Naučíme lineární model, který v původním prostoru nebude lineární

$$\begin{aligned} f(\mathbf{z}) &= w_1 z_1 + w_2 z_2 + \ldots + w_G z_G + w_0 \\ f(\mathbf{x}) &= w_1 \Phi_1(\mathbf{x}) + w_2 \Phi_2(\mathbf{x}) + \ldots + w_G \Phi_G(\mathbf{x}) + w_0 \end{aligned}$$

Dvě soustavy souřadnic

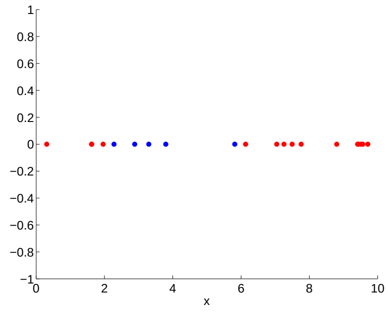
Transformace do vícerozměrného prostoru obrazů

Původní prostor	Prostor obrazů
$\mathbf{x} = (x_1, x_2, \ldots, x_D)$	$\mathbf{z} = (z_1, z_2, \ldots, z_G)$ $z_1 = \log x_1$ $z_2 = x_1^2 x_3$ $z_3 = e^{x_2}$ $\ldots$
$f(\mathbf{x}) = w_1 \log x_1 + w_2 x_1^2 x_3 + w_3 e^{x_2} + \ldots + w_0$	$f(\mathbf{z}) = w_1 z_1 + w_2 z_2 + w_3 z_3 + \ldots + w_0$

Učení lineární funkce v prostoru obrazů

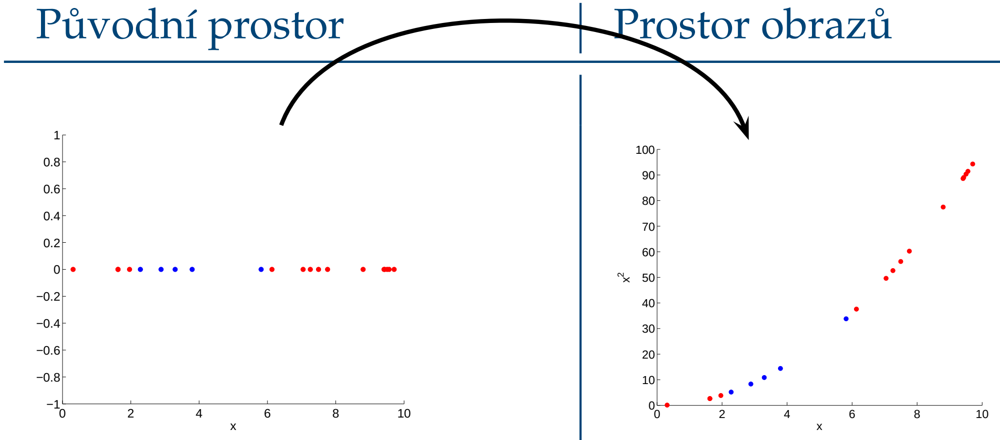
Nelineární funkce v původním prostoru

Dvě soustavy souřadnic: graficky

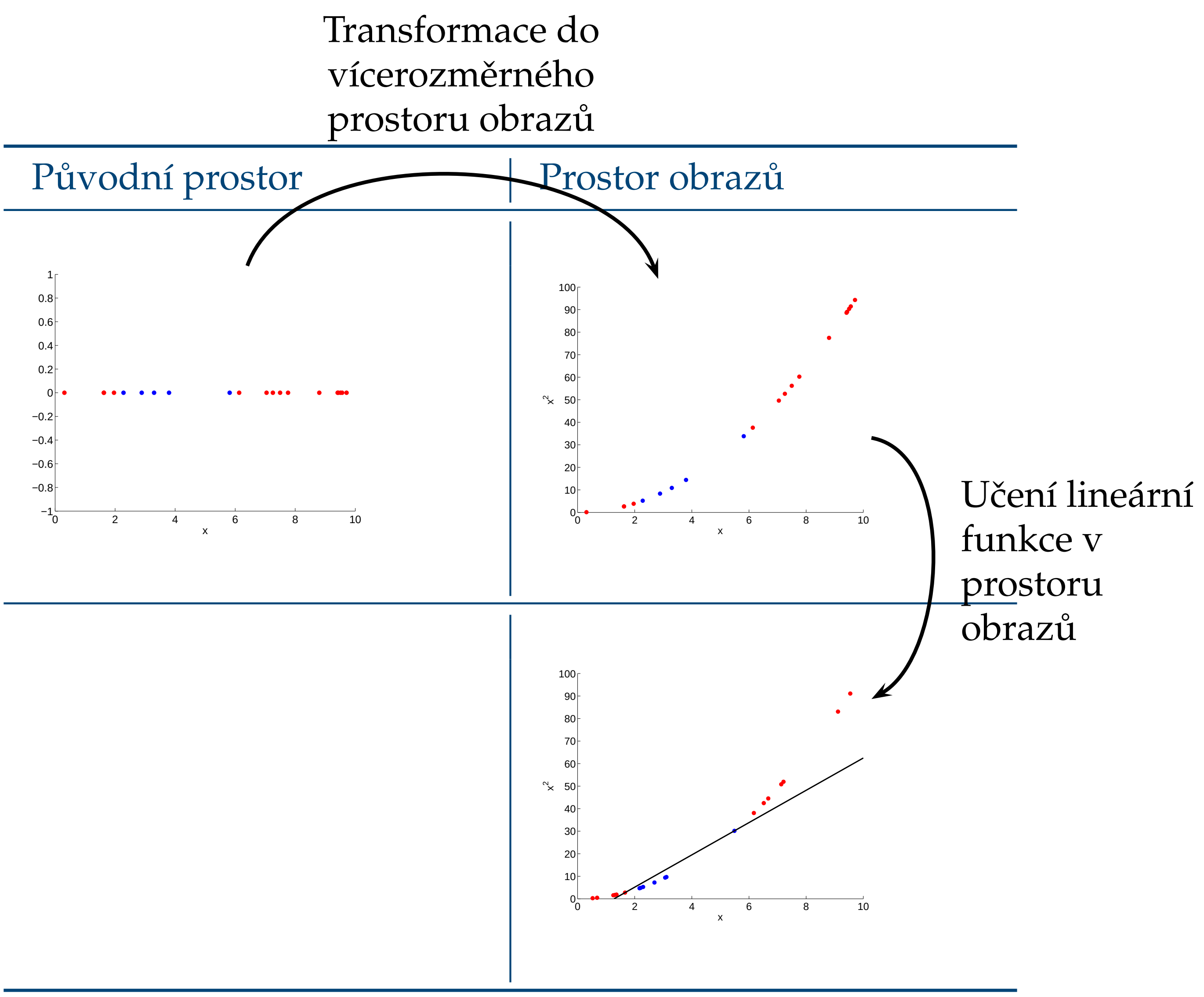
Původní prostor	Prostor obrazů
	

Transformace do vícerozměrného prostoru obrazů

Původní prostor	Prostor obrazů

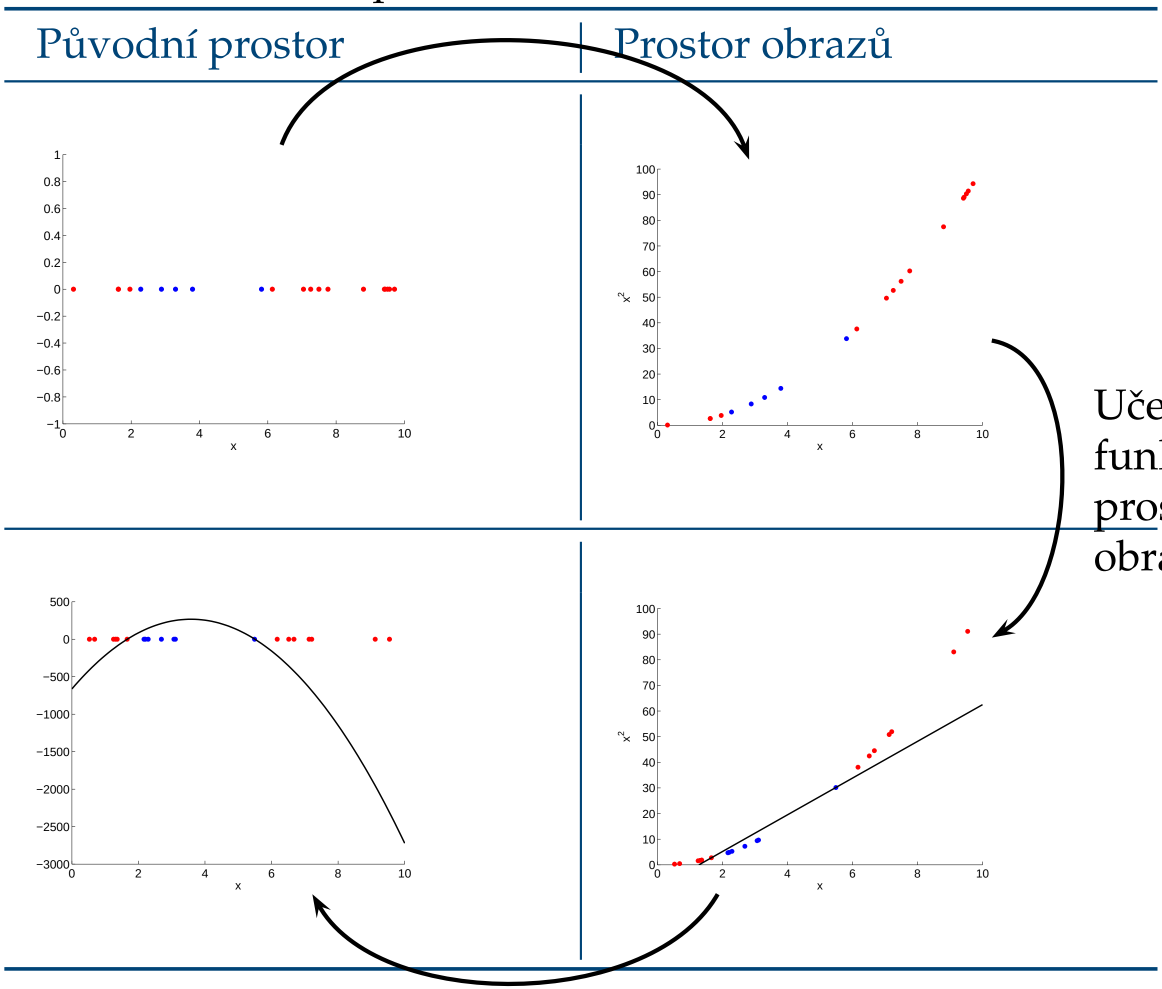


Dvě soustavy souřadnic: graficky



Dvě soustavy souřadnic: graficky

Transformace do vícerozměrného prostoru obrazů



Učení lineární funkce v prostoru obrazů

Nelineární funkce v původním prostoru

Rozšíření báze: poznámky

Výhody:

- ✔ univerzální, obecně použitelná metoda

Nevýhody:

- ✔ zůstává na člověku zvolit, které nové příznaky dá učicímu algoritmu k dispozici
- ✔ data je třeba skutečně namapovat do obrazového prostoru

Existuje metoda, jak využít rozšíření báze, aniž bychom mapování skutečně museli provést!!!

Optimální rozdělujíci nadplocha s rozšířením báze

Připomeňme: při učení optimální rozdělující nadplochy se vektory $\mathbf{x}$ vyskytují pouze

v optimalizačním kritériu $\sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_i \alpha_j y_i y_j \mathbf{x}_i^T \mathbf{x}_j$

a v diskriminační funkci $f(\mathbf{x}) = \text{sign}(\sum_{i=1}^{N} \alpha_i y_i \mathbf{x}_i^T \mathbf{x} + w_0)$.

Optimální rozdělujíci nadplocha s rozšířením báze

Připomeňme: při učení optimální rozdělující nadplochy se vektory $\mathbf{x}$ vyskytují pouze

v optimalizačním kritériu $\sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_i \alpha_j y_i y_j \mathbf{x}_i^T \mathbf{x}_j$

a v diskriminační funkci $f(\mathbf{x}) = \text{sign}(\sum_{i=1}^{N} \alpha_i y_i \mathbf{x}_i^T \mathbf{x} + w_0)$.

Použijeme-li rozšíření báze, změní se

optimalizační kritérium na $\sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_i \alpha_j y_i y_j \Phi(\mathbf{x}_i)^T \Phi(\mathbf{x}_j)$

a diskriminační funkce na $f(\mathbf{x}) = \text{sign}(\sum_{i=1}^{N} \alpha_i y_i \Phi(\mathbf{x}_i)^T \Phi(\mathbf{x}) + w_0)$.

Optimální rozdělujíci nadplocha s rozšířením báze

Připomeňme: při učení optimální rozdělující nadplochy se vektory $\mathbf{x}$ vyskytují pouze

$$\text{v optimalizačním kritériu } \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_i \alpha_j y_i y_j \mathbf{x}_i^T \mathbf{x}_j$$

$$\text{a v diskriminační funkci } f(\mathbf{x}) = \text{sign}(\sum_{i=1}^{N} \alpha_i y_i \mathbf{x}_i^T \mathbf{x} + w_0).$$

Použijeme-li rozšíření báze, změní se

$$\text{optimalizační kritérium na } \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_i \alpha_j y_i y_j \Phi(\mathbf{x}_i)^T \Phi(\mathbf{x}_j)$$

$$\text{a diskriminační funkce na } f(\mathbf{x}) = \text{sign}(\sum_{i=1}^{N} \alpha_i y_i \Phi(\mathbf{x}_i)^T \Phi(\mathbf{x}) + w_0).$$

Místo skalárního součinu v obrazovém prostoru lze použít funkci $K(\mathbf{x}_i, \mathbf{x}_j)$

$$\text{v optimalizačním kritériu } \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_i \alpha_j y_i y_j K(\mathbf{x}_i, \mathbf{x}_j)$$

$$\text{a v diskriminační funkci } f(\mathbf{x}) = \text{sign}(\sum_{i=1}^{N} \alpha_i y_i K(\mathbf{x}_i, \mathbf{x}) + w_0).$$

Kernel trik

Existují funkce dvou vektorových argumentů ($K(\mathbf{a}, \mathbf{b})$), jejichž hodnoty jsou rovny skalárnímu součinu ($\phi(\mathbf{a})^T\phi(\mathbf{b})$) vektorů vzniklých namapováním obou původních vektorů do mnoharozměrného prostoru obrazů. Takovým funkcím se říká **jádrové funkce** *nebo* **kernely**.

Kernel trik: Mějme lineární algoritmus, v němž se trénovací příklady vyskytují pouze ve skalárních součinech.

- ✔ Takový algoritmus lze znelinearizovat, když nahradíme skalární součiny dvou trénovacích vektorů jádrovou funkcí těchto vektorů.
- ✔ Výsledek je stejný, jako by se tentýž algoritmus učil v mnoharozměrném prostoru nelineárních bázových funkcí.
- ✔ Díky kernelům ovšem není nutné vlastní mapování provádět, algoritmus je mnohem efektivnější.

Často používané kernely:

Polynomiální: $K(\mathbf{a}, \mathbf{b}) = (\mathbf{a}^T\mathbf{b} + 1)^d$, kde d je stupeň polynomu.

Gaussovský (RBF): $K(\mathbf{a}, \mathbf{b}) = \exp(-\frac{|\mathbf{a} - \mathbf{b}|^2}{\sigma^2})$, kde σ^2 je „šířka“ Gaussiánu.

Support vector machine

Support vector machine (SVM)

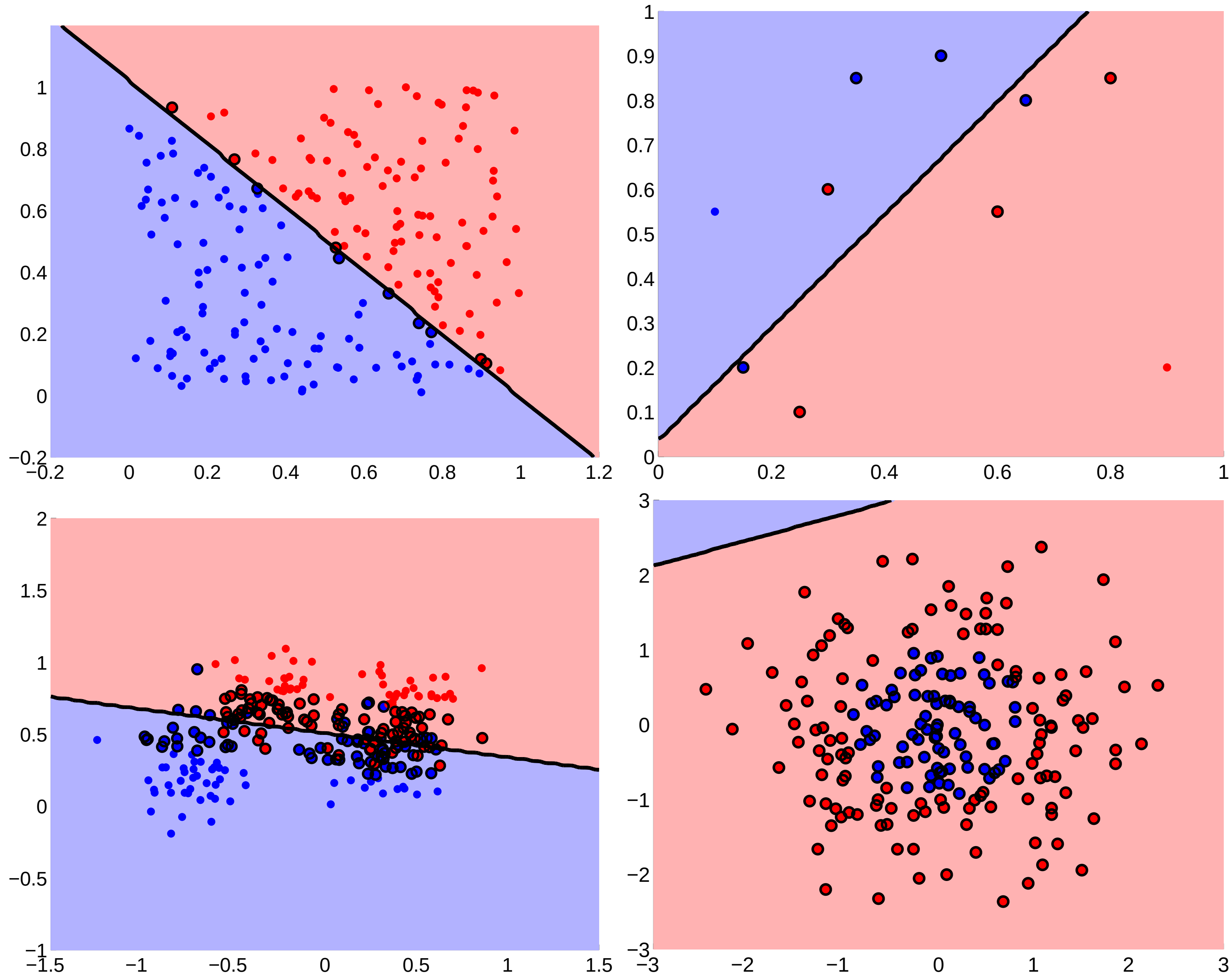
=

optimální rozdělující nadplocha

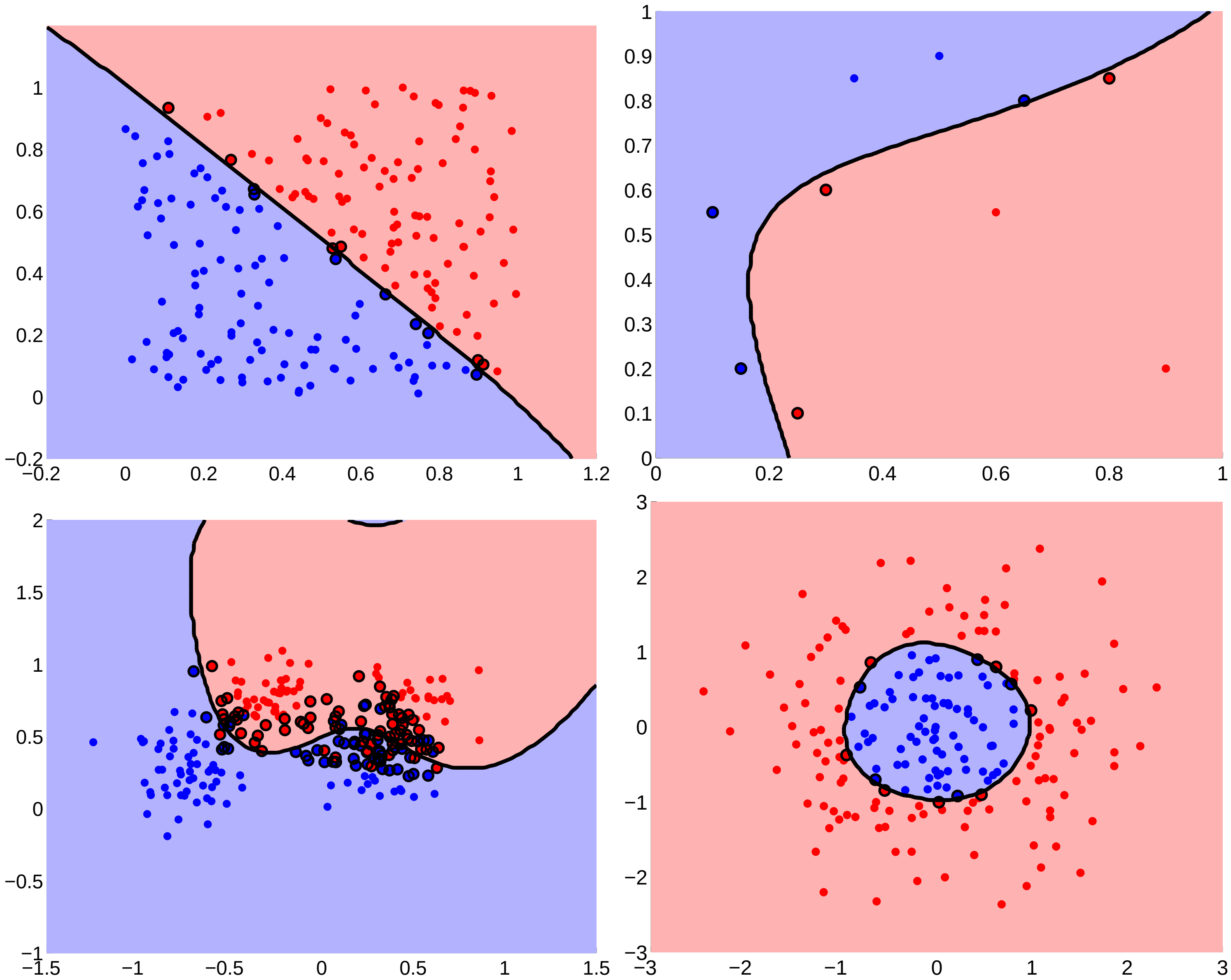
+

kernel trik

Demo: SVM s lineárním jádrem



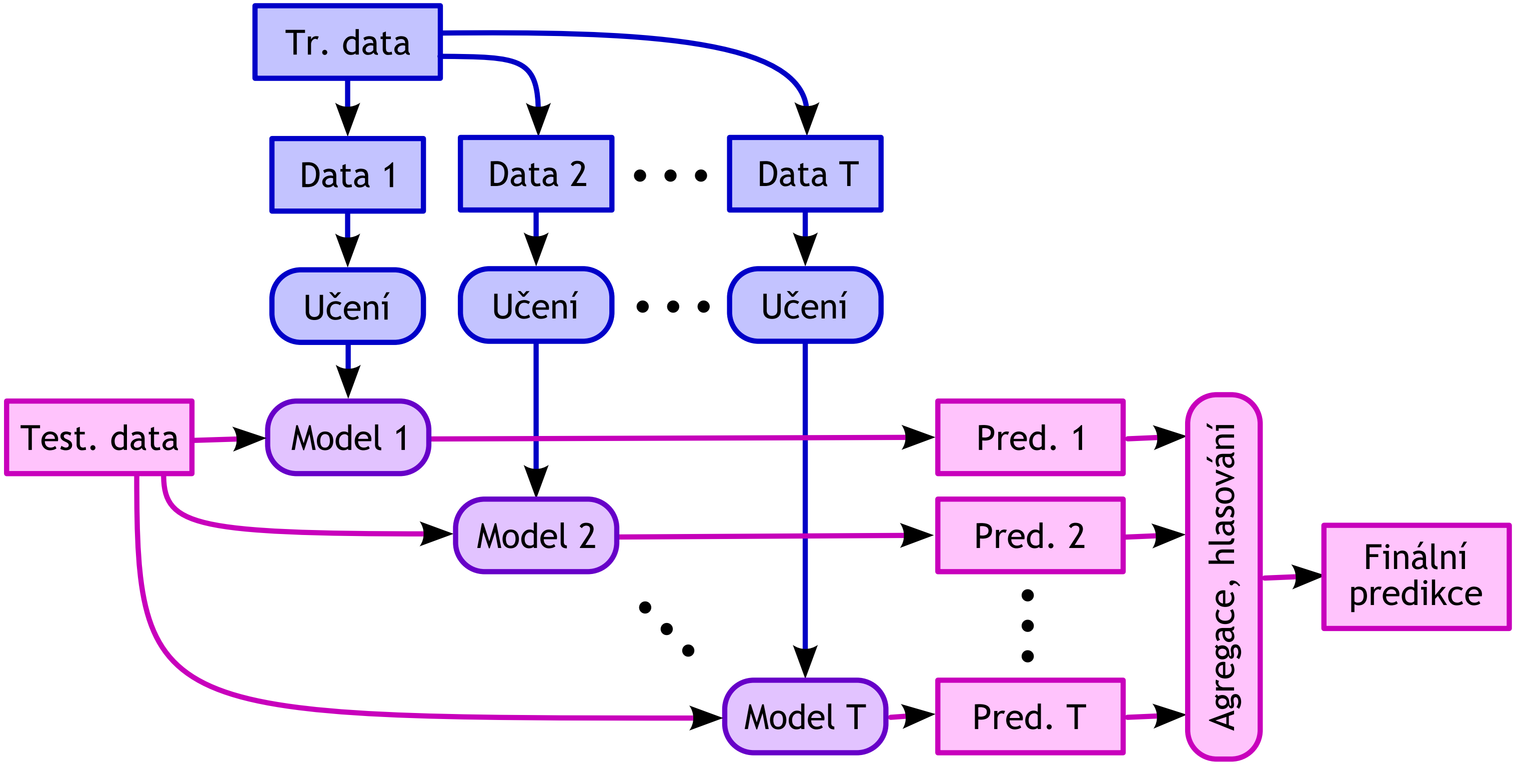
Demo: SVM s Gaussovským (RBF) jádrem



Boosting

Ensembles, committees

Ensemble je komise různých modelů, jejichž predikce jsou sloučeny dohromady např. hlasováním nebo vážením.



Jednotlivé metody si liší tím, jakým způsobem vytvářejí jednotlivé modely tak, aby byly *různé*.

Boosting

Boosting

- ✔ obecná metoda, jak ze *slabých klasifikátorů* vytvořit *silný klasifikátor*
- ✔ všechny modely jsou stejného typu, všechny jsou učeny stejným algoritmem
- ✔ různorodost modelů: každý z T modelů se učí na jinou podmnožinu trénovacích dat
- ✔ máme-li algoritmus pro učení slabého klasifikátoru, který umí konzistentně nacházet modely, jejichž přesnost je alespoň o málo větší než náhodné tipování, lze dokázat, že boosting umí vytvořit klasifikátor s vysokou úspěšností, např. 99 %.

AdaBoost

- ✔ trénovací data
  - ✘ všechny modely se učí na všech trénovacích datech (nikoli jen na podmnožinách)
  - ✘ v každé iteraci $t = 1, \ldots, T$ se liší váhy $D_t(i)$ trénovacích bodů $\mathbf{x}_i$
  - ✘ špatně zaklasifikované body mají v další iteraci větší váhu
- ✔ výsledný klasifikátor
  - ✘ vážené hlasování

Algoritmus 1: AdaBoost

Vstup: Trénovací množina správně ohodnocených bodů: $\{\mathbf{x}_i, y_i\}, \mathbf{x}_i \in \mathcal{R}^D, y_i \in \{+1, -1\}$, $i = 1, \ldots, m$

Výstup: Finální klasifikátor $H_{\text{final}}(\mathbf{x}) = \text{sign}\left(\sum_{t=1}^{T} \alpha_t h_t(\mathbf{x})\right)$

begin
- Inicializuj váhy trénovacích případů: $D_1(i) = \frac{1}{m}$.
- **for** $t = 1, \ldots, T$ **do**
  - Najdi slabý klasifikátor h_t
  - Spočti váženou chybu: $\epsilon_t = \sum_{i=1}^{m} D_t(i) I\left(y_i \neq h_t(\mathbf{x}_i)\right)$
  - Spočti váhu modelu h_t:

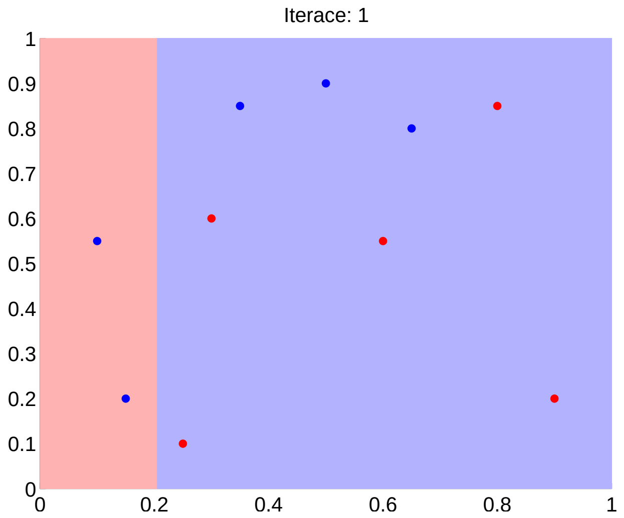
$$\alpha_t = \frac{1}{2} \ln\left(\frac{1-\epsilon_t}{\epsilon_t}\right) > 0$$

  - Uprav váhy trénovacích bodů:

$$\begin{aligned} D_{t+1}(i) &= \frac{D_t(i)}{Z_t} \times \begin{cases} e^{-\alpha_t}, & \text{pokud } y_i = h_t(\mathbf{x}_i), \\ e^{\alpha_t}, & \text{pokud } y_i \neq h_t(\mathbf{x}_i), \end{cases} \\ &= \frac{D_t(i)}{Z_t} \times \exp\left(-\alpha_t y_i h_t(\mathbf{x}_i)\right), \end{aligned}$$

    kde Z_t je normalizační konstanta.

Iterace 1:

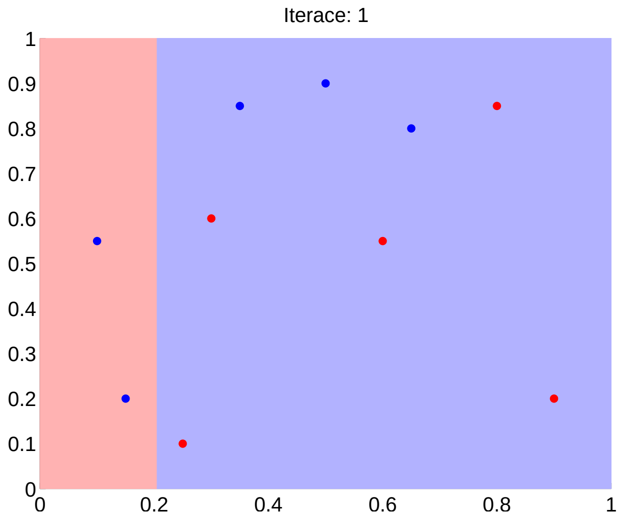


$$\epsilon_1 = 0.3$$

$$\alpha_1 = 0.42$$

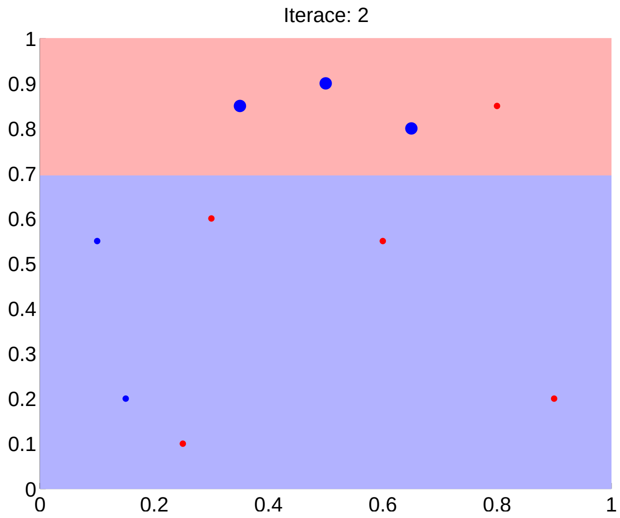
AdaBoost graficky

Iterace 1:



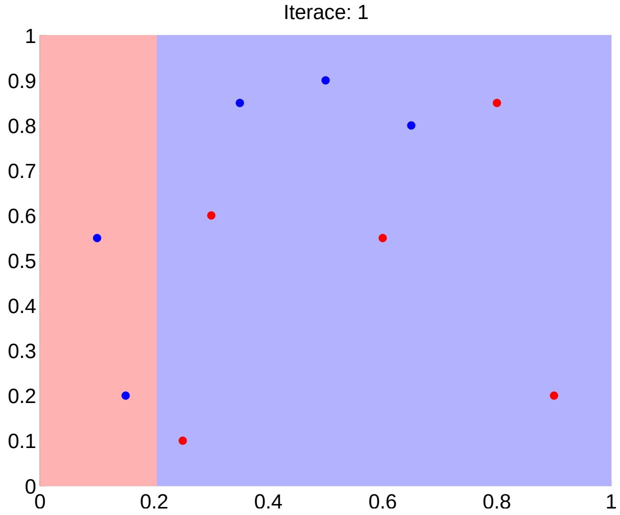
$$\epsilon_1 = 0.3$$
$$\alpha_1 = 0.42$$

Iterace 2:



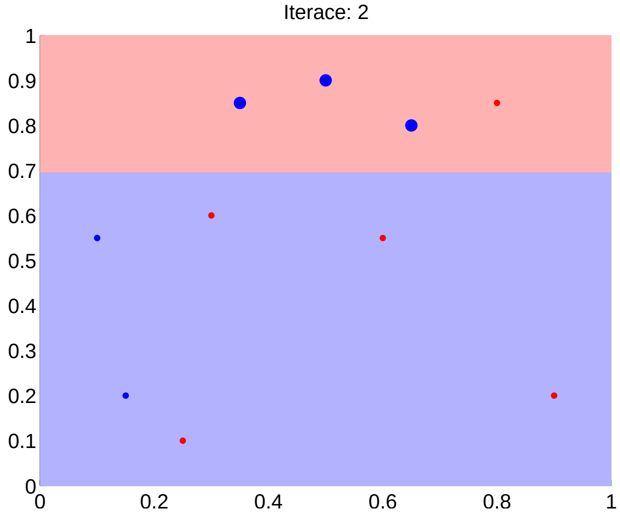
$$\epsilon_2 = 0.21$$
$$\alpha_2 = 0.65$$

Iterace 1:



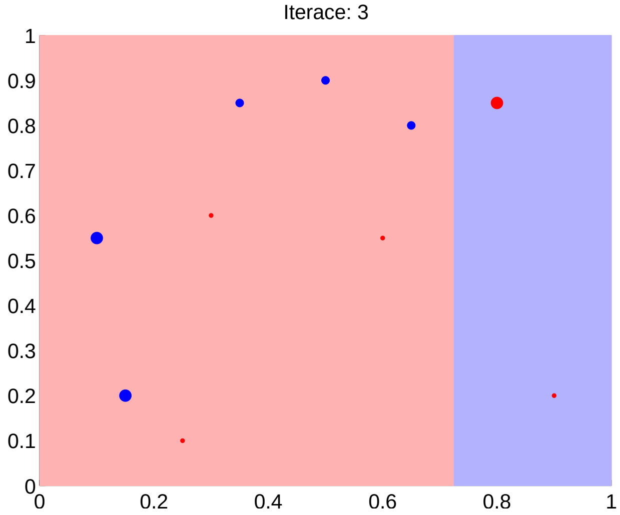
$$\epsilon_1 = 0.3$$
$$\alpha_1 = 0.42$$

Iterace 2:



$$\epsilon_2 = 0.21$$
$$\alpha_2 = 0.65$$

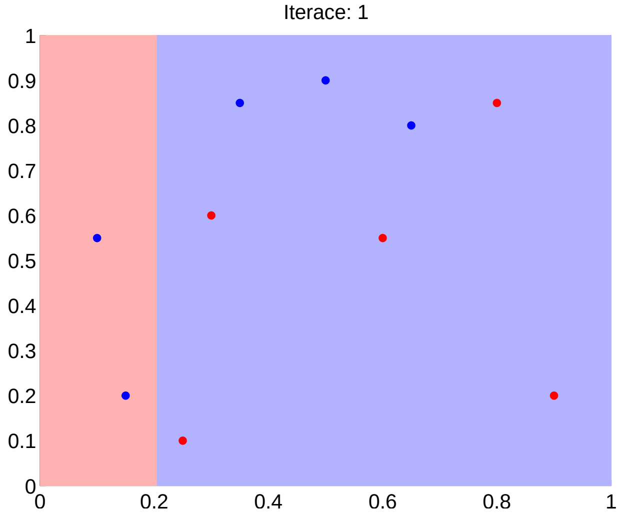
Iterace 3:



$$\epsilon_3 = 0.13$$
$$\alpha_3 = 0.92$$

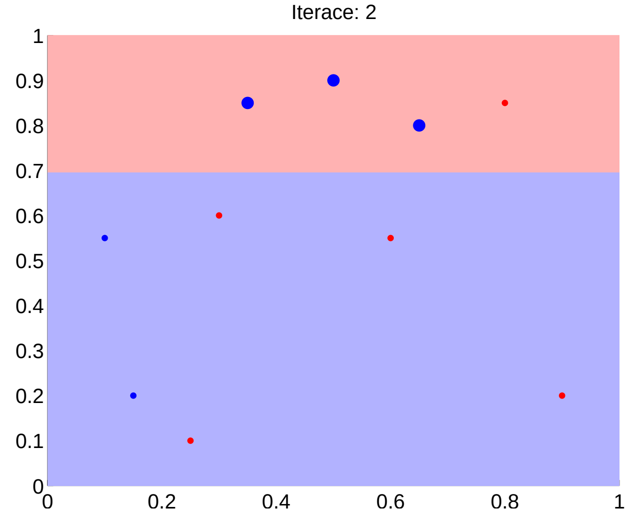
AdaBoost graficky

Iterace 1:



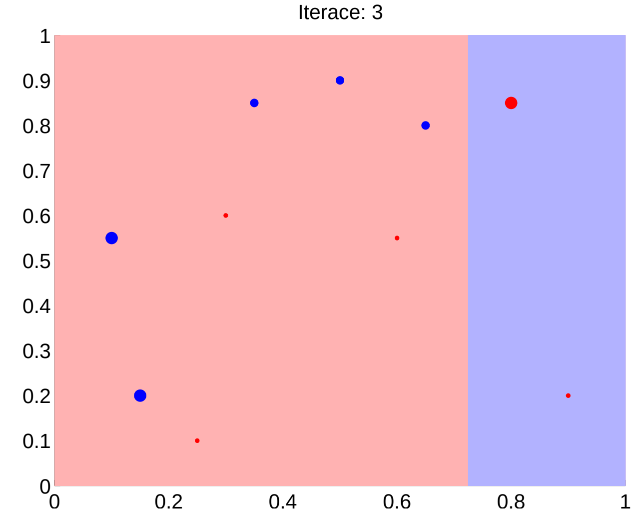
$$\epsilon_1 = 0.3$$
$$\alpha_1 = 0.42$$

Iterace 2:



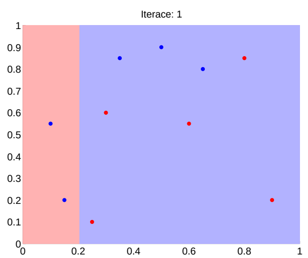
$$\epsilon_2 = 0.21$$
$$\alpha_2 = 0.65$$

Iterace 3:

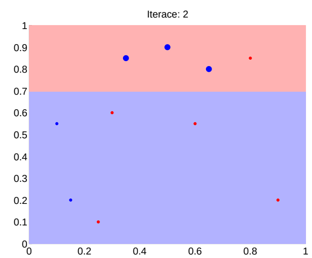


$$\epsilon_3 = 0.13$$
$$\alpha_3 = 0.92$$

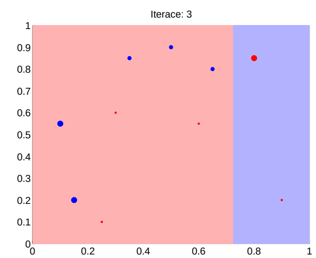
$0.42\times$



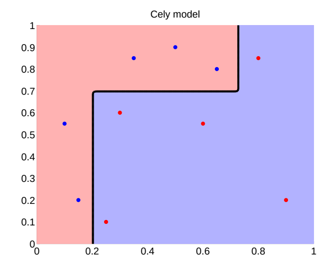
$+0.65\times$



$+0.92\times$



$=$



AdaBoost: poznámky

Trénovací chyba:

- ✔ označme $\gamma_t = 0.5 - \epsilon_t$, o co je t. model lepší než náhodné hádání
- ✔ označme $\gamma = \min_t \gamma_t$, neboli všechny modely $h(t)$ jsou na trénovací sadě úspěšnější než náhodné hádání alespoň o γ (platí také $\forall \gamma_t : \gamma_t \geq \gamma > 0$)
- ✔ lze ukázat, že trénovací chyba $\mathrm{Err}_{\mathrm{Tr}}(H_{\mathrm{final}}) \leq e^{-2\gamma^2 T}$

Závěr

Metody pro binární klasifikaci objektů popsaných *reálnými příznaky*:

- ✔ učení lineární diskriminační funkce
  - ✘ perceptron (pomalu konverguje)
  - ✘ optimální rozdělující nadplocha (úloha QP)
- ✔ využití lineárních metod pro konstrukci nelineárních modelů
  - ✘ rozšíření báze
  - ✘ kernel trik (SVM)
  - ✘ ensembles (AdaBoost)

Poznámky:

- ✔ SVM: existují jádra umožňující využít SVM i pro nenumerické příznaky (např. pro klasifikaci textů)
- ✔ AdaBoost: algoritmus je možné využít i pro kategoriální příznaky