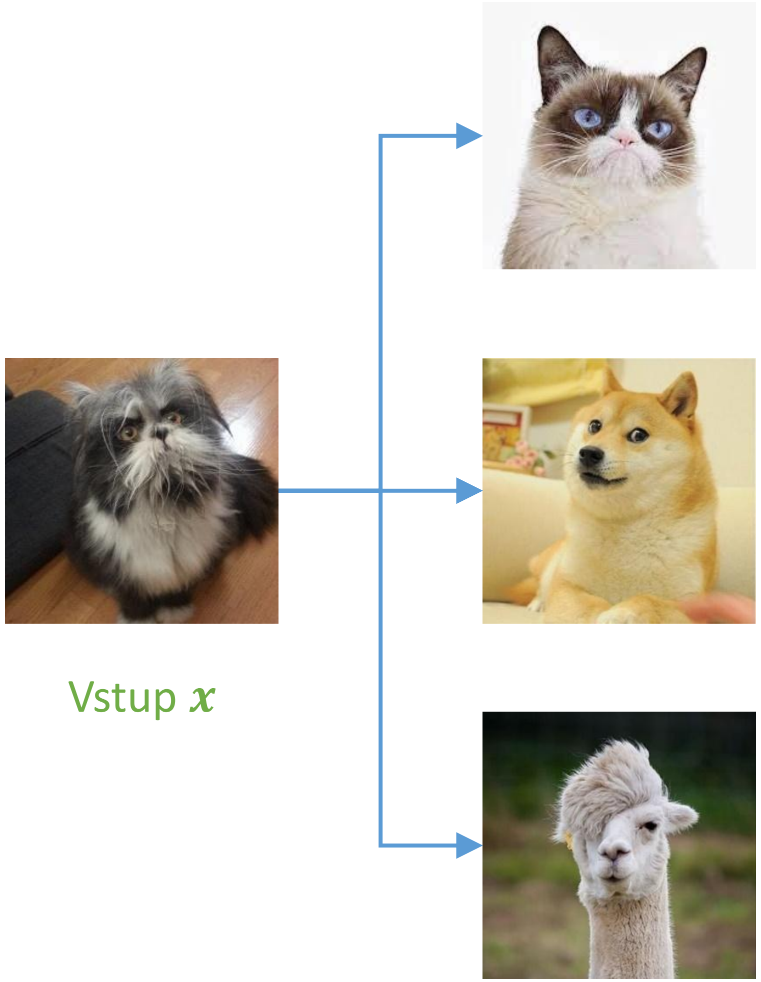
Aplikace neuronových sítí

Lineární klasifikace, Softmax, SVM

Úloha klasifikace obrázků

- Vstupem $\boldsymbol{x}_n$ je RGB obrázek
- Úkolem zařadit $\boldsymbol{x}_n$ do jedné ze tří předdefinovaných kategorií (tříd):
    1. “kočka”
    2. “pes”
    3. “*alpaka*”
- Počet tříd označíme jako K
- Výstupem bude celé číslo $\hat{y}_n \in \{1, \dots, K\}$



Vstup $\boldsymbol{x}$

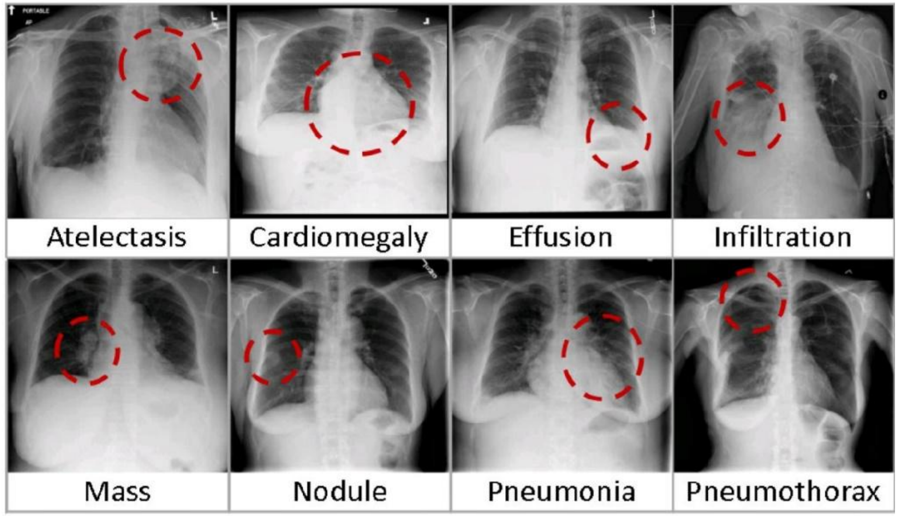
třída 1

třída 2

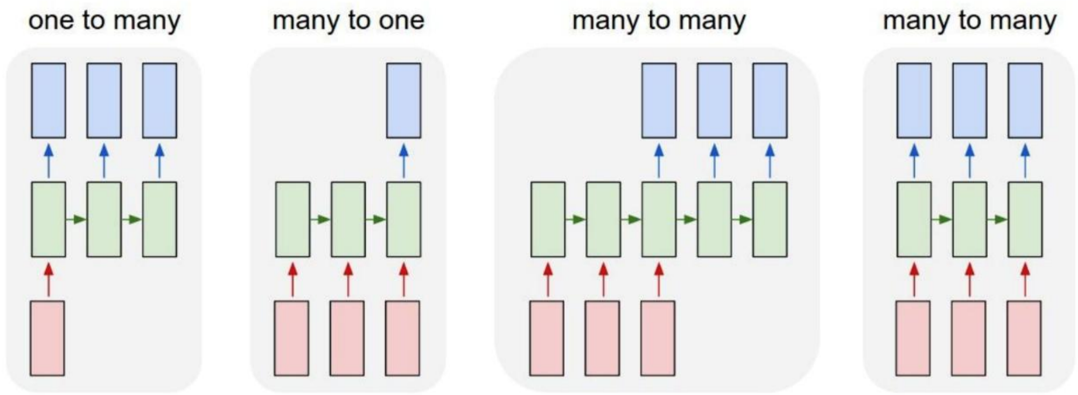
třída 3

Proč klasifikace?

Klasifikace je zajímavá a užitečná sama o sobě



zároveň ale tvoří základ mnoha dalších aplikací



tagování obrázku, generování hudby

analýza sentimentu, klasifikace videa

strojový překlad, rozpoznávání řeči

rozpoznávání fonémů, jazykový model

Návrh a trénování lineárního klasifkátoru

1. **Navrhneme diskriminativní klasifikační funkci s upravitelnými parametry**
2. Kvantifikujeme její úspěšnost klasifikace nějakým kritériem
3. Nastavíme parametry klasifikátoru tak, abychom optimalizovali zvolené kritérium

Lineární diskriminativní klasifikace

Diskriminativní klasifikace

- Zavedeme skóre $s_{n,k}$, které bude udávat, jak moc vstup $\boldsymbol{x}_n$ patří do třídy k
- Skóre $s_{n,k}$ bude reálné číslo (skalár) v intervalu $(-\infty, +\infty)$, tedy *ne pravděpodobnost*
- Jednotlivá skóre $s_{n,k}$ uspořádáme jako sloupcový vektor o délce (rozměru) K

$$\boldsymbol{s}_n = \underbrace{\left[s_{n,1}, \dots, s_{n,K}\right]}_{\text{počet prvků vektoru = počet tříd } K}^{\top}$$

- Výslednou třídu $\hat{y}_n$, do které zařadíme obrázek, vybereme jako tu s maximálním skóre

$$\hat{y}_n = \operatorname*{argmax}_k \boldsymbol{s}_n$$

Lineární klasifikace

- Lineární model předpokládá afinní* vztah mezi skóre třídy $\boldsymbol{s}_n$ a vstupem $\boldsymbol{x}_n$

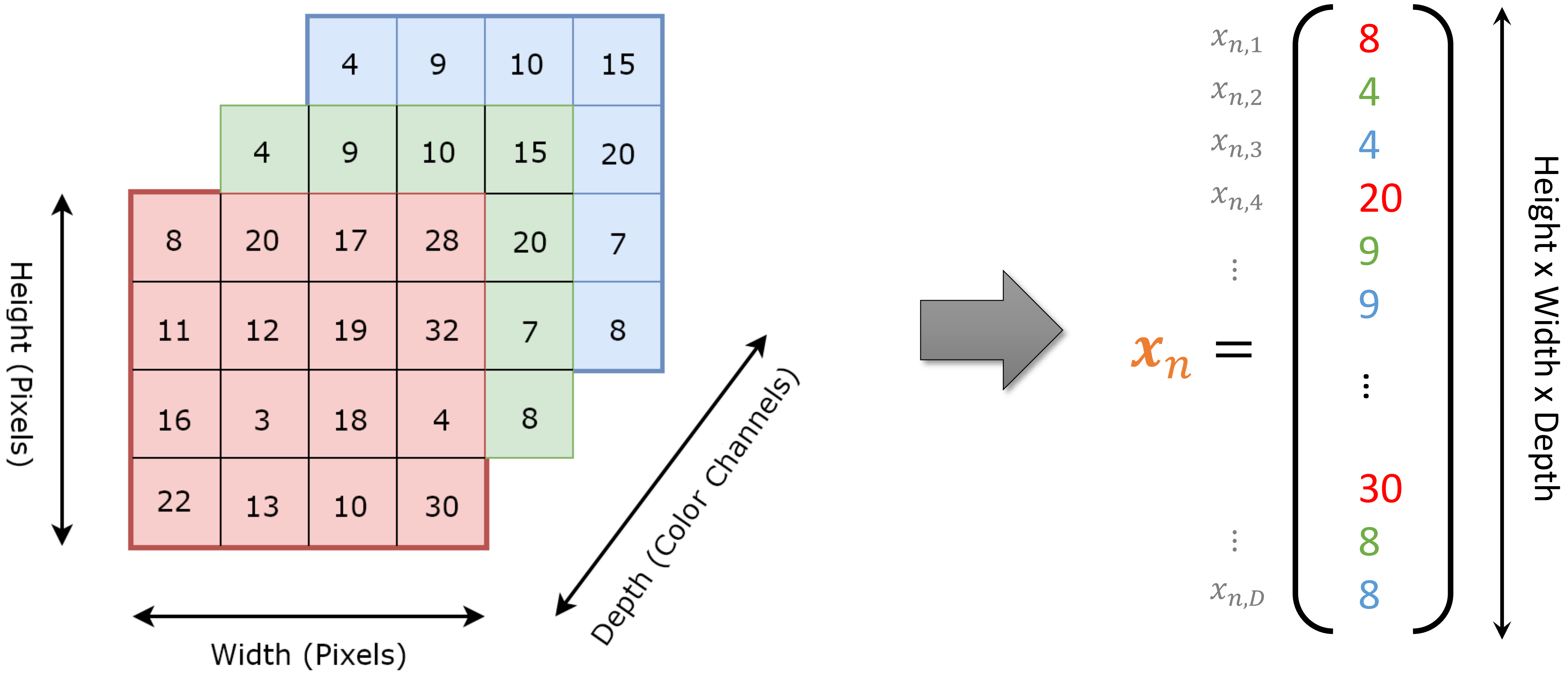
$$\boldsymbol{s}_n = \boldsymbol{w} \cdot \boldsymbol{x}_n + \boldsymbol{b}$$

kde

- $\boldsymbol{x}_n$ je sloupcový vektor o rozměru D
- $\boldsymbol{w}$ je matice vah klasifikátoru s rozměry $K \times D$
- $\boldsymbol{b}$ je sloupcový vektor biasů klasifikátoru s rozměrem K

parametry klasifikátoru

Reprezentace RGB obrázku jako vektoru



Tensor tvaru (výška, šířka, hloubka) (HWC formát)

Vektor délky D = výška x šířka x hloubka

obrázek: https://www.oreilly.com/library/view/machine-learning-with/9781789346565/6899a0ec-2cf9-4525-b3bf-56dfafa827e7.xhtml

Lineární predikce skóre: příklad

$$\boldsymbol{s}_n = \boldsymbol{w} \cdot \boldsymbol{x}_n + \boldsymbol{b}$$

$\boldsymbol{x}_n$: 2×2

$$\begin{bmatrix} 0.2 & -0.5 & 0.1 & 2.0 \\ 1.5 & 1.3 & 2.1 & 0.0 \\ 0.0 & 0.25 & 0.2 & -0.3 \end{bmatrix} \cdot \begin{bmatrix} 56 \\ 231 \\ 24 \\ 2 \end{bmatrix} + \begin{bmatrix} 2.0 \\ 0.0 \\ -0.3 \end{bmatrix} = \begin{bmatrix} -96.8 \\ 437.9 \\ 61.95 \end{bmatrix}$$

$\boldsymbol{w}: 3 \times 4$ $\boldsymbol{x}_n$: 4×1 $\boldsymbol{b}: 3 \times 1$ $\boldsymbol{s}_n: 3 \times 1$

příklad: https://cs231n.github.io/linear-classify/
https://web.eecs.umich.edu/~justincj/slides/eecs498/WI2022/598_WI2022_lecture03.pdf

Lineární predikce skóre: příklad bez biasu

$$\boldsymbol{s}_n = \boldsymbol{w}' \cdot \boldsymbol{x}_n$$

$\boldsymbol{x}_n: 2 \times 2$

$$\begin{bmatrix} 0.2 & -0.5 & 0.1 & 2.0 & \boxed{2.0} \\ 1.5 & 1.3 & 2.1 & 0.0 & \boxed{0.0} \\ 0.0 & 0.25 & 0.2 & -0.3 & \boxed{-0.3} \end{bmatrix} \cdot \begin{bmatrix} 56 \\ 231 \\ 24 \\ 2 \\ \boxed{1} \end{bmatrix} = \begin{bmatrix} -96.8 \\ 437.9 \\ 61.95 \end{bmatrix}$$

$\boldsymbol{w}': 3 \times 5$

$\boldsymbol{x}_n: 5 \times 1$

$\boldsymbol{s}_n: 3 \times 1$

Linearita predikcí*: $f(\alpha \cdot x) = \alpha \cdot f(x)$

$$\boldsymbol{s}_n = \boldsymbol{w}' \cdot \boldsymbol{x}_n$$

$$\boldsymbol{s}_m = \boldsymbol{w}' \cdot \boldsymbol{x}_m = \boldsymbol{w}' \cdot (0.5 \cdot \boldsymbol{x}_n) = 0.5 \cdot s_n$$

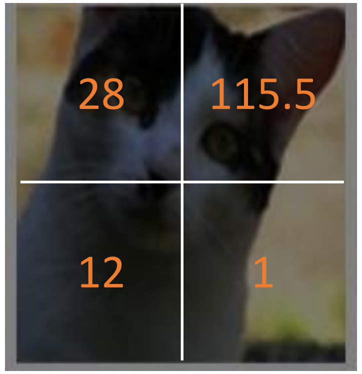
$\boldsymbol{x}_n$

-96.8

437.9

61.95

$\boldsymbol{s}_n$



$\boldsymbol{x}_m = 0.5 \cdot \boldsymbol{x}_n$

-48.4

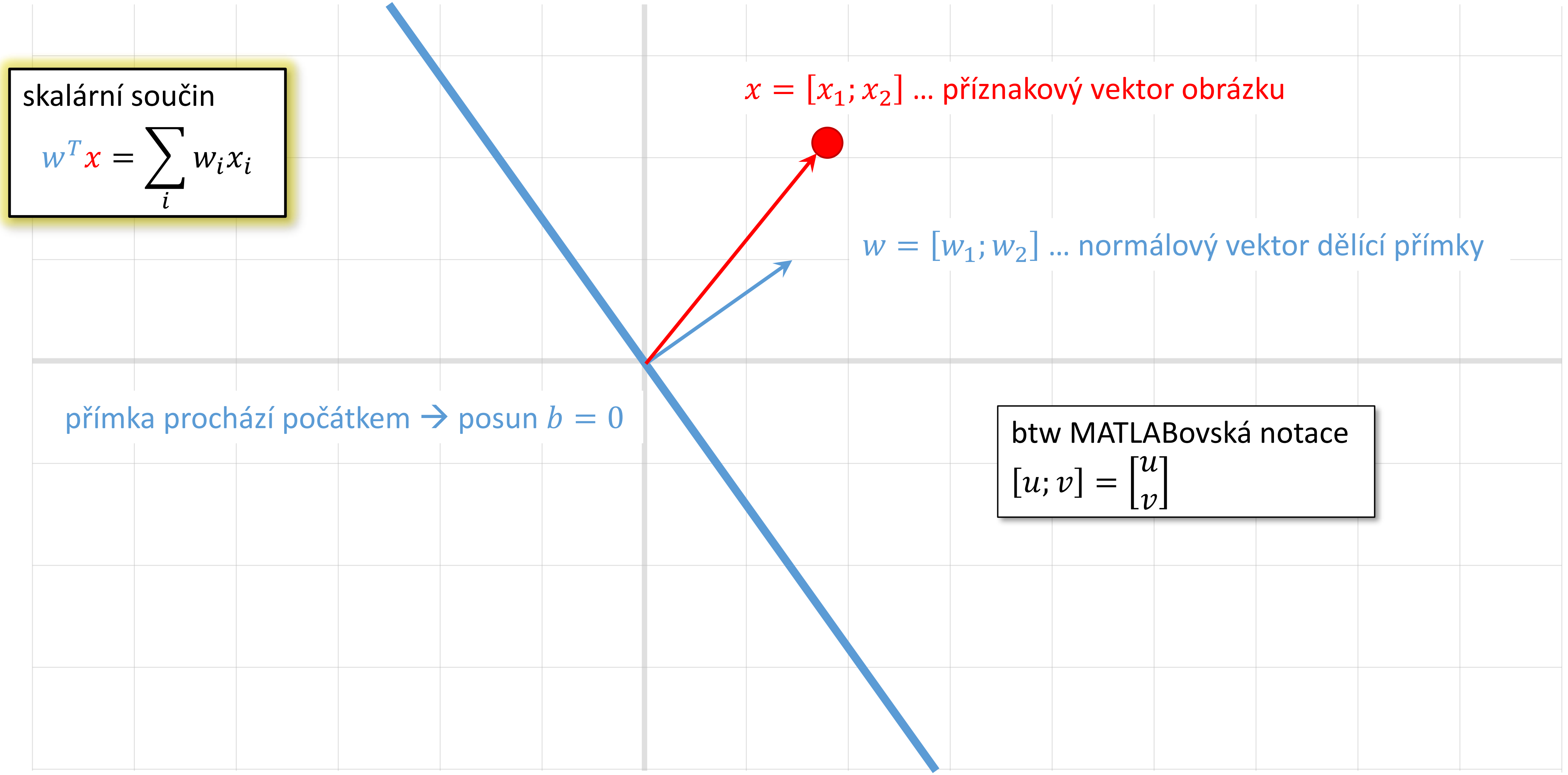
218.9

31.0

$\boldsymbol{s}_m$

*Neplatí pro afinní model s biasem

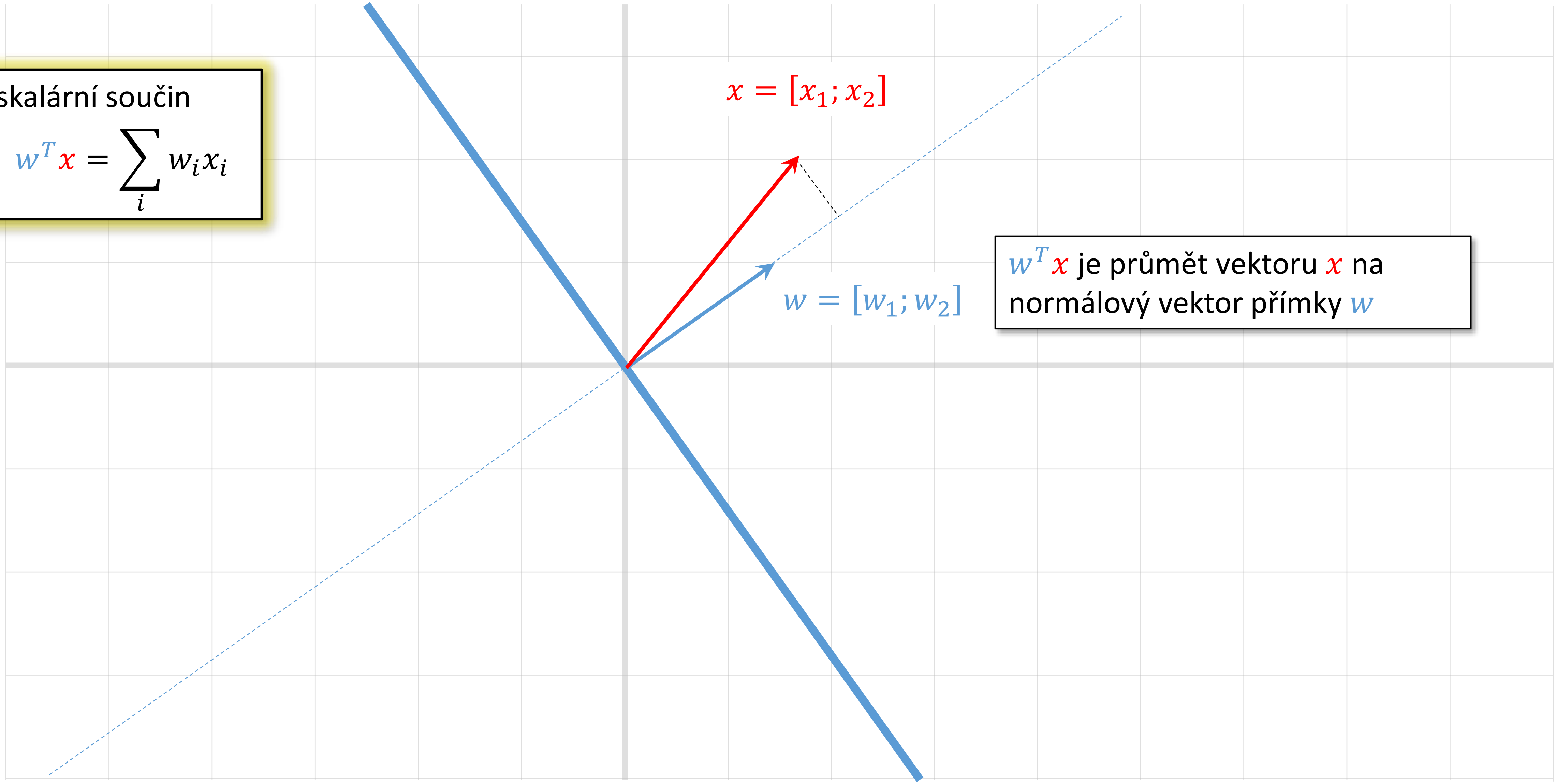
Geometrická interpretace



Geometrická interpretace

skalární součin

$$w^T x = \sum_i w_i x_i$$

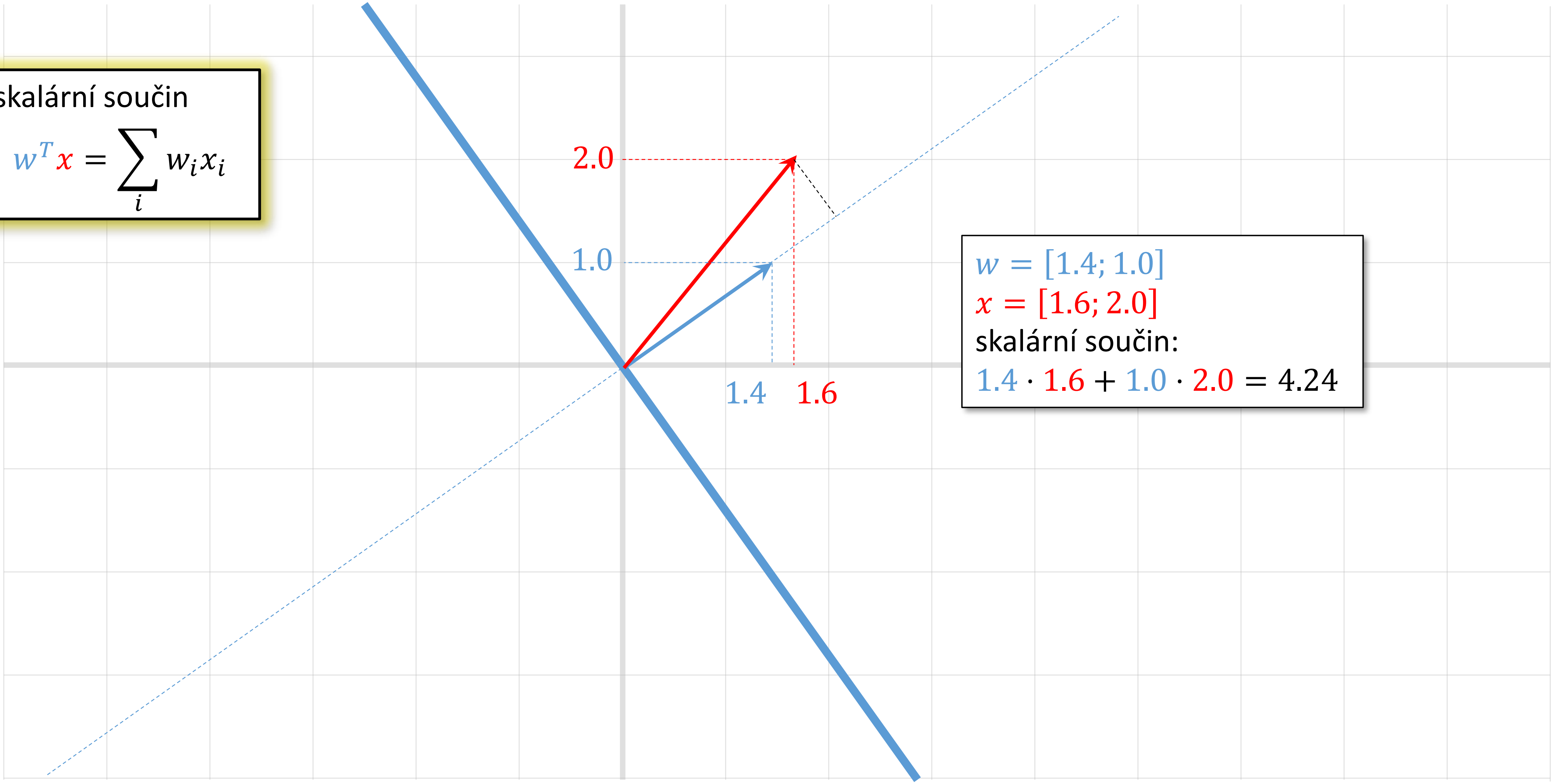


$w^T x$ je průmět vektoru x na normálový vektor přímky w

Geometrická interpretace

skalární součin

$$w^T x = \sum_i w_i x_i$$



$w = [1.4; 1.0]$

$x = [1.6; 2.0]$

skalární součin:

$1.4 \cdot 1.6 + 1.0 \cdot 2.0 = 4.24$

Geometrická interpretace

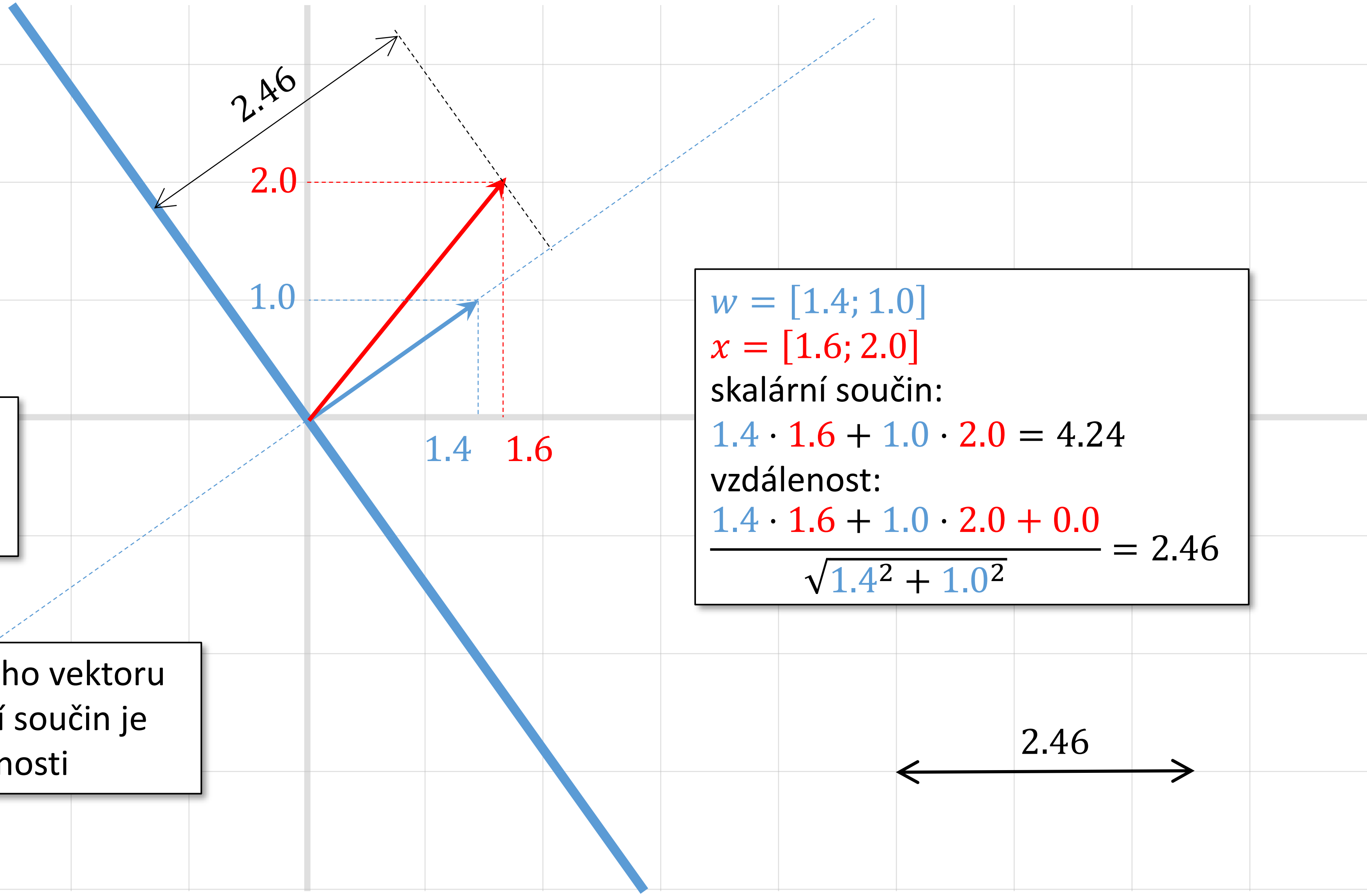
skalární součin

$$w^T x = \sum_i w_i x_i$$

vzdálenost bodu od přímky

$$d(w, x) = \frac{w^T x + b}{\|w\|}$$

pokud je norma normálového vektoru přímky jednotková, skalární součin je roven (orientované) vzdálenosti



$w = [1.4; 1.0]$
$x = [1.6; 2.0]$
skalární součin:
$1.4 \cdot 1.6 + 1.0 \cdot 2.0 = 4.24$
vzdálenost:

$$\frac{1.4 \cdot 1.6 + 1.0 \cdot 2.0 + 0.0}{\sqrt{1.4^2 + 1.0^2}} = 2.46$$

Geometrická interpretace

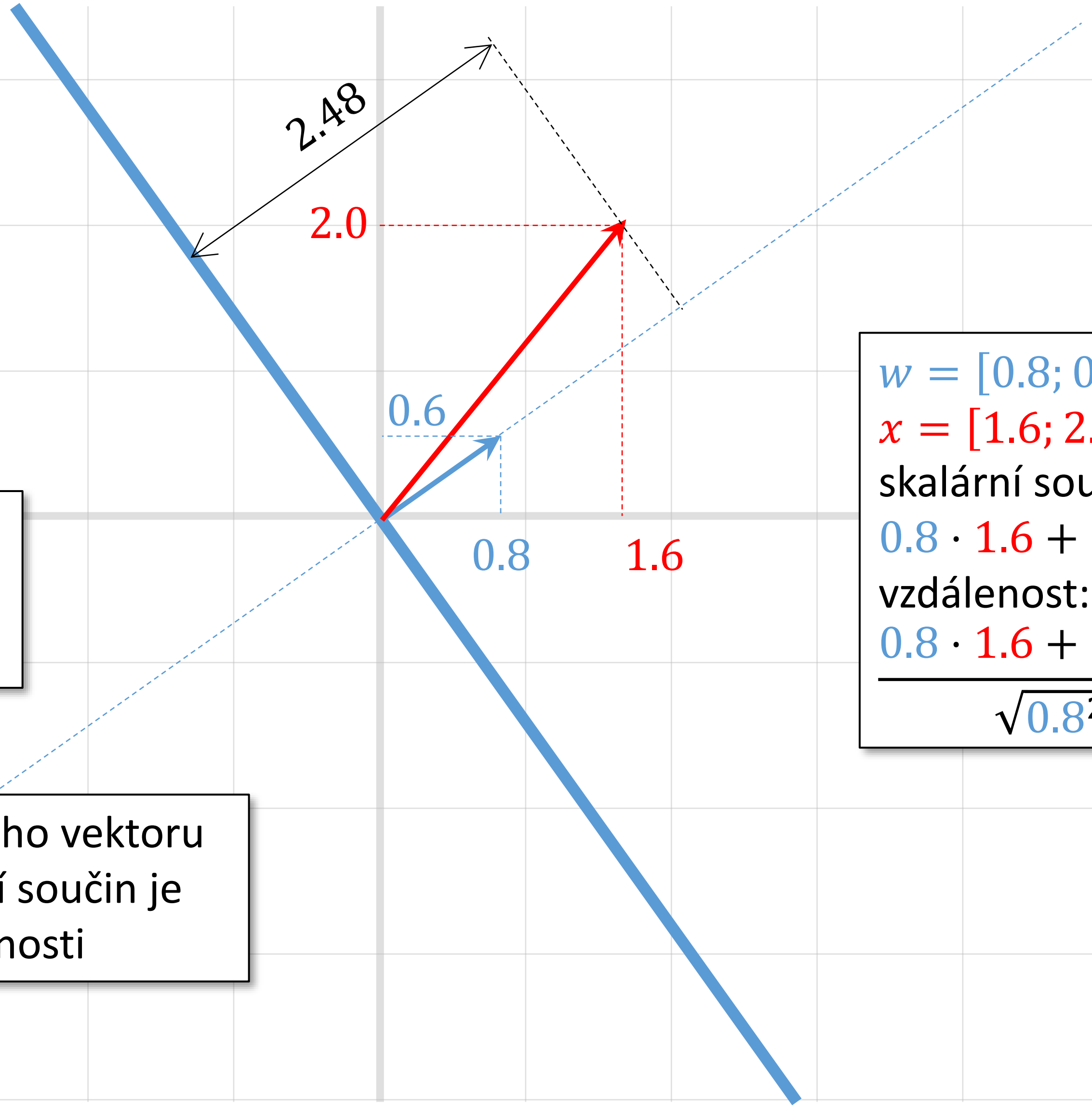
skalární součin

$$w^T x = \sum_i w_i x_i$$

vzdálenost bodu od přímky

$$d(w, x) = \frac{w^T x + b}{\|w\|}$$

pokud je norma normálového vektoru přímky jednotková, skalární součin je roven (orientované) vzdálenosti



$w = [0.8; 0.6]$

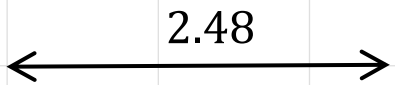
$x = [1.6; 2.0]$

skalární součin: (zaokrouhlování)

$0.8 \cdot 1.6 + 0.6 \cdot 2.0 = 2.48$

vzdálenost:

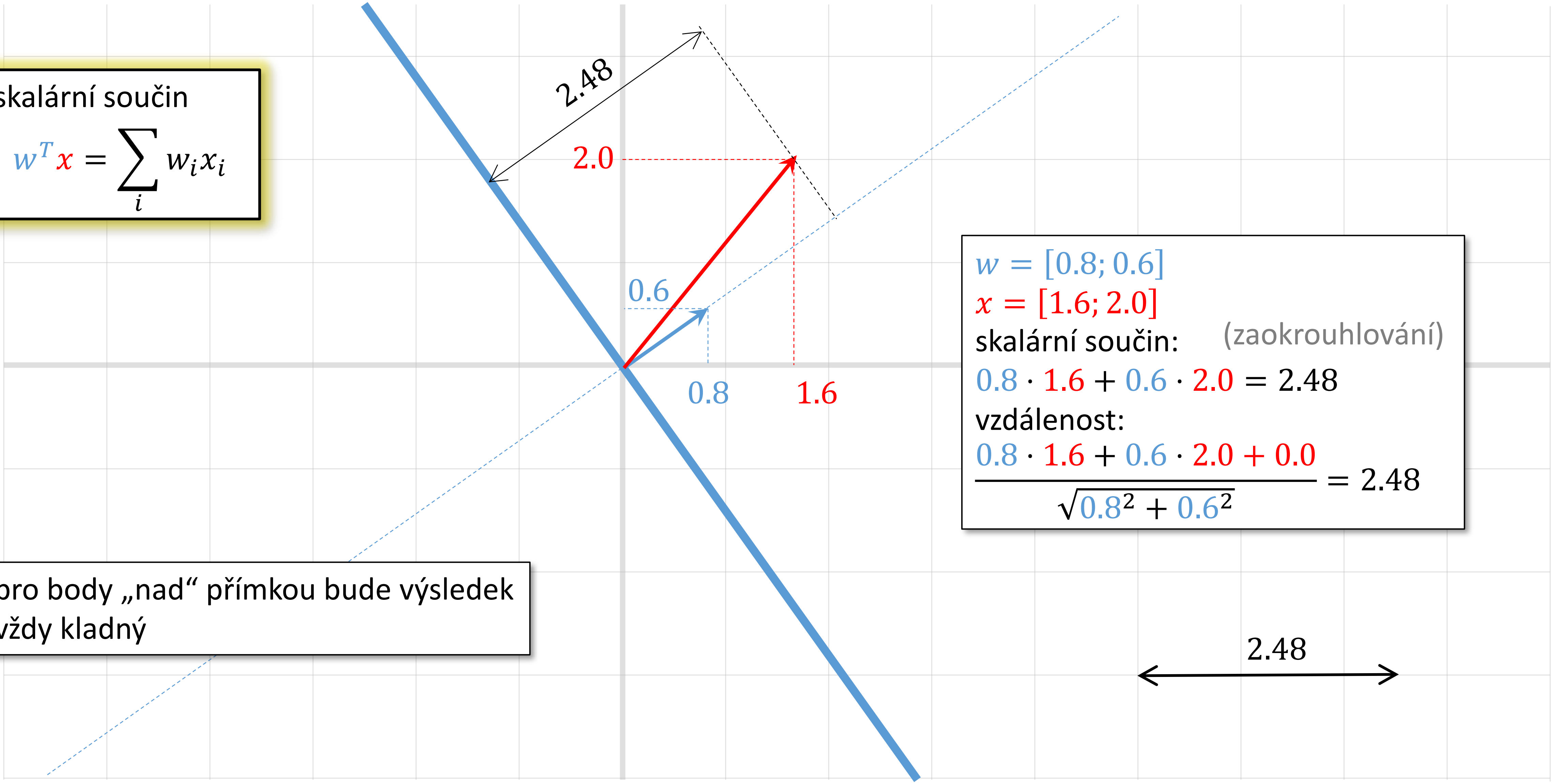
$$\frac{0.8 \cdot 1.6 + 0.6 \cdot 2.0 + 0.0}{\sqrt{0.8^2 + 0.6^2}} = 2.48$$



Geometrická interpretace

skalární součin

$$w^T x = \sum_i w_i x_i$$



$w = [0.8; 0.6]$
$x = [1.6; 2.0]$
skalární součin: (zaokrouhlování)

$$0.8 \cdot 1.6 + 0.6 \cdot 2.0 = 2.48$$

vzdálenost:

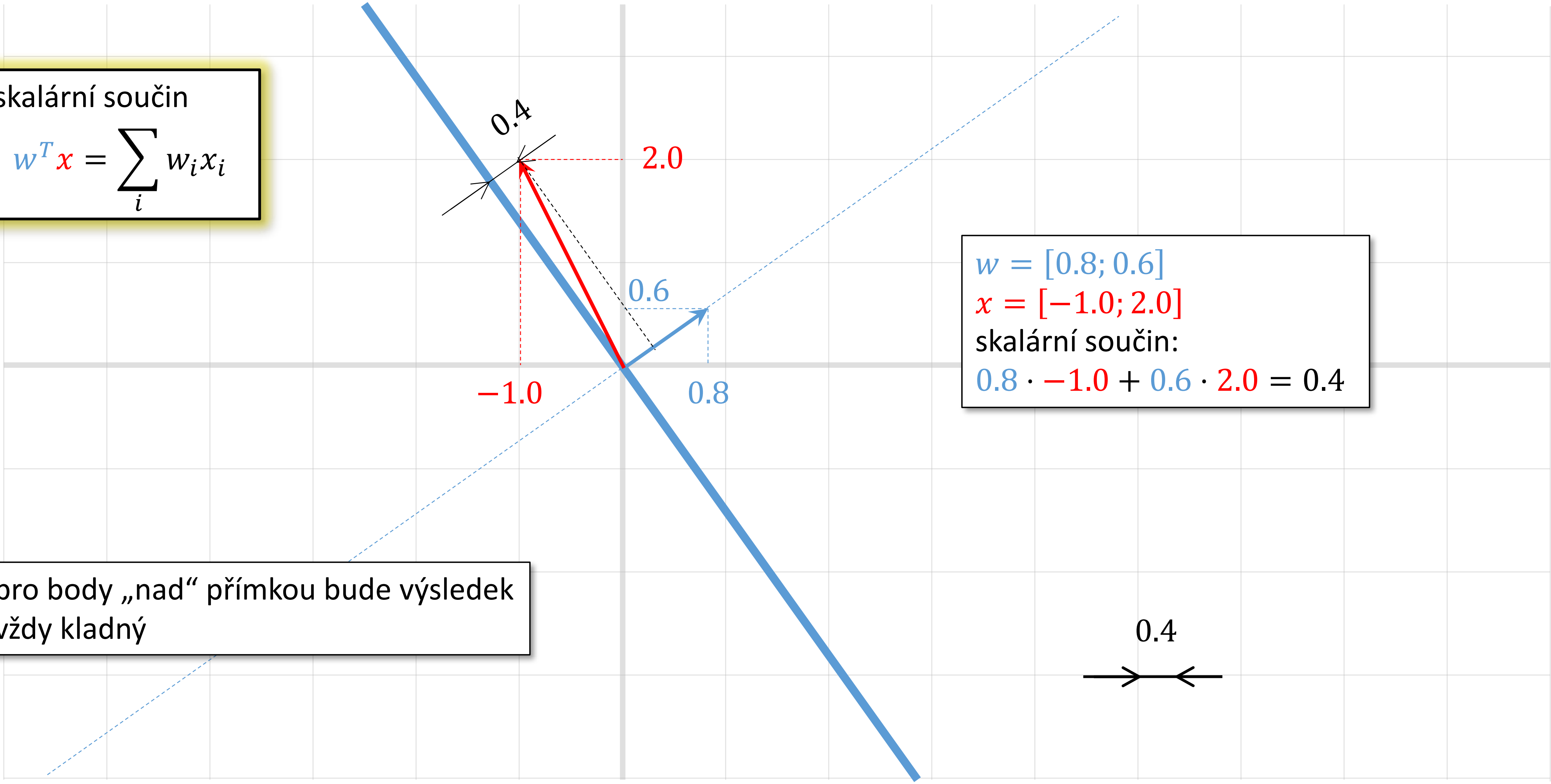
$$\frac{0.8 \cdot 1.6 + 0.6 \cdot 2.0 + 0.0}{\sqrt{0.8^2 + 0.6^2}} = 2.48$$

pro body „nad“ přímkou bude výsledek vždy kladný

Geometrická interpretace

skalární součin

$$w^T x = \sum_i w_i x_i$$



$w = [0.8; 0.6]$

$x = [-1.0; 2.0]$

skalární součin:

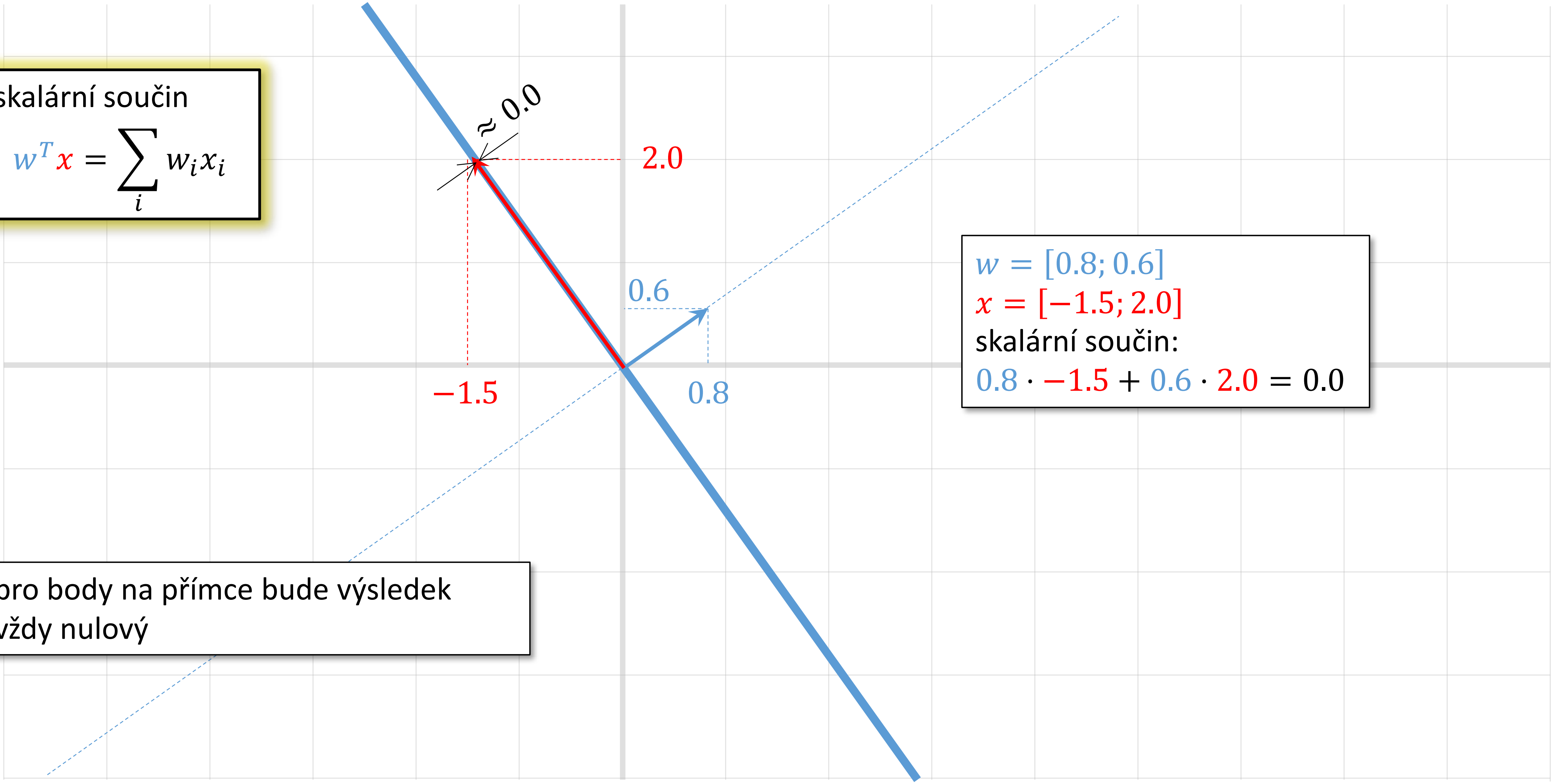
$0.8 \cdot -1.0 + 0.6 \cdot 2.0 = 0.4$

pro body „nad“ přímkou bude výsledek vždy kladný

Geometrická interpretace

skalární součin

$$w^T x = \sum_i w_i x_i$$



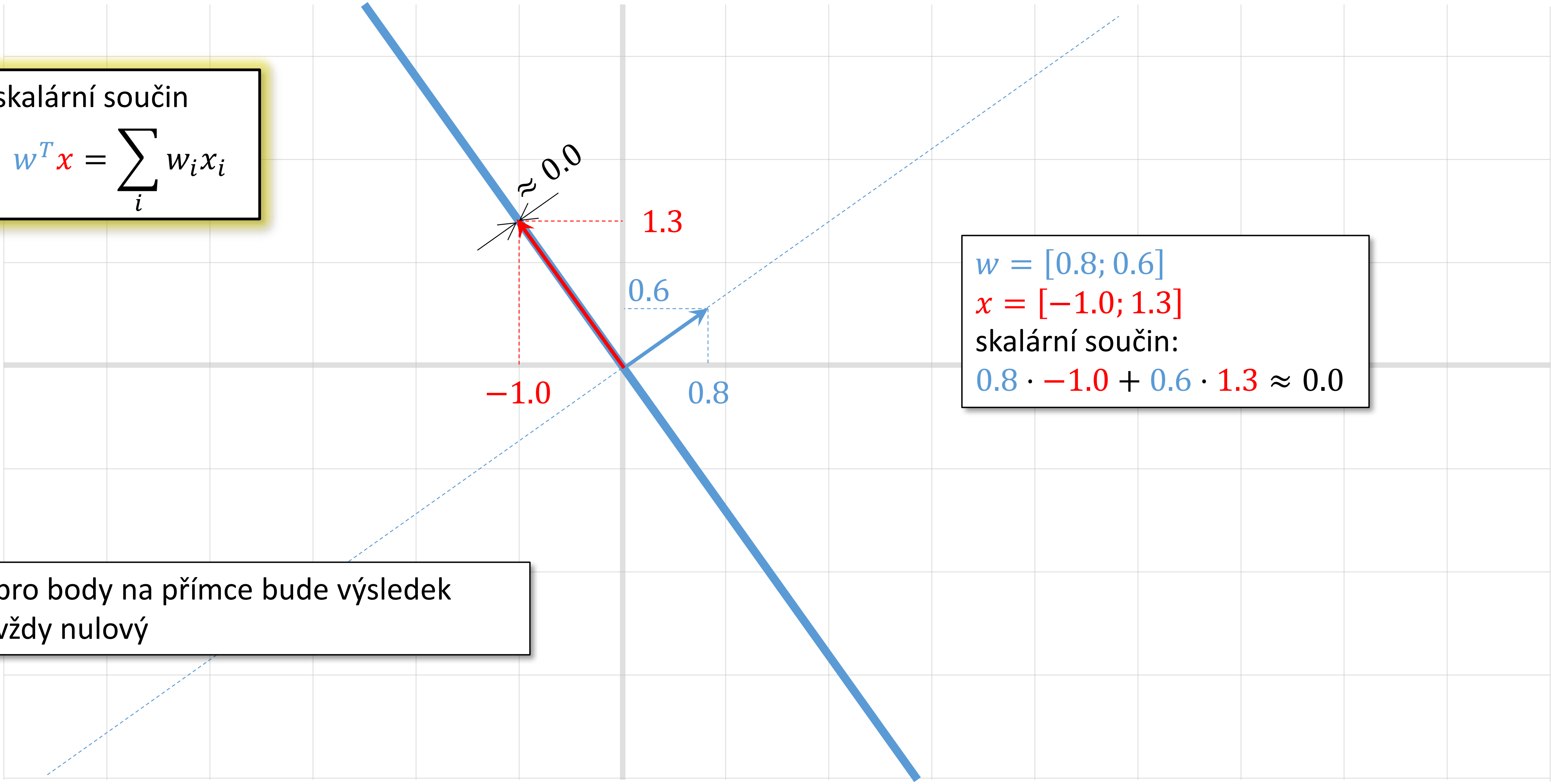
$w = [0.8; 0.6]$
$x = [-1.5; 2.0]$
skalární součin:
$0.8 \cdot -1.5 + 0.6 \cdot 2.0 = 0.0$

pro body na přímce bude výsledek vždy nulový

Geometrická interpretace

skalární součin

$$w^T x = \sum_i w_i x_i$$



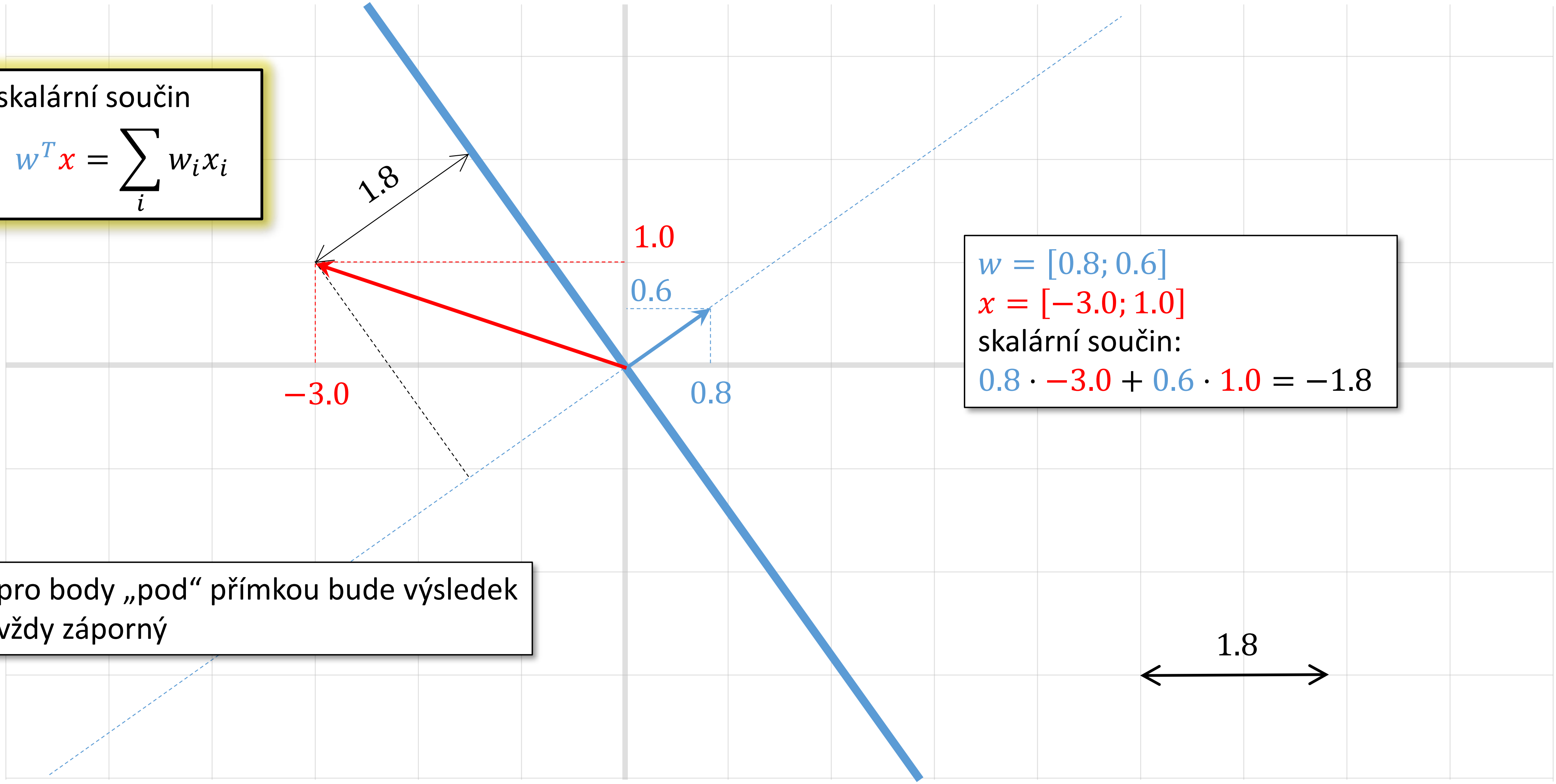
$w = [0.8; 0.6]$
$x = [-1.0; 1.3]$
skalární součin:
$0.8 \cdot -1.0 + 0.6 \cdot 1.3 \approx 0.0$

pro body na přímce bude výsledek vždy nulový

Geometrická interpretace

skalární součin

$$w^T x = \sum_i w_i x_i$$



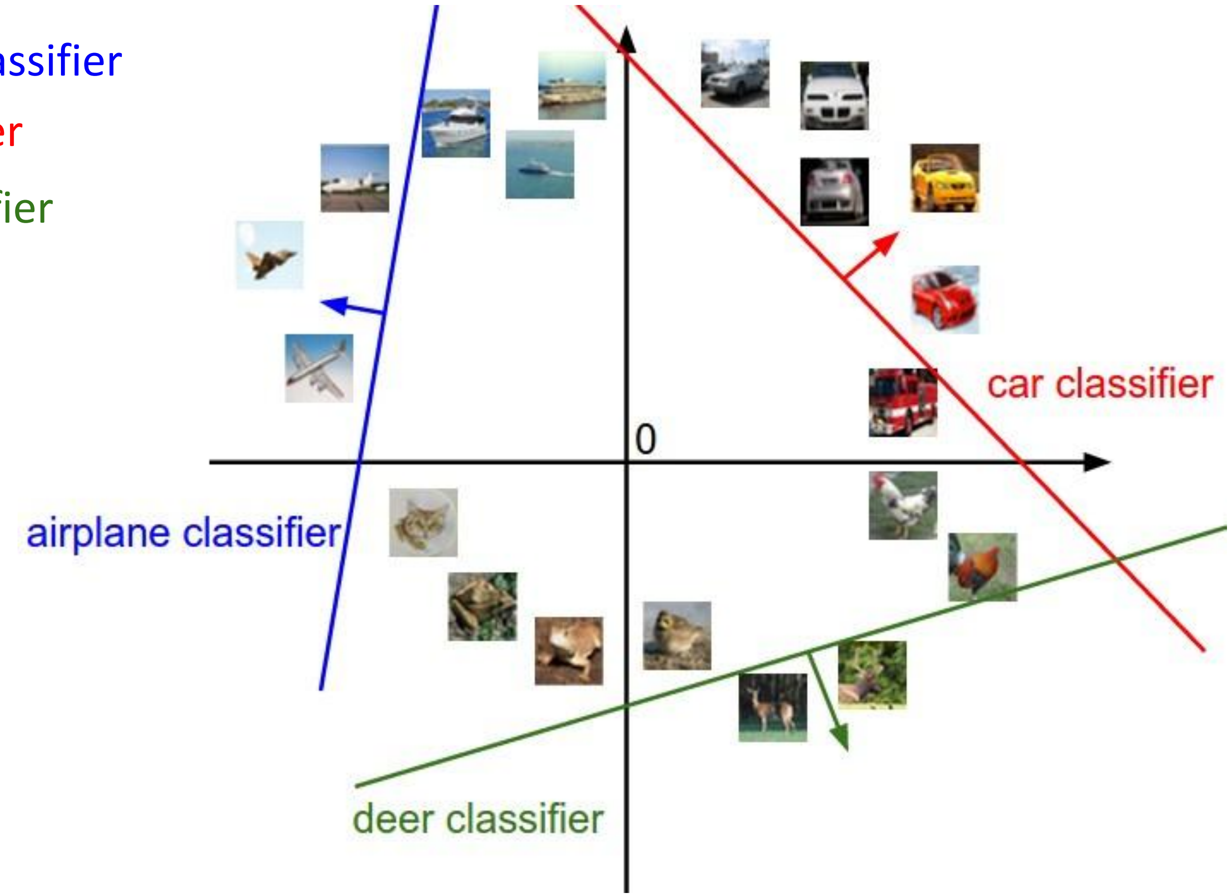
$w = [0.8; 0.6]$
$x = [-3.0; 1.0]$
skalární součin:
$0.8 \cdot -3.0 + 0.6 \cdot 1.0 = -1.8$

pro body „pod“ přímkou bude výsledek vždy záporný

Geometrická interpretace

$$w = \begin{bmatrix} w_{1,1} & w_{1,2} & \cdots & w_{1,D} \\ w_{2,1} & w_{2,2} & \cdots & w_{2,D} \\ w_{3,1} & w_{3,2} & \cdots & w_{3,D} \end{bmatrix} \begin{array}{l} \text{airplane classifier} \\ \text{car classifier} \\ \text{deer classifier} \end{array}$$

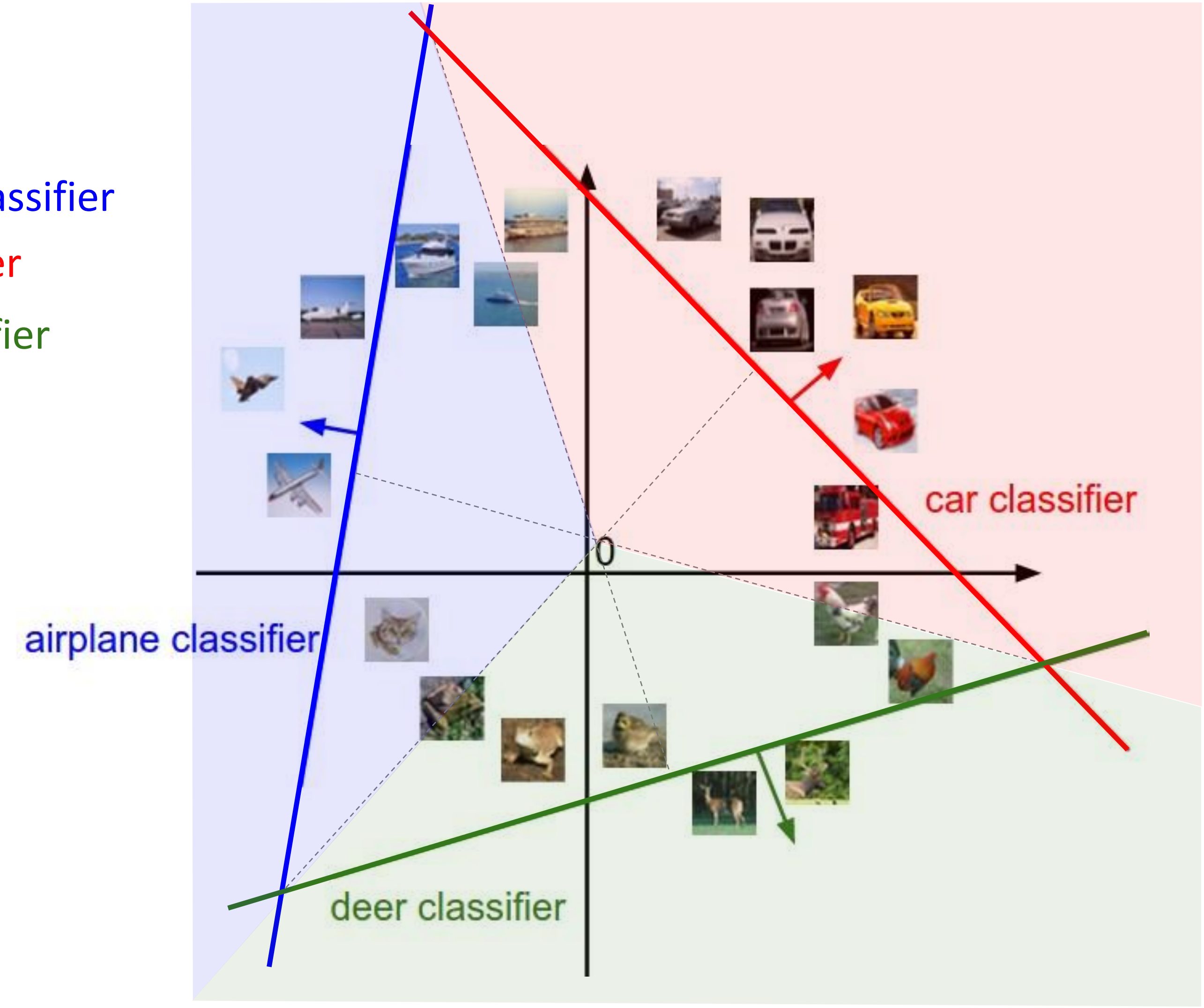
Každý řádek matice w je binární klasifikátor diskriminující třídu k od ostatních



obrázek: https://cs231n.github.io/linear-classify/

Geometrická interpretace

$$\boldsymbol{w} = \begin{bmatrix} w_{1,1} & w_{1,2} & \cdots & w_{1,D} \\ w_{2,1} & w_{2,2} & \cdots & w_{2,D} \\ w_{3,1} & w_{3,2} & \cdots & w_{3,D} \end{bmatrix} \begin{array}{l} \text{airplane classifier} \\ \text{car classifier} \\ \text{deer classifier} \end{array}$$



Každý řádek matice $\boldsymbol{w}$ je binární klasifikátor diskriminující třídu k od ostatních

obrázek: https://cs231n.github.io/linear-classify/

Vizuální interpretace: párování vzorů (template matching)

0.2	-0.5	0.1	2.0
1.5	1.3	2.1	0.0
0.0	0.25	0.2	-0.3

$\boldsymbol{w}: 3 \times 4$

0.2	-0.5
0.1	2.0

$\boldsymbol{w}_{1,:}: 2 \times 2$

1.5	1.3
2.1	0.0

$\boldsymbol{w}_{2,:}: 2 \times 2$

0.0	0.25
0.2	-0.3

$\boldsymbol{w}_{3,:}: 2 \times 2$

Jednotlivé řádky (klasifikátory) v matici uspořádáme jako obrázky (“reshape”) a vykreslíme

Vizuální interpretace: párování vzorů (template matching)

- Natrénované váhy obvykle reprezentují typický vzhled jednotlivých tříd
- Počítáme totiž lineární skóre tvaru $\boldsymbol{x} \cdot \boldsymbol{w} + \boldsymbol{b}$ a toto skóre, jak dále uvidíme, chceme maximalizovat (pro správnou třídu)
- Kdy bude skóre maximální?

$$\boldsymbol{w}^* = \underset{\boldsymbol{w}}{\operatorname{argmax}}\, \boldsymbol{x} \cdot \boldsymbol{w} + b$$ # bias zanedbáme a přepíšeme skalární součin

$$= \underset{\boldsymbol{w}}{\operatorname{argmax}} \|\boldsymbol{x}\| \cdot \|\boldsymbol{w}\| \cdot \cos\phi(\boldsymbol{x}, \boldsymbol{w})$$ # $\|\boldsymbol{x}\|$ je konst., $\|\boldsymbol{w}\|$ ovlivňuje pouze škálu, nikoliv podobu

$$= \underset{\boldsymbol{w}}{\operatorname{argmax}} \cos\phi(\boldsymbol{x}, \boldsymbol{w})$$ # kosinus má maximum v nule: $\cos(0) = 1$

$$= \alpha \cdot \boldsymbol{x}$$ # úhel $\phi(\boldsymbol{x}, \boldsymbol{w})$ mezi $\boldsymbol{x}$ a $\boldsymbol{w}$ bude nula právě když $\boldsymbol{w} = \alpha \cdot \boldsymbol{x}$

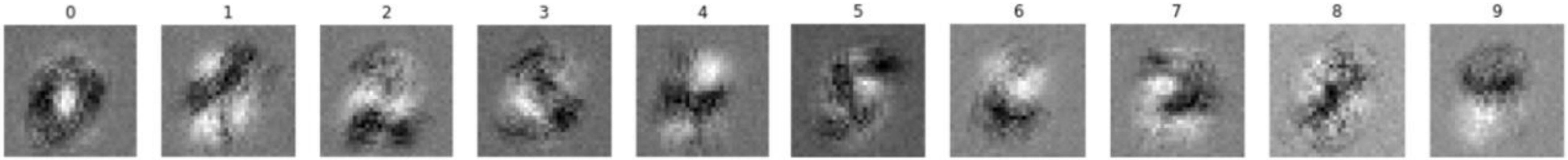
- Skóre lineárního klasifikátoru pro každou třídu maximalizujeme, pokud jsou váhy přímo úměrné obvyklému vstupu

Vizuální interpretace: párování vzorů (template matching)

Dataset CIFAR-10

obrázek: https://cs231n.github.io/linear-classify/

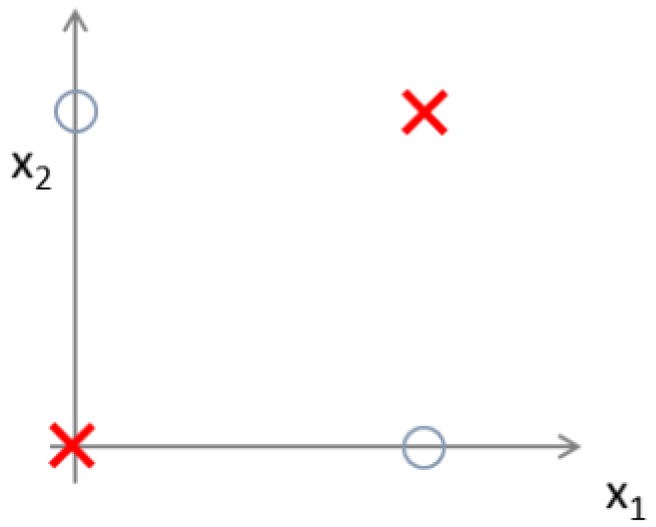
Dataset MNIST (invertované: černá = vysoká hodnota, bílá = nízká hodnota)

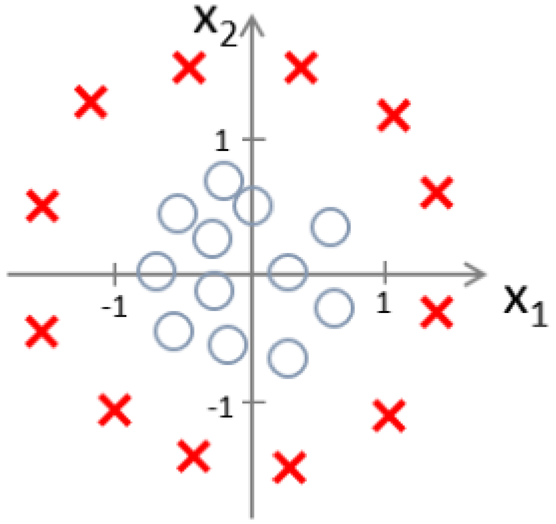


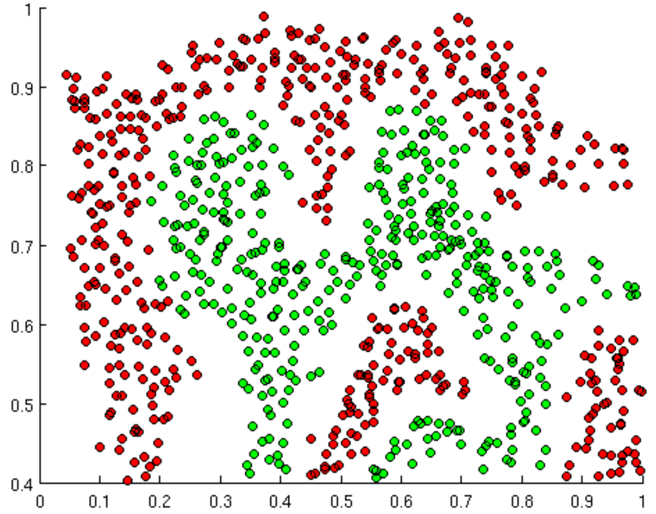
obrázek: USU, přednáška 7

Lineárně neseparovatelné případy

XOR







obrázky: USU, přednáška 9

obrázek: http://openclassroom.stanford.edu/...

Návrh a trénování lineárního klasifkátoru

1. Navrhneme diskriminativní klasifikační funkci s upravitelnými parametry
2. **Kvantifikujeme její úspěšnost klasifikace nějakým kritériem**
3. Nastavíme parametry klasifikátoru tak, abychom optimalizovali zvolené kritérium

Klasifikační kritérium

0-1 loss, křížová entropie + softmax

Přesnost klasifikace (accuracy)

- Klasifikátor predikuje číslo (index třídy) $\hat{y}_n \in \{1, \dots, K\}$

$$\hat{y}_n = \operatorname*{argmax}_k \boldsymbol{s}_n$$

- Pro každý obrázek $\boldsymbol{x}_n$ přitom známe správnou odpověď $y_n \in \{1, \dots, K\}$ (target)

- Celkem máme dataset N obrázků a tedy i párů $(\boldsymbol{x}_n, y_n)$

- Porovnáním predikcí a targetů můžeme spočítat, jak **dobře** klasifikátor klasifikuje

$$a_n = \mathbb{1}(\hat{y}_n = y_n)$$

Pro jeden obrázek

$$a = \frac{1}{N} \sum_{n=1}^{N} a_n$$

Pro celý dataset

- Čím větší přesnost a, tím lépe

Nepřesnost klasifikace (misclassification rate)

- Ekvivalentně můžeme spočítat, jak **špatně** klasifikátor klasifikuje

$$m_n = \mathbb{1}(\hat{y}_n \neq y_n)$$ ← Pro jeden obrázek

$$m = \frac{1}{N}\sum_{n=1}^{N} m_n$$ ← Pro celý dataset

- Čím **menší** nepřesnost m, tím lépe
- Proč? Protože budeme problém formulovat jako optimalizaci funkce a konvencí v literatuře je hledání minima, nikoliv maxima
- Nepřesnost (misclassification rate) se ve strojovém učení nazývá **0-1 loss**
- Jedná se o konkrétní příklad obecného kritéria (lossu), které kvantifikuje **chybu modelu**

Klasifikační kritérium obecně

- Pro každý pár $(\boldsymbol{x}_n, y_n)$ v trénovacím datasetu $\boldsymbol{X}$ spočteme

$$l_n = L_n(\boldsymbol{x}_n, y_n, \boldsymbol{w}, \boldsymbol{b})$$

kde $L_n(\boldsymbol{x}_n, y_n, \boldsymbol{w}, \boldsymbol{b})$ může být např. 0-1 loss

$$L_n(\boldsymbol{x}_n, y_n, \boldsymbol{w}, \boldsymbol{b}) = \left[\!\left[\operatorname*{argmax}_k \boldsymbol{w} \cdot \boldsymbol{x}_n + \boldsymbol{b} \neq y_n\right]\!\right]$$

Iverson bracket notace

$$[\![p]\!] = \begin{cases} 1 & p \text{ je pravdivá} \\ 0 & \text{jinak} \end{cases}$$

- A zprůměrujeme přes celý dataset

$$L(\boldsymbol{X}, \boldsymbol{w}, b) = \frac{1}{N} \sum_{n=1}^{N} L_n(\boldsymbol{x}_n, y_n, \boldsymbol{w}, b)$$

- Výsledná hodnota $l = L(\boldsymbol{X}, \boldsymbol{\theta})$ nám říká, jak špatné parametry $\boldsymbol{\theta} = \{\boldsymbol{w}, b\}$ jsou na datasetu $\boldsymbol{X} = \{\boldsymbol{x}_n, y_n | n = 1, \dots, N\}$

Význam skóre

- Který klasifikátor je lepší?

Klasifikátor A

$\boldsymbol{x}_n$ → $\boldsymbol{s}_n$

$\boldsymbol{s}_n$	
38.3	kočka
-2.1	pes
7.4	alpaka

Klasifikátor B

$\boldsymbol{x}_n$ → $\boldsymbol{s}_n$

$\boldsymbol{s}_n$	
8.1	kočka
6.6	pes
7.7	alpaka

- 0-1 loss velikost skóre nezohledňuje, roli hraje pouze pořadí
- 0-1 loss navíc není diferencovatelný → horší vlastnosti při použítí standardních optimalizačních algoritmů

Multiclass cross entropy

- Logistická regrese definuje „lepší“ kritérium (loss), tzv. **křížovou entropii**

$$\boxed{l_n = -\sum_{k=1}^{K} p_{n,k} \cdot \log(\hat{p}_{n,k})}$$

Pro jeden obrázek

- kde

$$\boldsymbol{p}_n = [p_{n,1}, \dots, p_{n,K}]^\top$$ ... cílové rozdělení (ground truth / target)

$$\hat{\boldsymbol{p}}_n = [\hat{p}_{n,1}, \dots, \hat{p}_{n,K}]^\top$$ ... výstupní pravd. (predikce) klasifikátoru

jsou vektory, na které nahlížíme jako na diskrétní pravděpodobnostní rozdělení

→ cross entropy = rozdíl mezi dvěma pravděpodobnostními rozděleními

Multiclass cross entropy

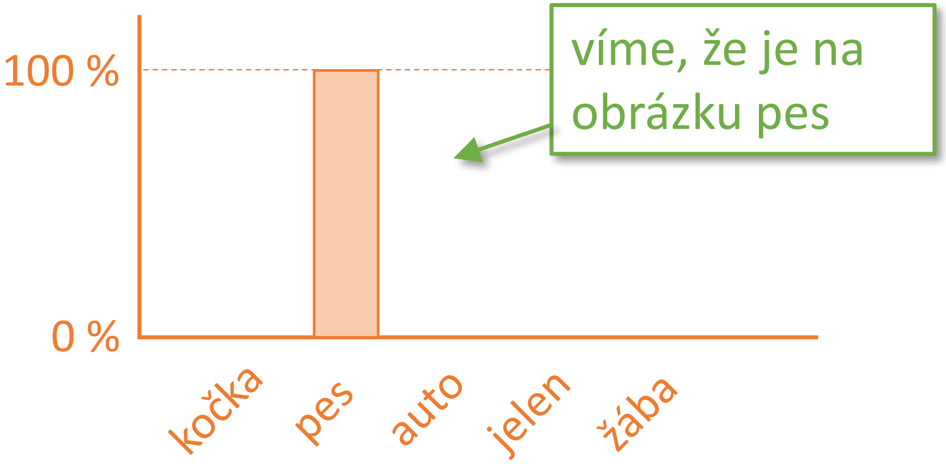
$$l_n = -\sum_{k=1}^{K} p_{n,k} \cdot \log\left(\hat{p}_{n,k}\right)$$

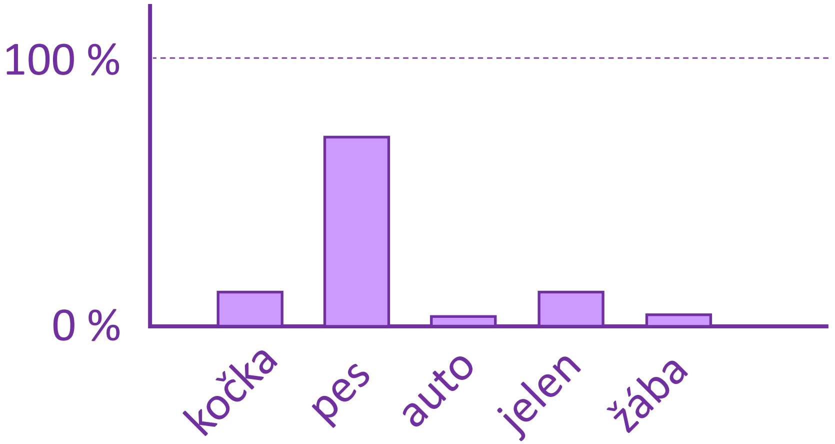
$\boldsymbol{p}_n = \left[p_{n,1}, \dots, p_{n,K}\right]^\top$

cílové rozdělení (ground truth / target)

$\hat{\boldsymbol{p}}_n = \left[\hat{p}_{n,1}, \dots, \hat{p}_{n,K}\right]^\top$

výstup (predikce) klasifikátoru



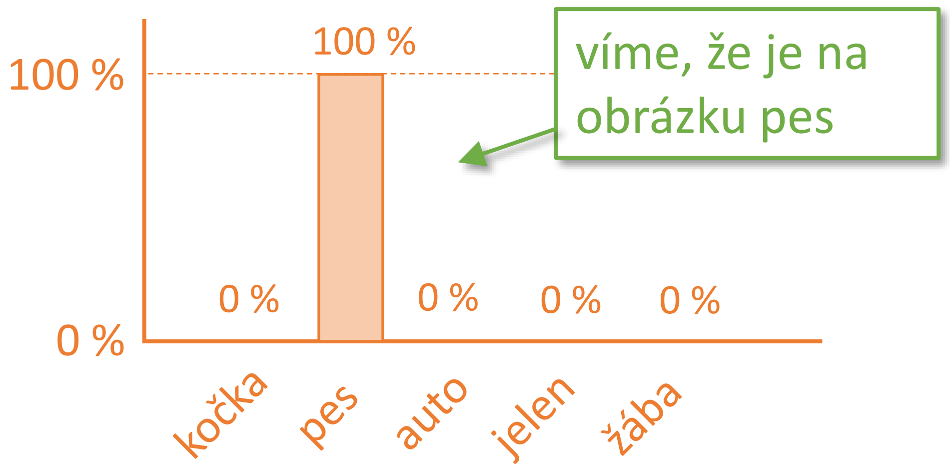


Multiclass cross entropy

$$l_n = -0 \cdot \log 0.12 - 1 \cdot \log 0.70 - 0 \cdot \log 0.02 - 0 \cdot \log 0.12 - 0 \cdot \log 0.04$$
$$= 0.357$$

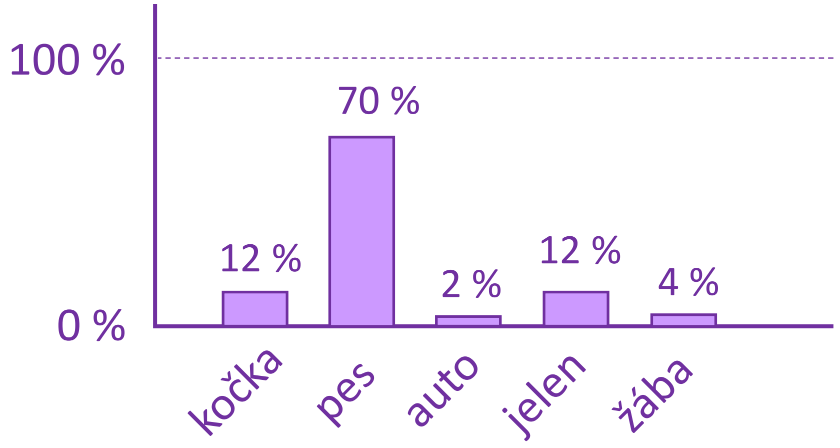
$\boldsymbol{p}_n = [p_{n,1}, \dots, p_{n,K}]^{\mathsf{T}}$

cílové rozdělení (ground truth / target)



$\hat{\boldsymbol{p}}_n = [\hat{p}_{n,1}, \dots, \hat{p}_{n,K}]^{\mathsf{T}}$

výstup (predikce) klasifikátoru



Multiclass cross entropy

$$l_n = -0 \cdot \log 0.19 - 1 \cdot \log 0.23 - 0 \cdot \log 0.18 - 0 \cdot \log 0.20 - 0 \cdot \log 0.20$$
$$= 1.470$$

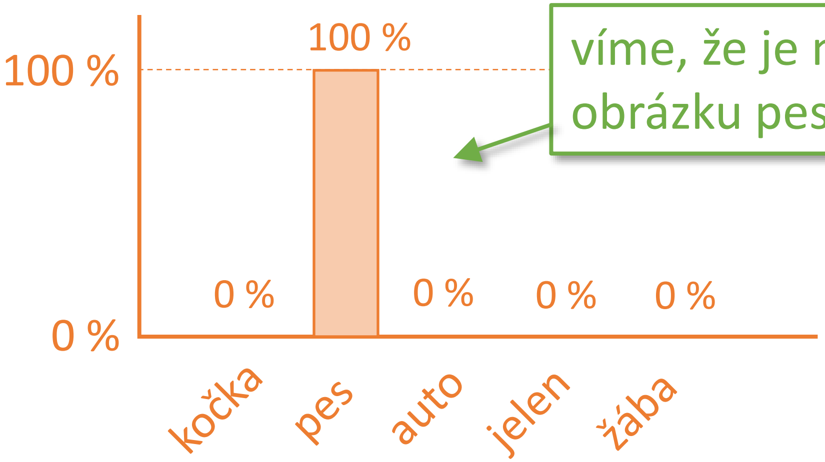
loss je vyšší

$$\boldsymbol{p}_n = [p_{n,1}, \dots, p_{n,K}]^\mathsf{T}$$

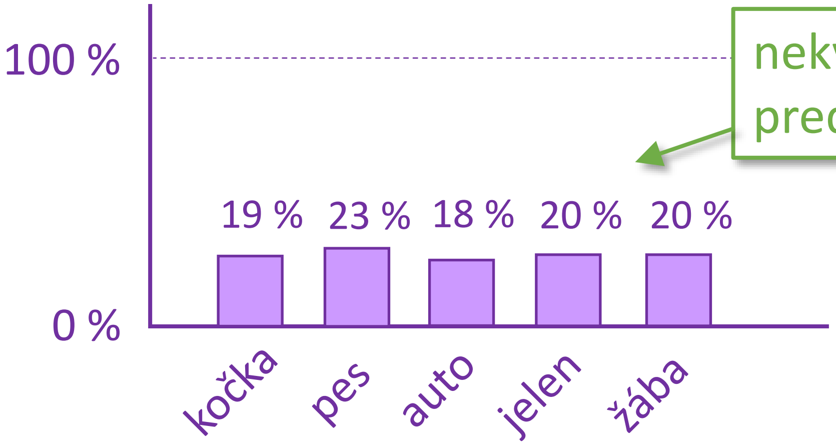
cílové rozdělení (ground truth / target)

$$\hat{\boldsymbol{p}}_n = [\hat{p}_{n,1}, \dots, \hat{p}_{n,K}]^\mathsf{T}$$

výstup (predikce) klasifikátoru



víme, že je na obrázku pes



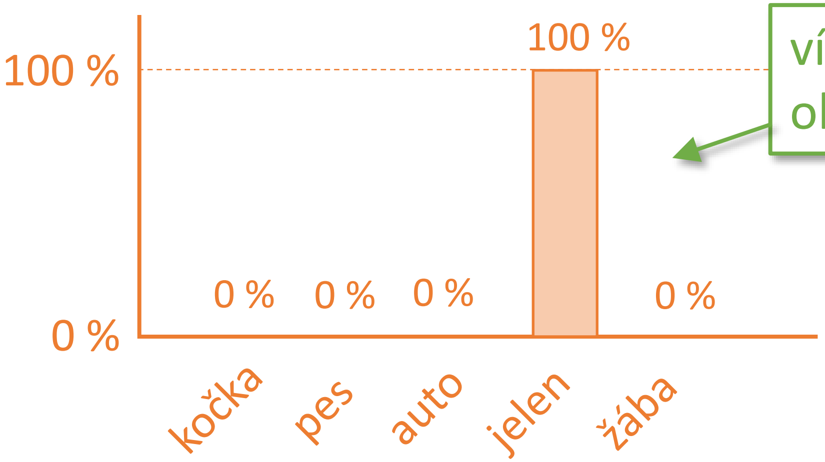
nekvalitní predikce

Multiclass cross entropy

$$l_n = -0 \cdot \log 0.20 - 0 \cdot \log 0.20 - 0 \cdot \log 0.05 - 1 \cdot \log 0.50 - 0 \cdot \log 0.05$$
$$= 0.693$$

$\boldsymbol{p}_n = [p_{n,1}, \dots, p_{n,K}]^{\mathsf{T}}$

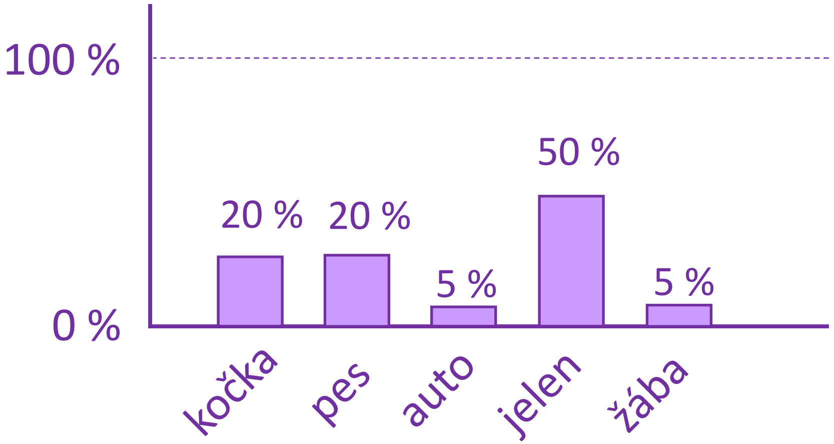
cílové rozdělení (ground truth / target)



víme, že je na obrázku jelen

$\hat{\boldsymbol{p}}_n = [\hat{p}_{n,1}, \dots, \hat{p}_{n,K}]^{\mathsf{T}}$

výstup (predikce) klasifikátoru



Převod číselného označení třídy na rozdělení: one hot encoding

- Značka y_n pro každý obrázek je celé číslo, tj. $y_n \in \{1, \dots, K\}$
- Pokud počet tříd $K = 5$ → požadované rozdělení je

$$y_n = 2 \quad \Rightarrow \quad \boldsymbol{p}_n = [0,1,0,0,0]^\top$$

$$y_n = 5 \quad \Rightarrow \quad \boldsymbol{p}_n = [0,0,0,0,1]^\top$$

Převod výstupních skóre modelu na rozdělení: softmax

- Normalizuje vektor skóre $\boldsymbol{s}_n$ tak, že výstup lze interpretovat jako pravděpodobnosti
- Pravděpodobnost, že na obrázku $\boldsymbol{x}_n$ je objekt třídy k definuje jako

$$\boxed{\hat{p}_{n,k} = P(\text{třída } k | \boldsymbol{x}_n) = \frac{e^{s_{n,k}}}{\sum_{i=1}^{K} e^{s_{n,i}}}}$$

- Výstupem K-dimezionální vektor $\hat{\boldsymbol{p}}_n$ pravděpodobností jednotlivých tříd

$$\hat{\boldsymbol{p}}_n = \left[\hat{p}_{n,1}, \dots, \hat{p}_{n,K}\right]^\top, \qquad 0 \leq \hat{p}_{n,k} \leq 1, \qquad \sum_{k=1}^{K} \hat{p}_{n,k} = 1$$

- Chová se jako “měkké” maximum: exponenciováním se zvýrazní rozdíly (nejvyšší hodnota vynikne), až teprve pak se normalizuje (ostatní jsou staženy k nule)

Softmax: příklad

$$\hat{p}_{n,k} = \frac{e^{s_{n,k}}}{\sum_{i=1}^{K} e^{s_{n,i}}}$$

	$\boldsymbol{s}_n$	$\xrightarrow{\exp(\boldsymbol{s}_n)}$	$\boldsymbol{u}_n$	$\xrightarrow{\frac{\boldsymbol{u}_n}{\sum_{k=1}^{K} u_{n,k}}}$	$\hat{\boldsymbol{p}}_n$	
kočka skóre	3.83		46.1		0.94	kočka pravděpodobnost
pes skóre	-0.21		0.81		0.02	pes pravděpodobnost
alpaka skóre	0.74		2.		0.04	alpaka pravděpodobnost

Softmax + cross entropy

- V cross entropy *pro klasifikaci* bude aktivní vždy pouze jeden člen sumy (když $k = y_n$):

$$l_n = -\sum_{k=1}^{K} p_{n,k} \log \hat{p}_{n,k} = -\log \hat{p}_{n,y_n} = -\log \frac{e^{s_{n,y_n}}}{\sum_{k=1}^{K} e^{s_{n,k}}}$$

což je zápis, jenž najdeme např. v [poznámkách cs231n](http://cs231n.github.io/linear-classify/)

- Pokud rozepíšeme logaritmus zlomku, dostaneme druhou častou variantu zápisu

$$l_n = -\log \frac{e^{s_{n,y_n}}}{\sum_{k=1}^{K} e^{s_{n,k}}} = -s_{n,y_n} + \log \sum_{k=1}^{K} e^{s_{n,k}}$$

- Softmax + CE tedy maximalizuje poměr pravděpodobnosti požadované třídy vůči součtu všech ostatních a to pro každý vzorek

http://cs231n.github.io/linear-classify/

Citlivost cross entropy na změnu skóre

$\boxed{1.3}$	kočka
0.8	pes
-0.2	alpaka

$\boldsymbol{x}_n$ → $\boldsymbol{s}_n$

$$
\begin{aligned}
l_n &= -\log \frac{2.718^{1.3}}{2.718^{1.3} + 2.718^{0.8} + 2.718^{-0.2}} \\
&= -\log \frac{3.67}{3.67 + 2.23 + 0.82} \\
&= -\log 0.55 \\
&= 0.6
\end{aligned}
$$

obrázek: http://cs231n.github.io/linear-classify/

Citlivost cross entropy na změnu skóre

$\boldsymbol{x}_n$

3.0	**kočka**
0.8	pes
-0.2	alpaka

$\boldsymbol{s}_n$

lineární skóre target třídy se zvýší (zlepší)

$$
\begin{aligned}
l_n &= -\log \frac{2.718^{3.0}}{2.718^{3.0} + 2.718^{0.8} + 2.718^{-0.2}} \\
&= -\log \frac{20.09}{20.09 + 2.23 + 0.82} \\
&= -\log 0.87 \\
&= 0.14
\end{aligned}
$$

softmax cross entropy se zmenší (zlepší)

obrázek: http://cs231n.github.io/linear-classify/

Citlivost cross entropy na změnu skóre

3.0	kočka
0.8	pes
-4.0	**alpaka**

$\boldsymbol{s}_n$

lineární skóre nesprávné třídy se sníží (zlepší)

$$
\begin{aligned}
l_n &= -\log \frac{e^{3.0}}{2.718^{3.0} + 2.718^{0.8} + 2.718^{-4.0}} \\
&= -\log \frac{20.09}{20.09 + 2.23 + 0.02} \\
&= -\log 0.90 \\
&= 0.11
\end{aligned}
$$

softmax cross entropy se opět zmenší (zlepší)

obrázek: http://cs231n.github.io/linear-classify/

Citlivost cross entropy na změnu skóre

3.0

kočka

- Cross entropy se sníží (zlepší):
  - zvýšením skóre správné třídy
  - snížením skóre nesprávné třídy
  - obojím zároveň
- Pokud je skóre správné třídy nejvyšší, lze dosáhnout snížení lossu pouhým vynásobením vektoru skóre číslem > 1
- Takto jednoduchému “učení” je potřeba zabránit regularizací

lineární skóre nesprávné třídy se sníží (zlepší)

$$= -\log 0.90$$
$$= 0.11$$

softmax cross entropy se opět zmenší (zlepší)

obrázek: http://cs231n.github.io/linear-classify/

L2 regularizace

Vliv přeškálování parametrů

$$\boldsymbol{s}_n = \boldsymbol{w} \cdot \boldsymbol{x}_n$$

$$\hat{\boldsymbol{p}}_n = \frac{e^{\boldsymbol{s}_n}}{\sum_{k=1}^{K} e^{s_{n,k}}}$$

Matice parametrů vynásobená 10x

$$\begin{bmatrix} 0.21 \\ 0.09 \\ -0.40 \end{bmatrix} = \begin{bmatrix} 0.016 & -0.002 & 0.003 \\ 0.003 & -0.006 & 0.006 \\ -0.004 & -0.006 & -0.008 \end{bmatrix} \begin{bmatrix} 10 \\ 20 \\ 30 \end{bmatrix}$$

$$\begin{bmatrix} 0.41 \\ 0.36 \\ 0.22 \end{bmatrix} = \text{Softmax}\left(\begin{bmatrix} 0.21 \\ 0.09 \\ -0.40 \end{bmatrix}\right)$$

$$y_n = 0$$

$$l_n = -\log(0.41) = 0.89$$

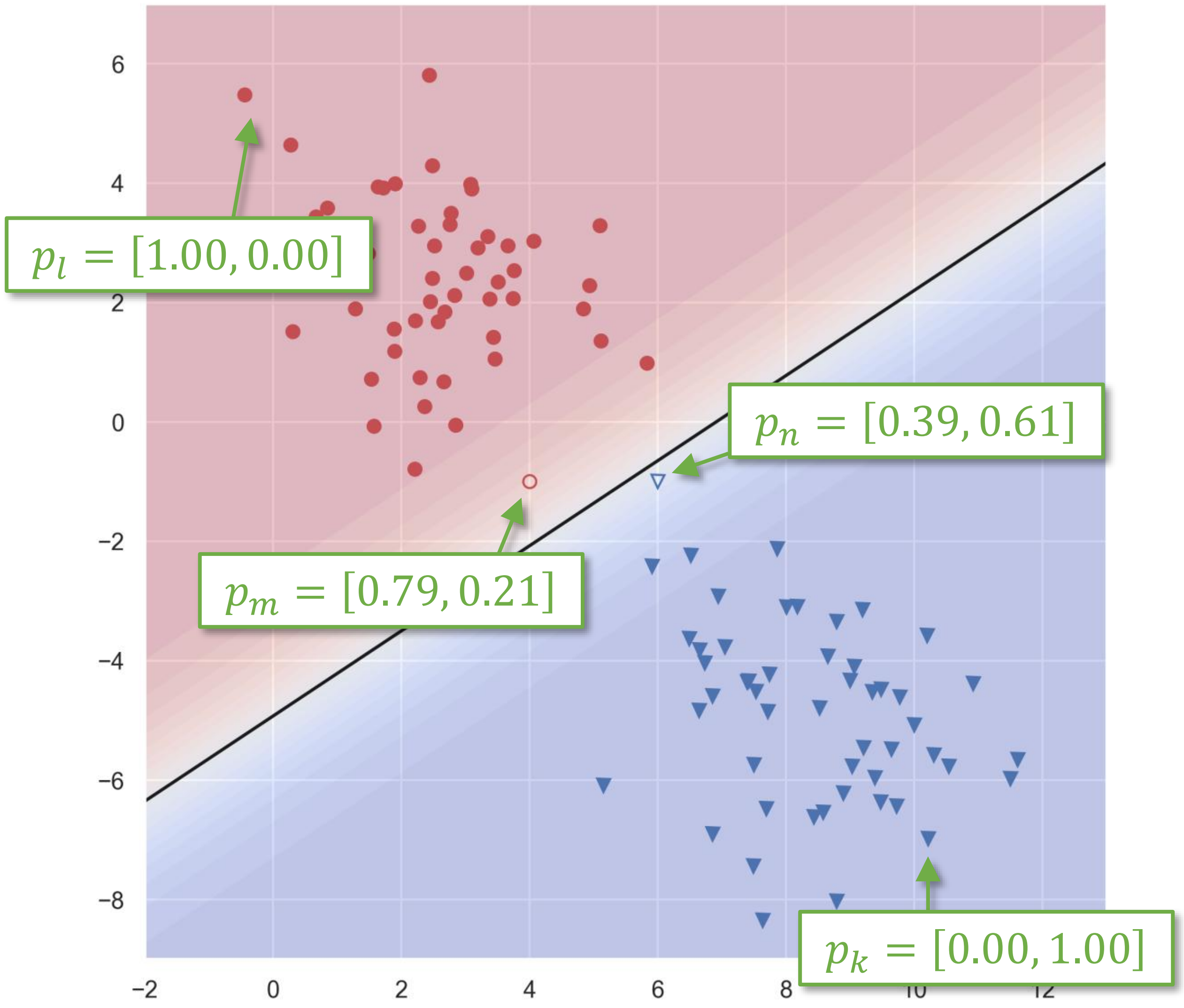
$$\begin{bmatrix} 2.05 \\ 0.90 \\ -4.00 \end{bmatrix} = \begin{bmatrix} 0.16 & -0.02 & 0.03 \\ 0.03 & -0.06 & 0.06 \\ -0.04 & -0.06 & -0.08 \end{bmatrix} \begin{bmatrix} 10 \\ 20 \\ 30 \end{bmatrix}$$

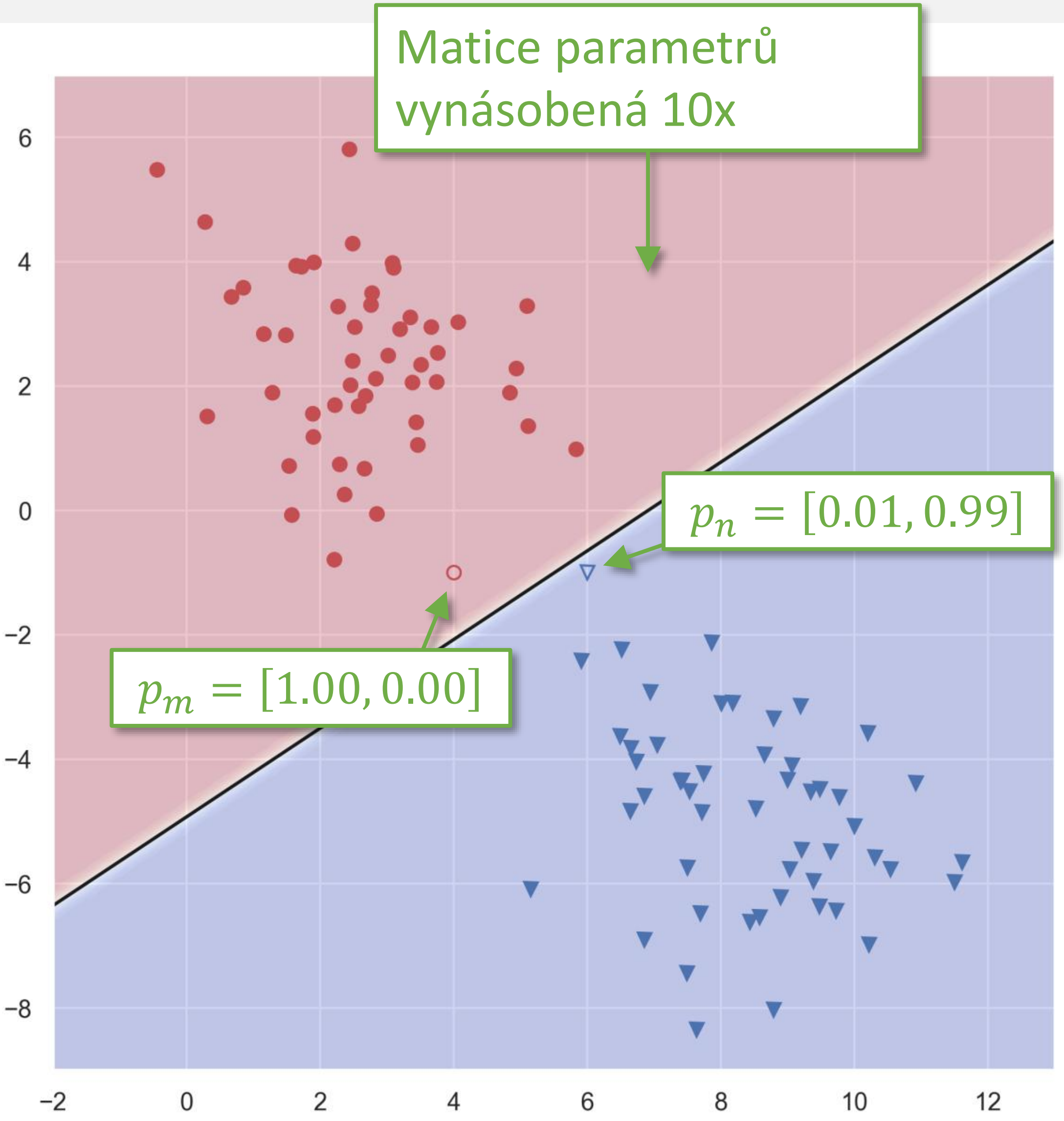
$$\begin{bmatrix} 0.76 \\ 0.24 \\ 0.00 \end{bmatrix} = \text{Softmax}\left(\begin{bmatrix} 2.05 \\ 0.90 \\ -4.00 \end{bmatrix}\right)$$

$$l_n = -\log(0.76) = 0.27$$

- Přeškálováním parametrů se zvýrazní rozdíly, ale nezmění znaménko skóre (logitů) ani pořadí pravděpodobností na výstupu, tj. klasifikátor predikuje stále stejně
- Hodnota kritéria (lossu) se ale přitom zmenší

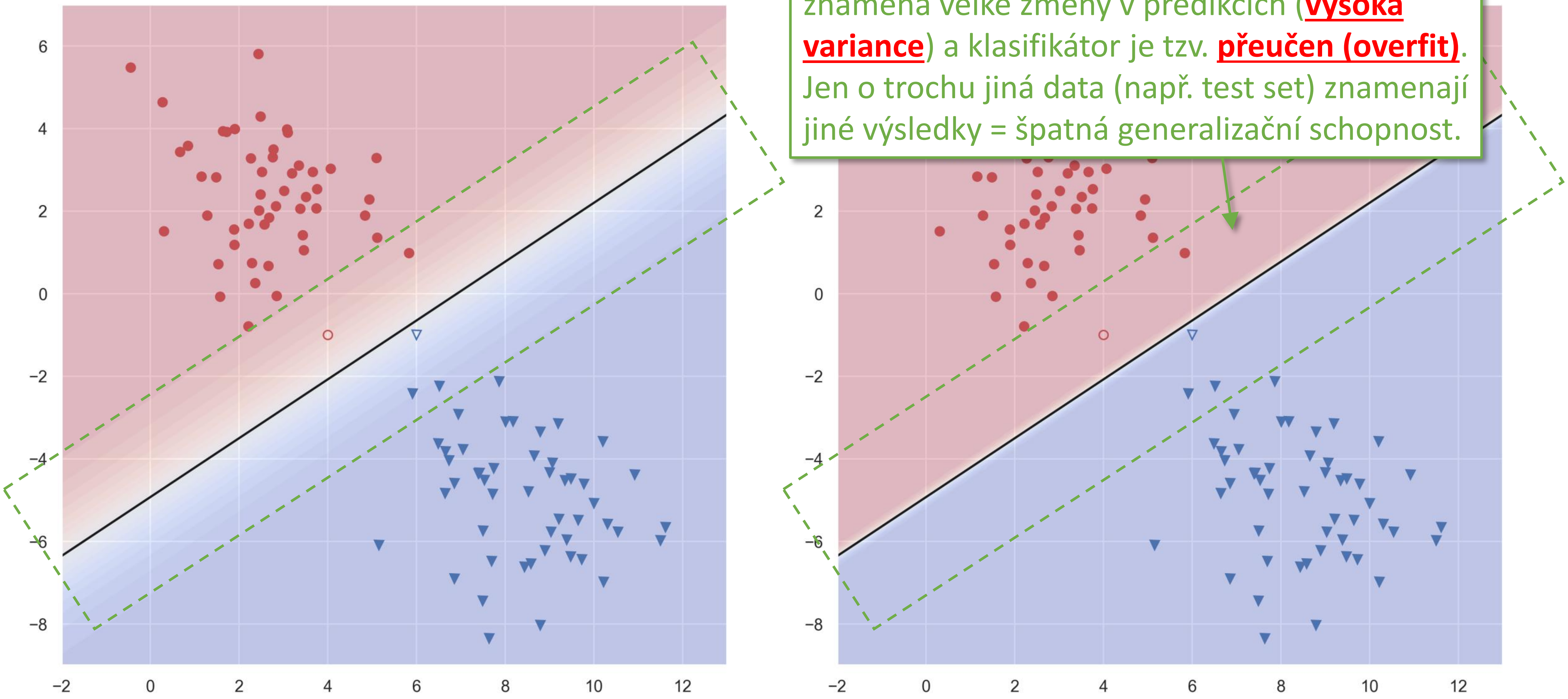
Vliv přeškálování parametrů





Vliv přeškálování parametrů

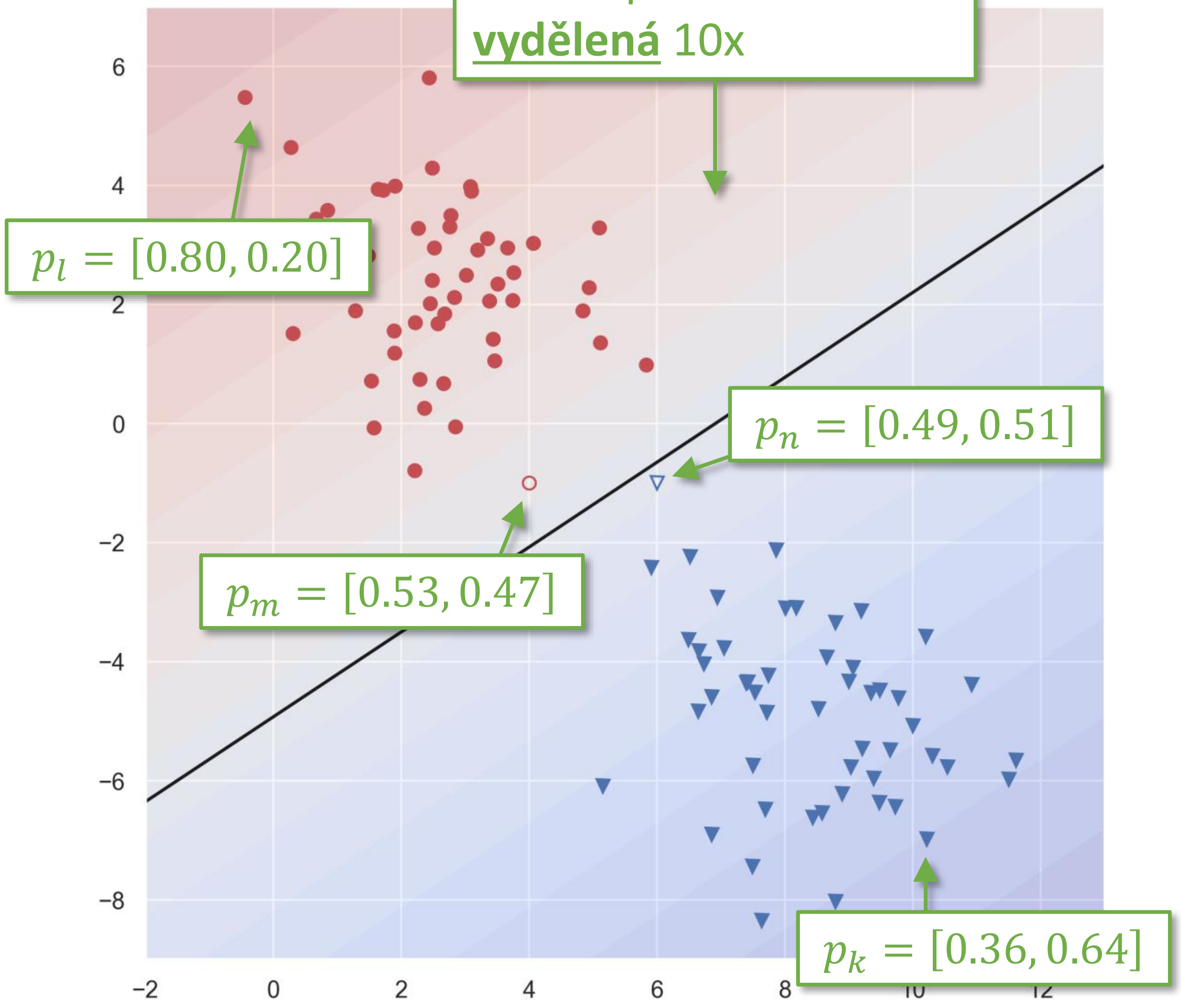
Přílišné “sebevědomí” v oblastech, kde jsou data řídká. Malá změna na vstupu potom znamená velké změny v predikcích (**vysoká variance**) a klasifikátor je tzv. **přeučen (overfit)**. Jen o trochu jiná data (např. test set) znamenají jiné výsledky = špatná generalizační schopnost.

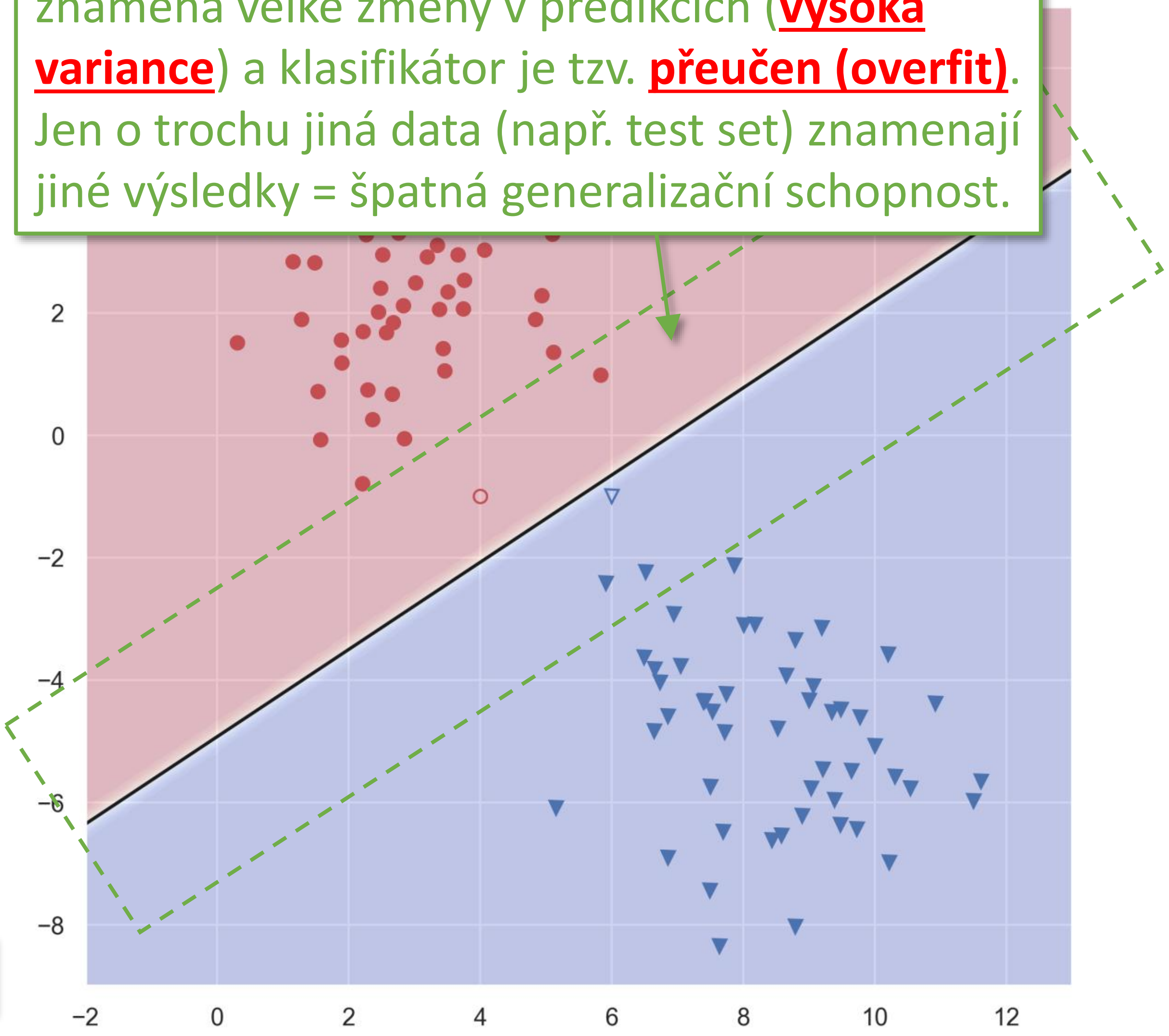


Vliv přeškálování parametrů

Přílišné “sebevědomí” v oblastech, kde jsou data řídká. Malá změna na vstupu potom znamená velké změny v predikcích (**vysoká variance**) a klasifikátor je tzv. **přeučen (overfit)**. Jen o trochu jiná data (např. test set) znamenají jiné výsledky = špatná generalizační schopnost.

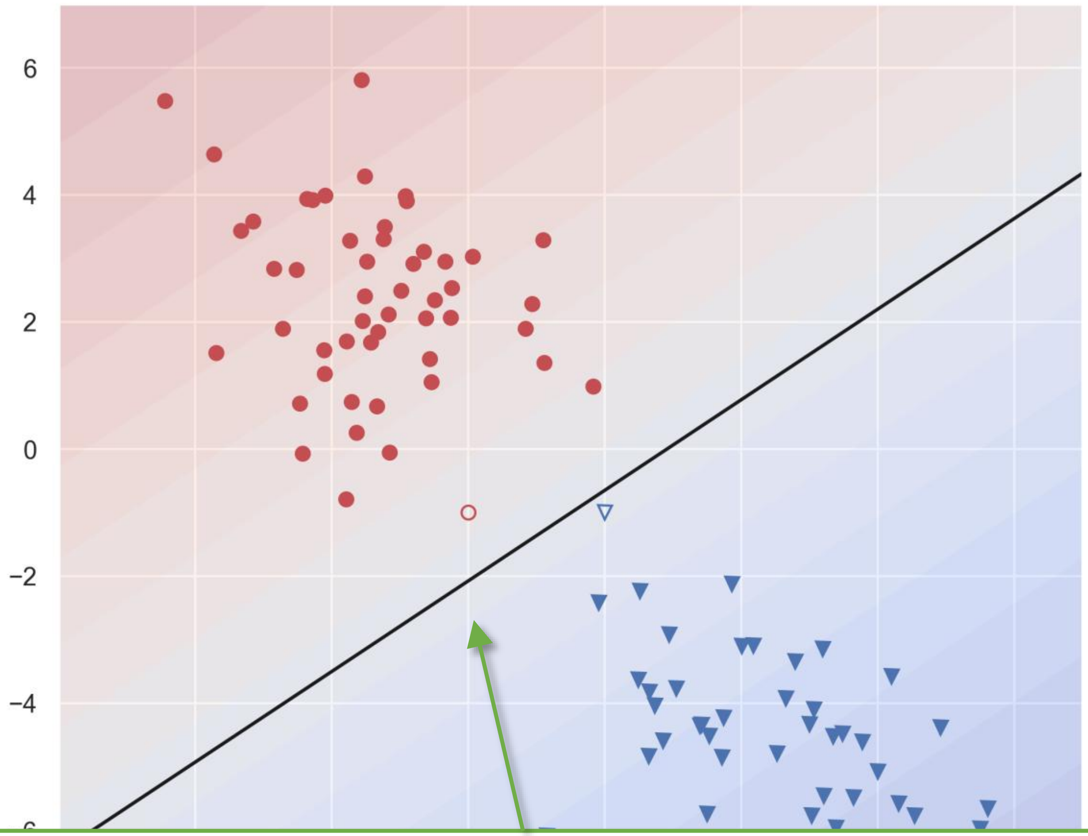
Matice parametrů **vydělená** 10x

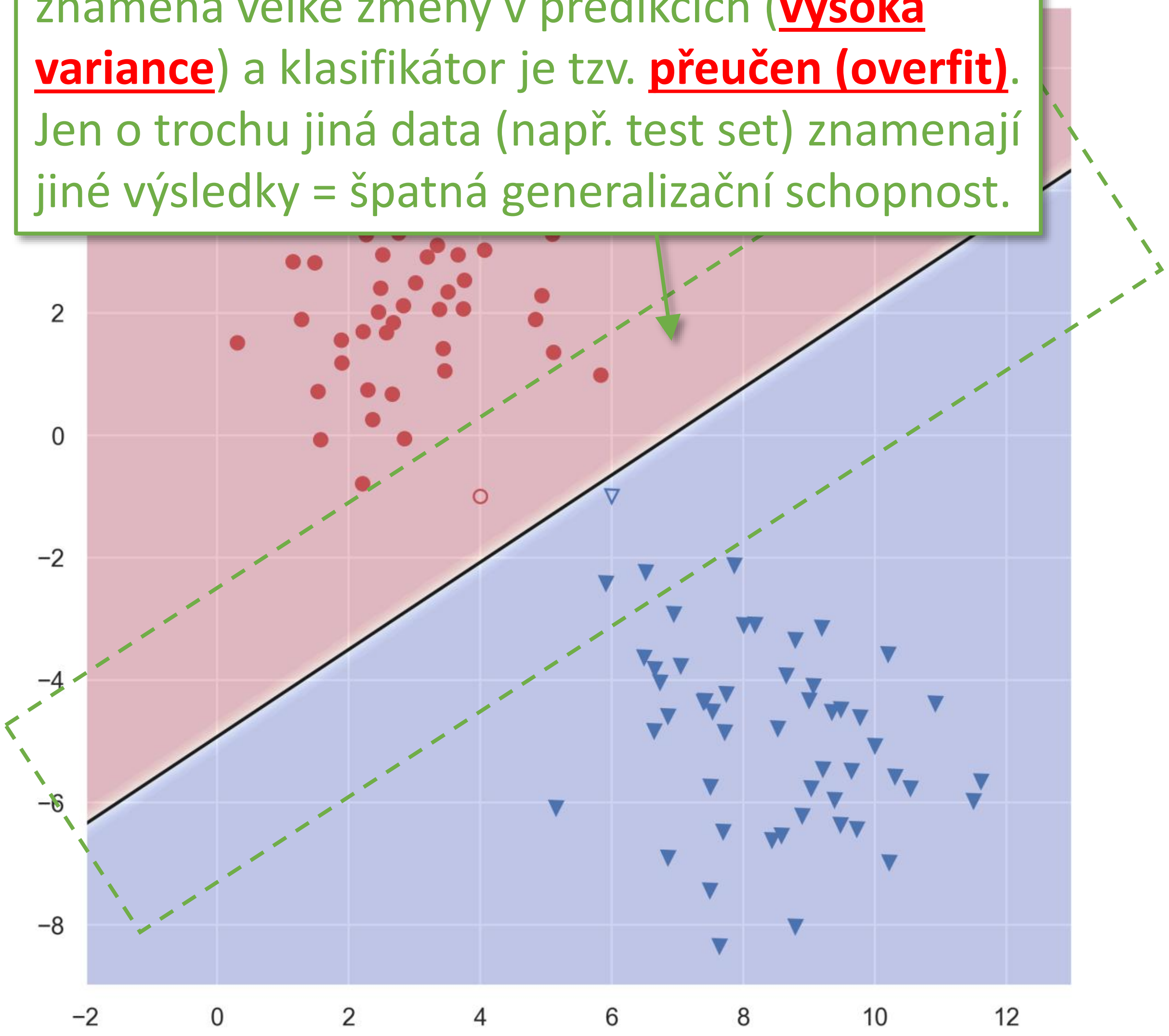




Vliv přeškálování parametrů

Přílišné “sebevědomí” v oblastech, kde jsou data řídká. Malá změna na vstupu potom znamená velké změny v predikcích (**vysoká variance**) a klasifikátor je tzv. **přeučen (overfit)**. Jen o trochu jiná data (např. test set) znamenají jiné výsledky = špatná generalizační schopnost.





Přílišná “opatrnost” i v oblastech s reprezentativními daty znamená malé změny na výstupu i při velkých rozdílech ve vstupech (**vysoký bias**). Klasifikátor je tzv. **nedoučen (underfit)** a není schopen modelovat okrajové případy.

Škálování vah aditivní regularizací

- Spočívá v penalizaci příliš vysokých vah zavedením dodatečného členu do lossu

Celkový loss na trénovacích datech $\boldsymbol{X}$ a s parametry $\boldsymbol{\theta}$ včetně regularizace

Loss vypočtený z velikosti parametrů

$$L(\boldsymbol{X}, \boldsymbol{\theta}) = \frac{1}{N} \sum_{n=1}^{N} L_n(\boldsymbol{x}_n, \boldsymbol{\theta}) + \lambda \cdot R(\boldsymbol{\theta})$$

Loss vypočtený z dat (např. softmax cross entropy)

- λ je hyperparametr ☹
- často jako weight decay*
- ovlivňuje sílu regularizace
- typicky malé číslo $\lambda \approx 10^{-3}$

Nejčastěji L2 regularizace

$$R(\boldsymbol{w}) = \sum_{k,d} w_{k,d}^2$$

Celkové kritérium včetně aditivní regularizace

- Včetně aditivní regularizace tedy budeme chybovost klasifikátoru posuzovat jako

$$\boxed{L(\boldsymbol{X}, \boldsymbol{\theta}) = L_{\text{data}}(\boldsymbol{X}, \boldsymbol{\theta}) + \lambda \cdot L_{\text{reg}}(\boldsymbol{\theta})}$$

Klasifikační kritérium

Support Vector Machine (SVM)

Weston-Watkins multiclass hinge loss

- Zkusme měřit data loss jinak:

skóre nesprávné třídy

skóre správné třídy

$$l_n = \qquad s_{n,c} - s_{n,y_n}$$

Mohli bychom chtít, aby rozdíl

skóre_nesprávné_třídy – skóre_správné_třídy

byl co nejvíce záporný

chceme malé skóre_nesprávné_třídy

chceme velké skóre_správné_třídy

Weston-Watkins multiclass hinge loss

- Zkusme měřit data loss jinak:

skóre nesprávné třídy

skóre správné třídy

$$l_n = \max(0, s_{n,c} - s_{n,y_n})$$

V rámci prevence overfitu nám ale bude stačit, když

skóre_správné_třídy > skóre_nesprávné_třídy

Pokud se tato podmínka splní, pak rozdíl $s_{n,c} - s_{n,y_n}$ uvnitř bude záporný a loss bude nula → už nebude kam dál optimalizovat. Dokud se nesplní, hodnota lossu bude přímo úměrná rozdílu.

Weston-Watkins multiclass hinge loss

skóre nesprávné třídy

skóre správné třídy

- Zkusme měřit data loss jinak:

$$l_n = \max(0, s_{n,c} - s_{n,y_n} + \Delta)$$

Pro jistotu ale budeme chtít, aby rozdíl byl alespoň o nějaký margin Δ. Budeme tedy nakonec chtít:

skóre_správné_třídy > skóre_nesprávné_třídy + Δ

Dokud se podmínka nesplní, $s_{n,c} - s_{n,y_n} + \Delta$ bude kladné a loss proto nenulový.

Weston-Watkins multiclass hinge loss

- Zkusme měřit data loss jinak:

skóre nesprávné třídy

skóre správné třídy

$$l_n = \max(0, s_{n,c} - s_{n,y_n} + 1)$$

Margin Δ se obvykle nastavuje na hodnotu 1, lze ho totiž nahradit přeškálováním skóre - a to lze “štelovat” regularizací.

Weston-Watkins multiclass hinge loss

- Zkusme měřit data loss jinak:

skóre nesprávné třídy

skóre správné třídy

$$l_n = \sum_{c \neq y_n} \max(0, s_{n,c} - s_{n,y_n} + 1)$$

Podmínka dostatečného rozdílu by měla platit pro všechny **nesprávné** třídy

Weston-Watkins multiclass hinge loss

- Zkusme měřit data loss jinak:

$$\boxed{l_n = \sum_{c \neq y_n} \max(0, s_{n,c} - s_{n,y_n} + 1)}$$

finální podoba hinge loss

Weston-Watkins multiclass hinge loss $l_n = \sum_{c \neq y_n} \max(0, s_{n,c} - s_{n,y_n} + 1)$

$\boldsymbol{s}_n$	
$\boxed{-1.1}$	kočka
2.3	pes
-6.2	alpaka

$\boldsymbol{x}_n$ $\boldsymbol{s}_n$

$\boldsymbol{s}_n$	
3.1	kočka
$\boxed{5.8}$	pes
2.6	alpaka

$\boldsymbol{x}_n$ $\boldsymbol{s}_n$

$$l_n = \sum_{c \in \{\text{pes}, \text{alpaka}\}} \max(0, s_{n,c} - s_{n,y_n} + 1)$$

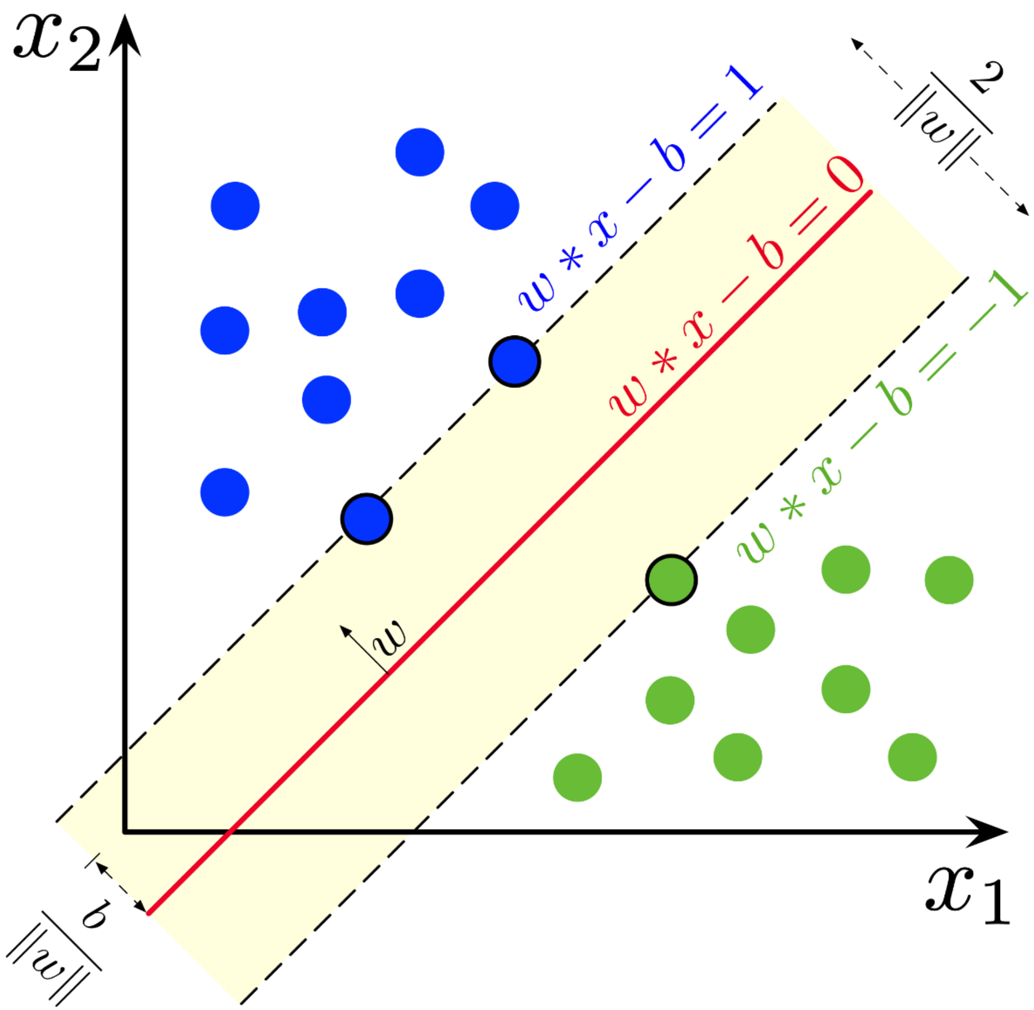
$$\begin{aligned} l_n &= \max(0, 2.3 - (-1.1) + 1) + \max(0, -6.2 - (-1.1) + 1) \\ &= \max(0, 4.4) + \max(0, -4.1) \\ &= 4.4 + 0 \\ &= 4.4 \end{aligned}$$

$$l_n = \sum_{c \in \{\text{kočka}, \text{alpaka}\}} \max(0, s_{n,c} - s_{n,y_n} + 1)$$

$$\begin{aligned} l_n &= \max(0, 3.1 - 5.8 + 1) + \max(0, 2.6 - 5.8 + 1) \\ &= \max(0, -1.7) + \max(0, -2.2) \\ &= 0 + 0 \\ &= 0 \end{aligned}$$

Hard Margin Support Vector Machine (SVM)

- Multiclass hinge loss se snaží, aby každý bod byl přemapován na vzdálenost alespoň Δ od všech ostatních bodů, které patří do jiné třídy
- Výsledkem potom je separující obecně nadrovina taková, která maximalizuje vzdálenost mezi této nadrovině nejbližšími body z rozdílných tříd (tzv. support vektory)
- Úloha je konvexní optimalizace → pokud jsou třídy lineárně separovatelné, vždy najde optimální řešení



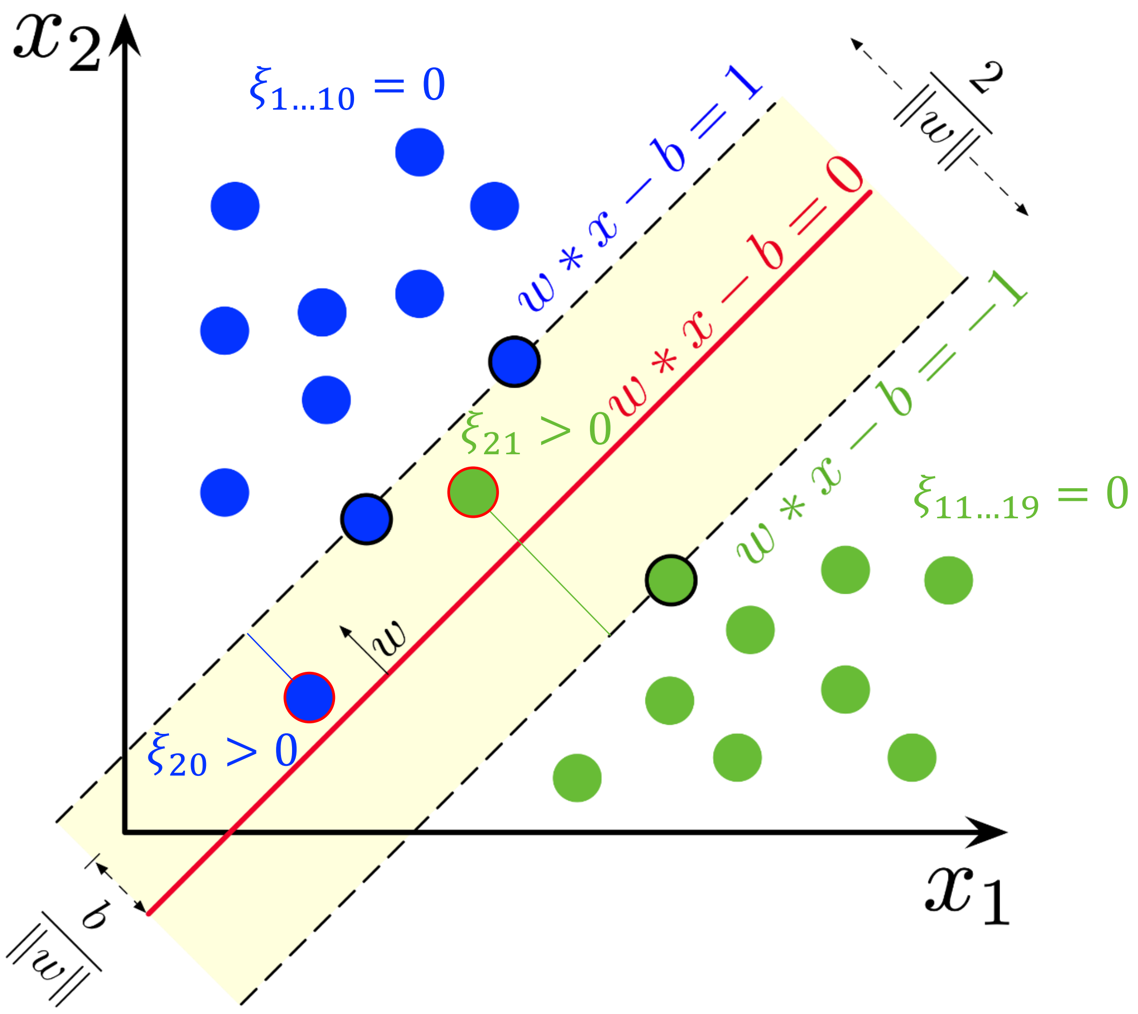
obrázek: https://en.wikipedia.org/wiki/Support_vector_machine

Soft Margin Support Vector Machine (SVM)

- Pokud třídy nejsou lineárně separovatelné, hinge loss nebude nula
- Pro některé body tedy podmínka minimální vzdálenosti nebude splněna
- To, jak moc každý bod $\boldsymbol{x}_n$ podmínku porušuje, říká vnitřek sumy ve vztahu pro hinge loss

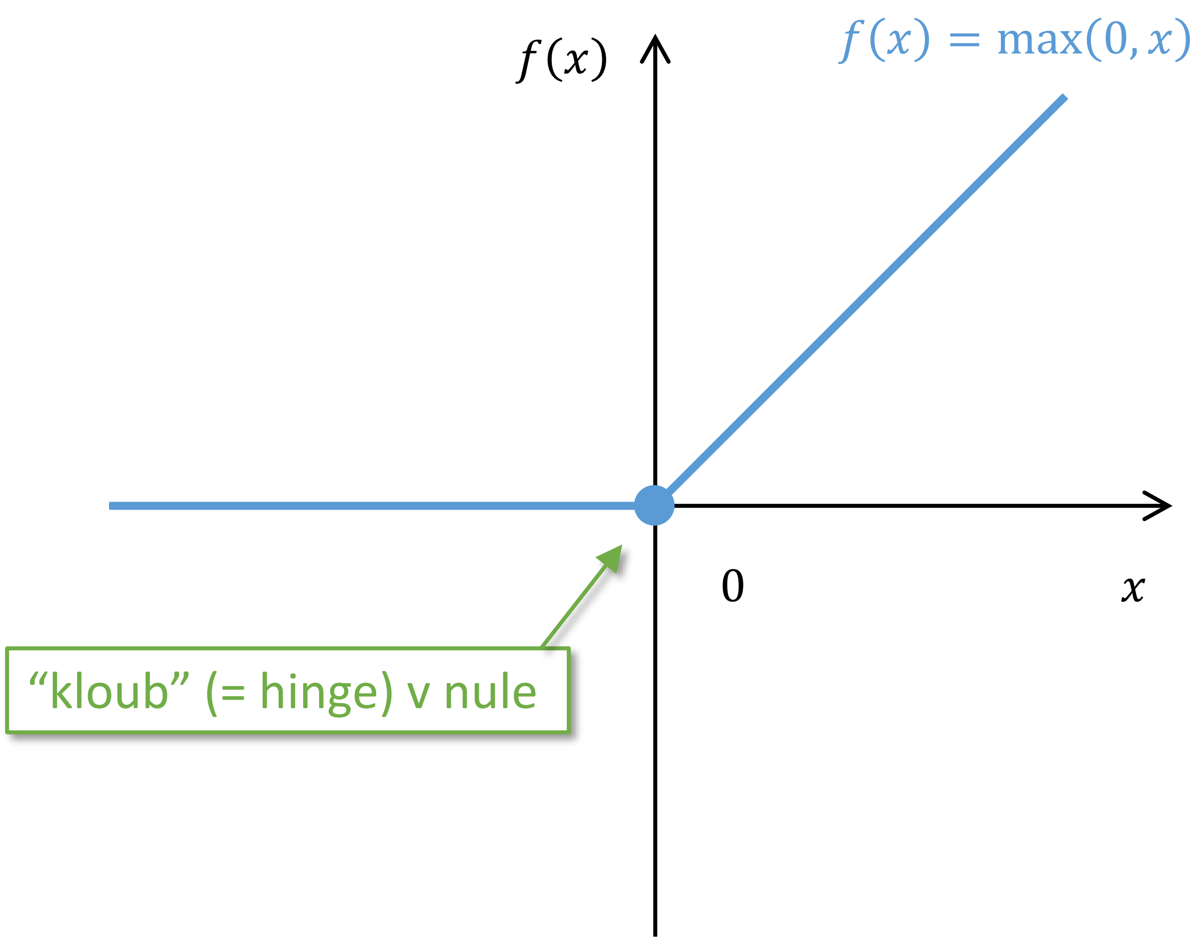
$$\xi_n = \max\left(0, s_{n,c} - s_{n,y_n} + 1\right)$$

- Člen ξ_n se označuje jako tzv. uvolňující proměnná (slack variable)
- Pojmenování pochází z primární formulace SVM jako kvadratické minimalizace s podmínkami, které zavedením ξ_n zmírníme (uvolníme)



obrázek: https://en.wikipedia.org/wiki/Support_vector_machine

Proč název hinge loss?



Citlivost hinge lossu na změnu skóre

$\boldsymbol{x}_n$

$\boxed{1.3}$	kočka
0.8	pes
-0.2	alpaka

$\boldsymbol{s}_n$

$$
\begin{aligned}
l_n &= \max(0, 0.8 - 1.3 + 1) + \max(0, -0.2 - 1.3 + 1) \\
&= \max(0, 0.5) + \max(0, -0.5) \\
&= 0.5 + 0 \\
&= 0.5
\end{aligned}
$$

obrázek: http://cs231n.github.io/linear-classify/

Citlivost hinge lossu na změnu skóre

$\boldsymbol{x}_n$

$\boldsymbol{s}_n$	
1.6	**kočka**
0.8	pes
-0.2	alpaka

lineární skóre target třídy se zvýší (zlepší)

$$
\begin{aligned}
l_n &= \max(0, 0.8 - 1.6 + 1) + \max(0, -0.2 - 1.6 + 1) \\
&= \max(0, 0.2) + \max(0, -0.8) \\
&= 0.2 + 0 \\
&= 0.2
\end{aligned}
$$

hinge loss se zmenší (zlepší)

ke zlepšení došlo, protože skóre kočky nebylo dostatečně daleko od skóre psa

obrázek: http://cs231n.github.io/linear-classify/

Citlivost hinge lossu na změnu skóre

$\boldsymbol{x}_n$

1.6	kočka
0.8	pes
-4.0	alpaka

$\boldsymbol{s}_n$

lineární skóre dostatečně dobře klasifikované třídy se ještě více zmenší

$$
\begin{aligned}
l_n &= \max(0, 0.8 - 1.6 + 1) + \max(0, -4.0 - 1.6 + 1) \\
&= \max(0, 0.2) + \max(0, -4.6) \\
&= 0.2 + 0 \\
&= 0.2
\end{aligned}
$$

hinge loss už se nesníží

k dalšímu zlepšení už nedošlo, protože kočka byla od alpaky již dostatečně dobře oddělena

obrázek: http://cs231n.github.io/linear-classify/

Citlivost hinge lossu na změnu skóre

- Hinge loss se sníží (zlepší):
  - zvýšením skóre správné třídy, pokud je nějaká nesprávná blíže než 1
  - snížením skóre nesprávné třídy, pokud je správné blíže než 1
  - obojím zároveň
- Pokud je podmínka minimální vzdálenosti (marginu) pro nějaký bod splněna pro skóre všech tříd, hinge loss bude nulový a není ho již možné zlepšit
- SVM v sobě tedy zahrnuje “vnitřní regularizaci"

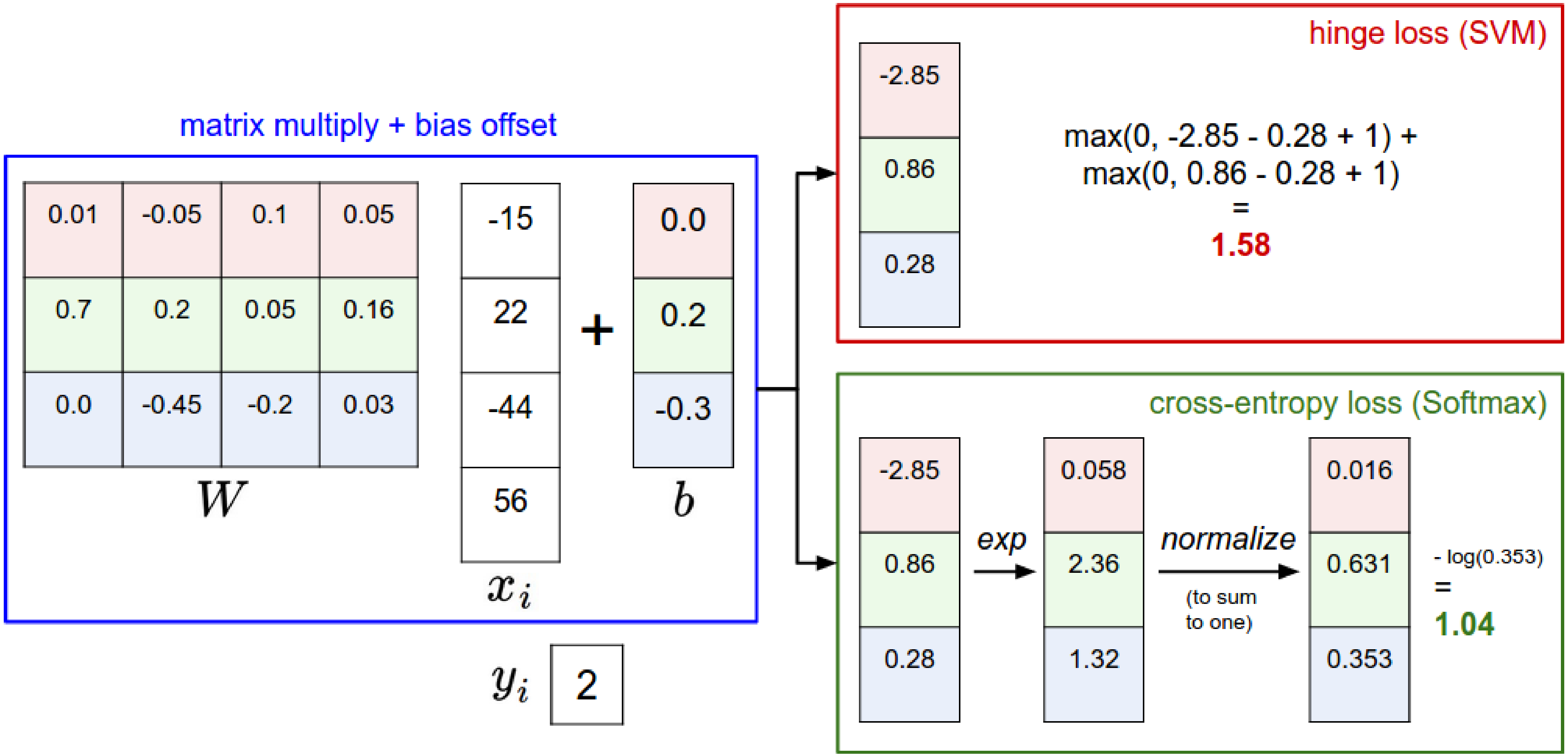
neární skóre dostatečně
obře klasifikované třídy
e ještě více zmenší

šímu zlepšení už nedošlo,
ože kočka byla od alpaky
již dostatečně dobře oddělena

obrázek: http://cs231n.github.io/linear-classify/

Závěr

Lineární model a klasifikační loss



obrázek: http://cs231n.github.io/linear-classify/

Cross entropy vs hinge loss

- SVM zahrnuje vnitřní “regularizaci”: pokud je hinge podmínka u nějakého bodu splněna, kritérium zde má nulovou hodnotu a tedy i přírůstek gradientu od tohoto bodu je nulový
- Logistická regrese naopak bez explicitní regularizace nikdy nekonverguje, skóre se donekonečna snaží zlepšit
- Pro účely lineární klasifikace obě kritéria přibližně stejně dobrá
- Díky robustnosti vůči outlierům na menších datasetech výkonnější spíše SVM
- Trénování neuronových sítí pro klasifikaci v drtivé většině využívá cross entropy

Návrh a trénování lineárního klasifkátoru

1. Navrhneme diskriminativní klasifikační funkci s upravitelnými parametry
2. Kvantifikujeme její úspěšnost klasifikace nějakým kritériem
3. **Nastavíme parametry klasifikátoru tak, abychom optimalizovali zvolené kritérium**

Přednáška 02