

## Support vector machines (SVM)

(Algoritmy podpůrných vektorů)

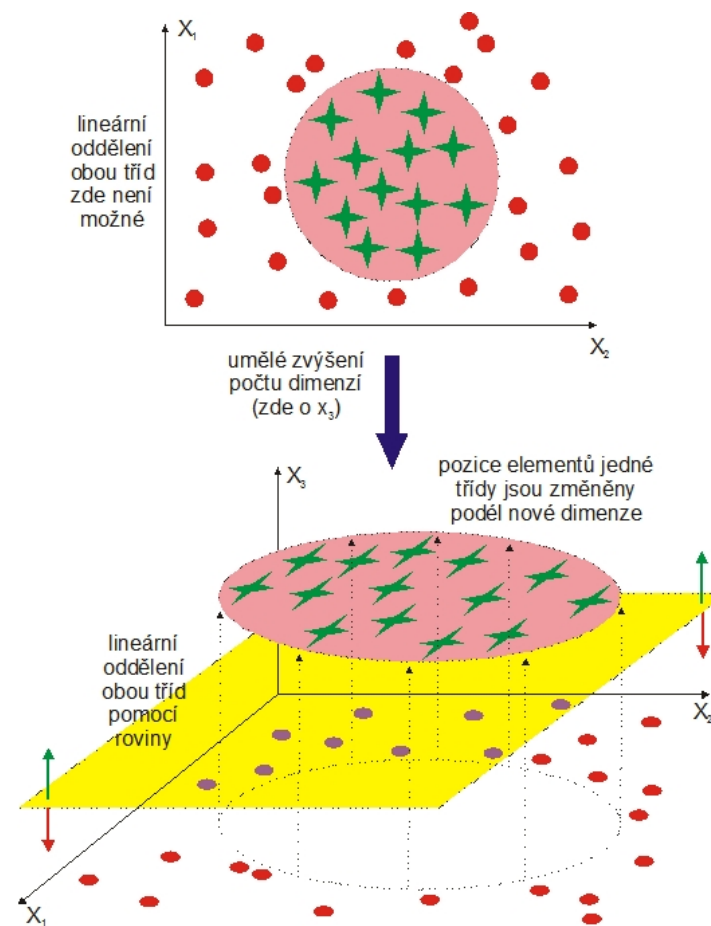
Řada metod strojového učení se vyznačuje jednoduchými a efektivními algoritmy učení, např. jednovrstvé umělé neuronové sítě. Ale z hlediska řešení obecné úlohy najít hranice, které oddělují určité třídy ve vstupním prostoru, jsou velmi silně omezeny schopností naučit se pouze lineární oddělovače (přímky, roviny, nadroviny).

Na druhé straně existují metody jako mnohvrstvé umělé neuronové sítě (např. sigmoidální trénované zpětným šířením chyb – backpropagation), které jsou schopny reprezentovat obecné nelineární funkce. Jejich nevýhodou však je často velmi obtížné učení, protože prakticky vždy existuje riziko uvíznutí v lokálním minimu chybové funkce a navíc je učení silně komplikováno hledáním vysokého počtu vah v mnohadimensionálním prostoru.

K alternativním, relativně novým metodám patří podpůrné vektory (*support vector machines*, SVM), které tvoří určitou kategorii tzv. jádrových algoritmů (*kernel machines*). Tyto metody se snaží využít výhody poskytované efektivními algoritmy pro nalezení lineární hranice a zároveň jsou schopny reprezentovat vysoce složité nelineární funkce. Jedním ze základních principů je převod daného původního vstupního prostoru do jiného, vícedimensionálního, kde již lze od sebe oddělit třídy lineárně.

Tato myšlenka je v podstatě jednoduchá, jak ukazuje obrázek obr. 1. V původním dvourozměrném prostoru jsou dvě třídy, oddělené nelineárně kružnicí. Přidáním další dimenze vznikne možnost prvkům třídy uvnitř kružnice přidat další souřadnici, která je posune např. nahoru podél nové osy  $x_3$ , takže pro oddělení obou tříd již lze použít rovinu rovnoběžnou s rovinou danou osami  $x_1$  a  $x_2$ .

Otázka je, kam nejlépe lineární hranici umístit tak, aby byla vedena co nejefektivněji z hlediska kategorizace budoucích dat, která nebylo při tréninku možno použít. Samotná optimalizace umístění hranice je záležitost komplikovanější, ale v zásadě řešitelná.

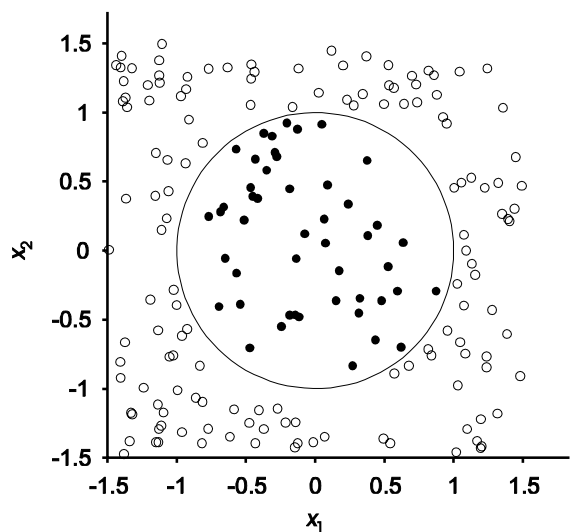


Obr. 1. Princip vzniku možnosti lineárního oddělení dvou tříd s nelineárními hranicemi pomocí přidané dimenze.

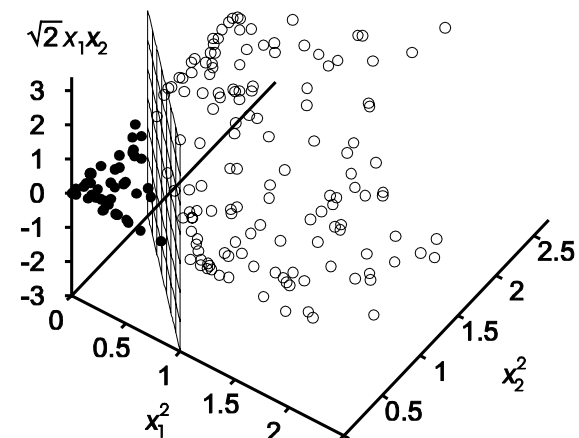
Obr. 2 znázorňuje dvourozměrný vstupní prostor definovaný atributy  $\mathbf{x} = (x_1, x_2)$ , přičemž pozitivní trénovací příklady, dávající hodnotu  $y = +1$ , jsou uvnitř kruhu a negativní ( $y = -1$ ) vně. Je zřejmé, že lineární oddělovací hranice neexistuje. Když se vstupní data vhodně modifikují, může vzniknout nový vstupní prostor, kde jsou pozitivní i negativní příklady oddělitelné lineárně. Modifikací může být mapování vstupního vektoru  $\mathbf{x}$  do nového vektoru s jinými hodnotami atributů:  $F(\mathbf{x})$ .

Konkrétně zde lze každý dvourozměrný vektor změnit přidáním třetího atributu založeného na prvních dvou, takže místo původních dvou atributů  $(x_1, x_2)$  budou tři, definované následujícími funkcemi  $f_1, f_2, f_3$ :

$$f_1 = x_1^2, \quad f_2 = x_2^2, \quad f_3 = \sqrt{2}x_1x_2.$$



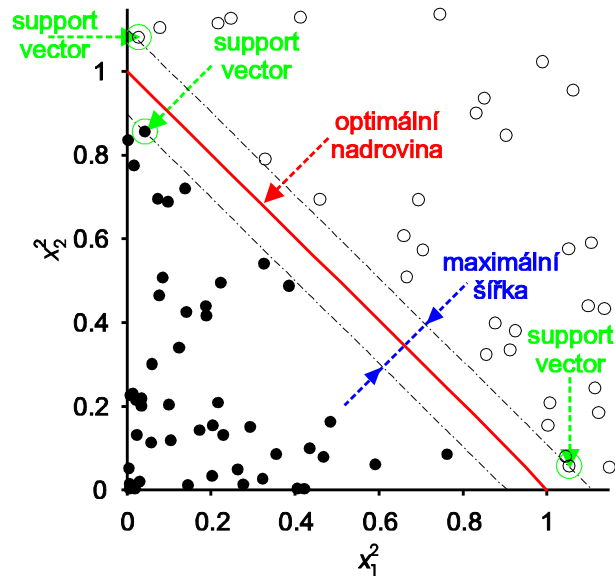
Obr. 2. Dvourozměrný prostor s trénovacími příklady pozitivními ● a negativními ○. Skutečná oddělovací hranice je představována funkcí  $x_1^2 + x_2^2 \leq 1$ .



Obr. 3. Tatáž data mapovaná do trojrozměrného prostoru  $(x_1^2, x_2^2, \sqrt{2}x_1x_2)$ . Kruhová rozhodovací hranice z obr. 2 se stane lineární.

Obrázek 3 ukazuje původní data v novém prostoru, a důležité je, že nyní jsou obě třídy lineárně separovatelné. Tento jev je obecný: při mapování do prostoru s dostatečným počtem dimenzí lze nakonec vždy najít lineární oddělovač (nadrovinu). V příkladu byly použity pouze tři dimenze (i když konkrétně zde by se daly použít dokonce jen dvě, tj.  $f_1$  a  $f_2$ ), ale obecně platí, že  $N$  datových bodů je možno vždy (kromě některých speciálních případů) lineárně oddělit v prostoru s  $N-1$  nebo více dimenzemi (důkaz zde nebude probírán).

Řešení je ovšem komplikováno faktem, že v  $d$ -rozměrném prostoru je lineární oddělovač definován rovnicí, která má  $d$  parametrů, takže hrozí nebezpečí, že dojde ke ztrátě obecnosti klasifikátoru “přetrénováním” pokud  $d \approx N$ ; dojde totiž k podobnému efektu, jako např. při prokládání datových bodů polynomem vysokého stupně (aproximace neznámé funkce). Uvedená potíž je proto důvodem k tomu, že zmiňovaná metoda jádrových funkcí se snaží najít *optimální* lineární oddělovač, tj. takový, který poskytuje co nejširší pásmo mezi ním a pozitivními příklady na jedné straně a negativními na druhé (a zároveň podporuje robustnost klasifikace), jak ilustruje následující obr. 4.



Obr. 4. Pohled na optimální oddělovací hranici (viz obr. 3) vzniklý projekcí do prvních dvou dimenzí.

Optimální lineární oddělovač se v algoritmu *support vector machines* hledá pomocí metody *kvadratického programování*. Zde je jistá obdoba s hledáním maxima jako u lineárního programování, problém je však složitější. Předpokládejme, že jsou příklady  $\mathbf{x}_i$  s klasifikací  $y_i = \pm 1$  a cílem je najít optimální oddělovač. Problém lze převést na hledání hodnot parametrů  $\alpha_i$ , které maximalizují výraz

$$\sum_i \alpha_i - \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j y_i y_j (\mathbf{x}_i \cdot \mathbf{x}_j),$$

přičemž platí omezení

$$\alpha_i \geq 0 \text{ a } \sum_i \alpha_i y_i = 0.$$

Výraz má dvě významné vlastnosti: jednak má jediné globální maximum, které lze efektivně najít, a dále datové body do výrazu vstupují ve formě skalárního součinu jednotlivých dvojic. Tato druhá vlastnost platí i pro rovnici lineárního oddělovače.

Jakmile jsou jednou spočteny optimální parametry  $\alpha_i$ , je oddělovač dán rovnicí

$$h(\mathbf{x}) = \text{sign} \left( \sum_i \alpha_i y_i (\mathbf{x} \cdot \mathbf{x}_i) \right).$$

Vzhledem k omezením při hledání  $\alpha_i$  platí, že lineární oddělovač má *nulové* váhy  $\alpha_i$  pro každý datový bod kromě těch bodů, které jsou nejbližší vlastnímu oddělovači. Tyto nejbližší body se právě nazývají **support vectors** – v překladu **podpůrné vektory** vzhledem k tomu, že jejich funkcí je “podpora” oddělovací nadroviny, která na nich spočívá; ostatní body-vektory nejsou pro oddělovač vůbec zapotřebí, takže metoda SVM je schopna najít ty *trenovací příklady*, které jsou pro nalezení oddělovače tříd *podstatné* (obvykle učící algoritmy používají *všechny* trenovací příklady, což při vysokém počtu bodů může vést k nízké efektivitě).

Velkou výhodou je skutečnost, že těchto podpůrných vektorů je obvykle mnohem méně než datových bodů, takže efektivní počet parametrů definujících optimální oddělovač je pak mnohem menší než  $N$ .

Běžně nelze očekávat, že lineární oddělovač bude nalezen přímo v originálním vstupním prostoru  $\mathbf{x}$  (i když to nelze vyloučit). Je ovšem zřejmé, že lineární oddělovače lze hledat ve vícerozměrném prostoru  $F(\mathbf{x})$  jednoduše tím, že se nahradí člen  $\mathbf{x}_i \cdot \mathbf{x}_j$  členem  $F(\mathbf{x}_i) \cdot F(\mathbf{x}_j)$ . Tato náhrada má stejný efekt pro každý učící algoritmus, ale zde vzhledem ke skalárnímu součinu existují zvláštní vlastnosti. Platí totiž, že  $F(\mathbf{x}_i) \cdot F(\mathbf{x}_j)$  není většinou nutné stanovovat přes výpočty  $F$  pro všechny body. Ve výše uvedeném příkladu se třemi dimenzemi lze ukázat, že

$$F(\mathbf{x}_i) \cdot F(\mathbf{x}_j) = (\mathbf{x}_i \cdot \mathbf{x}_j)^2.$$

Výraz  $(\mathbf{x}_i \cdot \mathbf{x}_j)^2$  se nazývá **jádrová funkce** (kernel function)  $K(\mathbf{x}_i, \mathbf{x}_j)$ . V souvislosti s algoritmy SVM je to funkce, která může být aplikována na dvojice vstupních dat k vyhodnocení skalárního součinu v nějakém odpovídajícím prostoru. Z toho tedy plyne, že lze najít lineární oddělovače ve vícerozměrném prostoru  $F(\mathbf{x})$  jednoduše náhradou  $\mathbf{x}_i \cdot \mathbf{x}_j$  jádrovou funkcí  $K(\mathbf{x}_i, \mathbf{x}_j)$ . Jinak řečeno, lze provádět učení ve vícerozměrném prostoru, ale stačí počítat pouze jádrové funkce místo úplného seznamu atributů pro každý datový bod.

Na uvedené jádrové funkci  $K(\mathbf{x}_i, \mathbf{x}_j) = (\mathbf{x}_i \cdot \mathbf{x}_j)^2$  není nic zvláštního, pouze odpovídá jednomu z možných vícerozměrných prostorů; zároveň je řada dalších jádrových funkcí, které odpovídají jiným prostorům. Existuje tzv. Mercerův teorém, který říká, že jakákoliv “rozumná” jádrová funkce [tj. v principu taková, kde matice  $\mathbf{K}_{ij} = K(\mathbf{x}_i, \mathbf{x}_j)$  je pozitivně definitní] odpovídá nějakému prostoru atributů. Tyto prostory mohou být velmi vysoce rozsáhlé, například polynomická jádrová funkce  $K(\mathbf{x}_i, \mathbf{x}_j) = (1 + \mathbf{x}_i \cdot \mathbf{x}_j)^d$  odpovídá prostoru atributů, jehož dimenze roste exponenciálně s  $d$ . S použitím takovýchto jádrových funkcí lze nalézt lineární oddělovače efektivně v prostorech majících miliardy (nebo v některých případech nekonečně mnoho) rozměrů.

Nalezené lineární oddělovače lze samozřejmě mapovat zpět do původního prostoru, čímž lze získat libovolně zvlněné, nelineární hranice mezi pozitivními a negativními příklady.

Jakýkoliv algoritmus, který lze převést na uvedený skalární součin, může být nahradou tohoto součinu jádrovou funkcí převeden na jádrovou (kernelized) verzi, např.  $k$ -NN (nejbližší sused), perceptronové učení, a další.

Algoritmus využívající jádrové funkce je v praxi využíván např. pro rozeznávání rukou psaných číslic nebo na úlohy s velmi vysokým počtem atributů, úspěšně funguje také jako filtr např. textových dokumentů, které se často vyznačují desítkami tisíc atributů (tj. různých slov), apod.

Existuje dostupný software, např. velmi často se používá program, jehož autorem je Joachim Thorsten; více informací viz URL <http://www.joachims.org/>

Binární verze lze pro Linux a Windows získat zdarma (ovšem pouze pro akademické účely) na následujících URL:

```
ftp://ftp-ai.cs.uni-dortmund.de/pub/Users/thorsten/  
svm_light/current/svm_light_linux.tar.gz
```

```
ftp://ftp-ai.cs.uni-dortmund.de/pub/Users/thorsten/  
svm_light/current/svm_light_windows.zip
```