

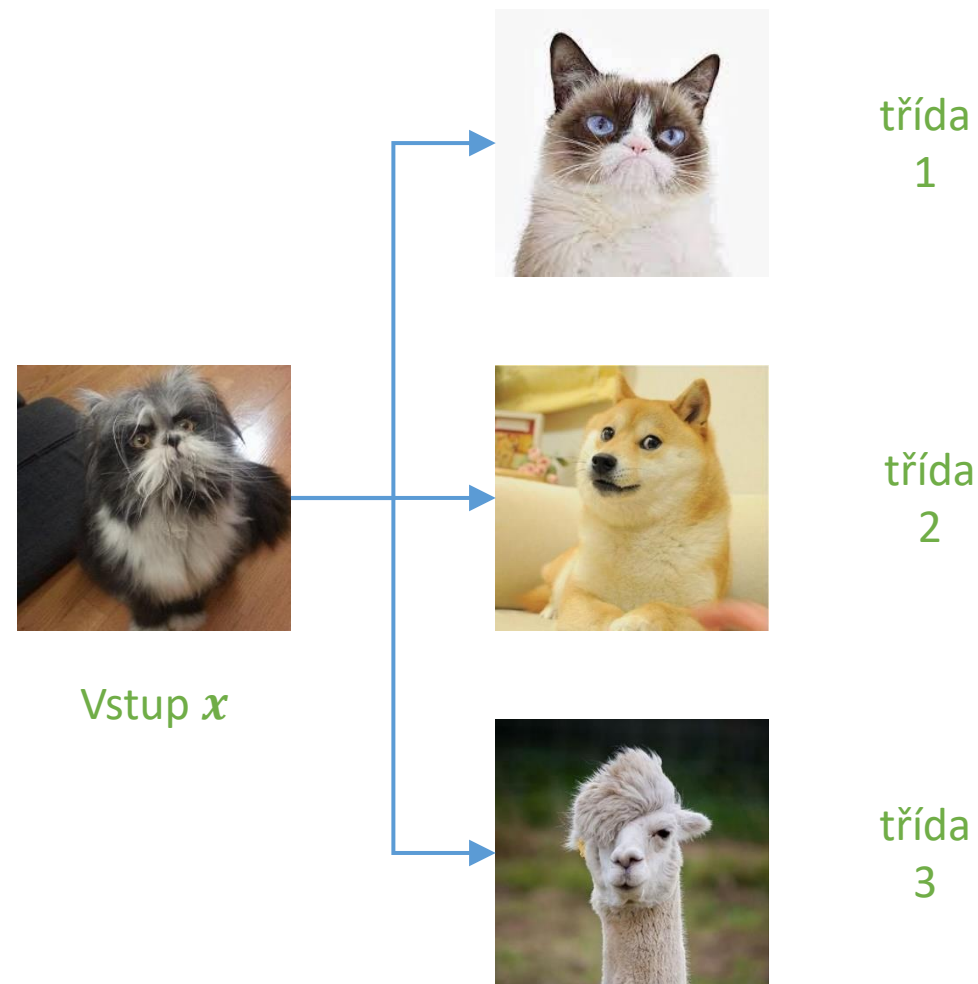
# Aplikace neuronových sítí

---

Lineární klasifikace, Softmax, SVM

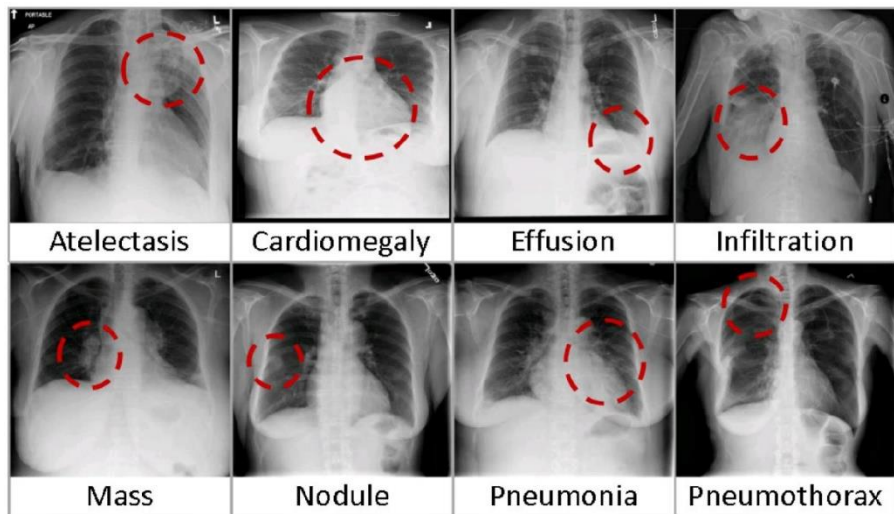
# Úloha klasifikace obrázků

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



# Proč klasifikace?

Klasifikace je zajímavá a užitečná sama o sobě



Amanita\_fulva1



Lepista\_saeva48



Coprinopsis\_atramentaria35



Morchella\_esculenta33



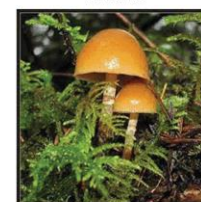
Amanita\_fulva11



Auricularia\_auricula-judae22



Pleurocybella\_porrigena13



Galerina\_sulciiceps19

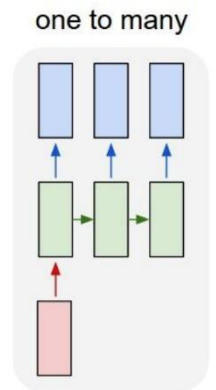


Amanita\_arocheae37

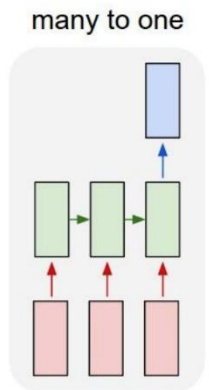


Clavulinaceae15

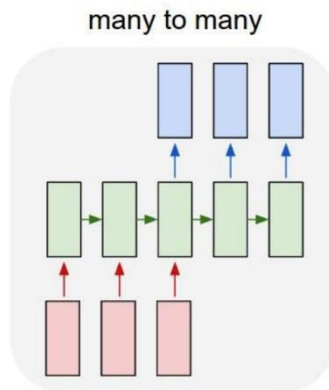
zároveň ale tvoří základ mnoha dalších aplikací



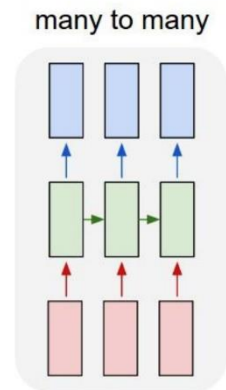
tagování obrázku,  
generování hudby



analýza sentimentu,  
klasifikace videa



strojový překlad,  
rozpznávání řeči



rozpoznávání fonémů, jazykový  
model



# Návrh a trénování lineárního klasifikátoru

1. Navrhujeme 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  $x_n$  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)  $K$

$$\mathbf{s}_n = \underbrace{[s_{n,1}, \dots, s_{n,K}]}_{\text{počet prvků vektoru} = \text{počet tříd } K}^T$$

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

$$\hat{y}_n = \underset{k}{\operatorname{argmax}} \mathbf{s}_n$$

# Lineární klasifikace

- Lineární model předpokládá afinní\* vztah mezi skóre třídy  $s_n$  a vstupem  $x_n$

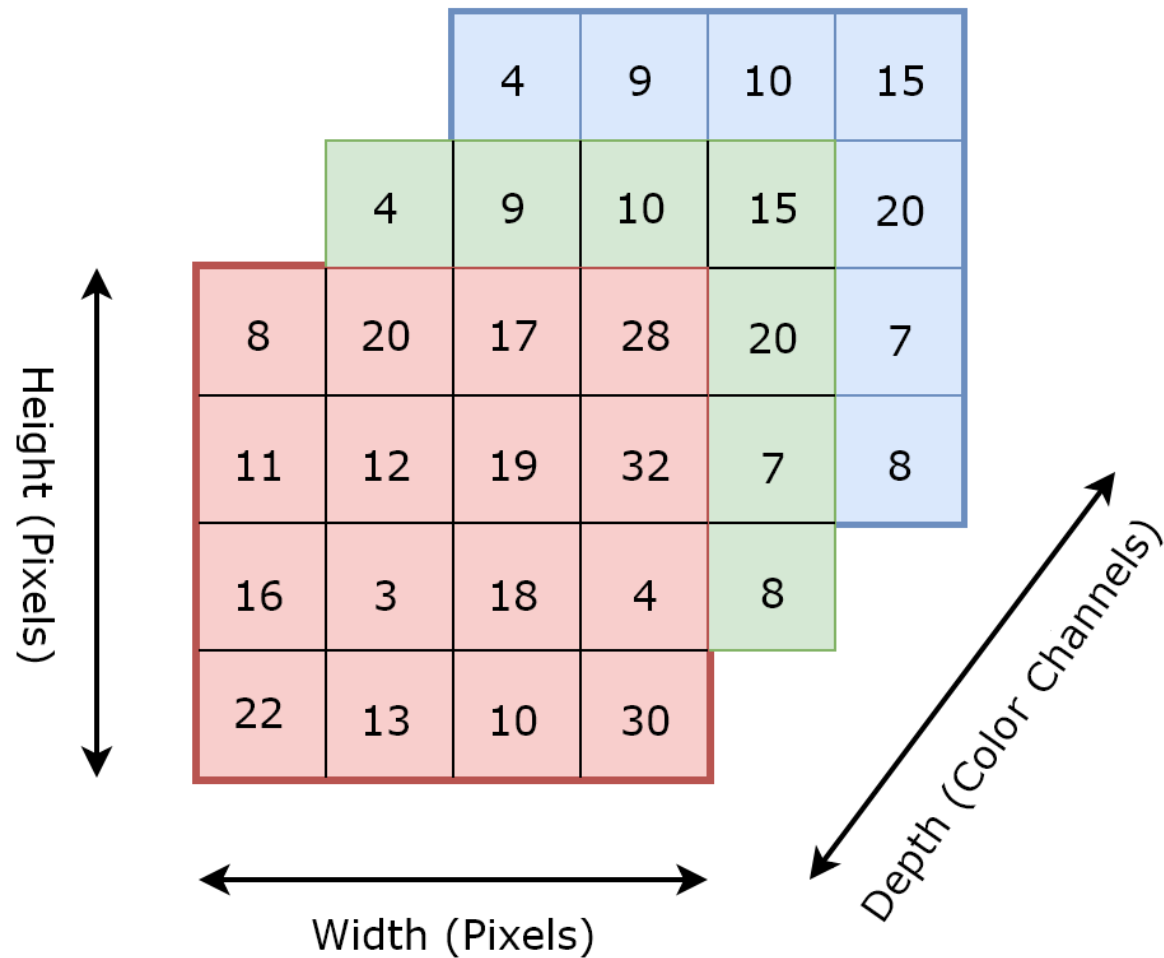
$$s_n = w \cdot x_n + b$$

kde

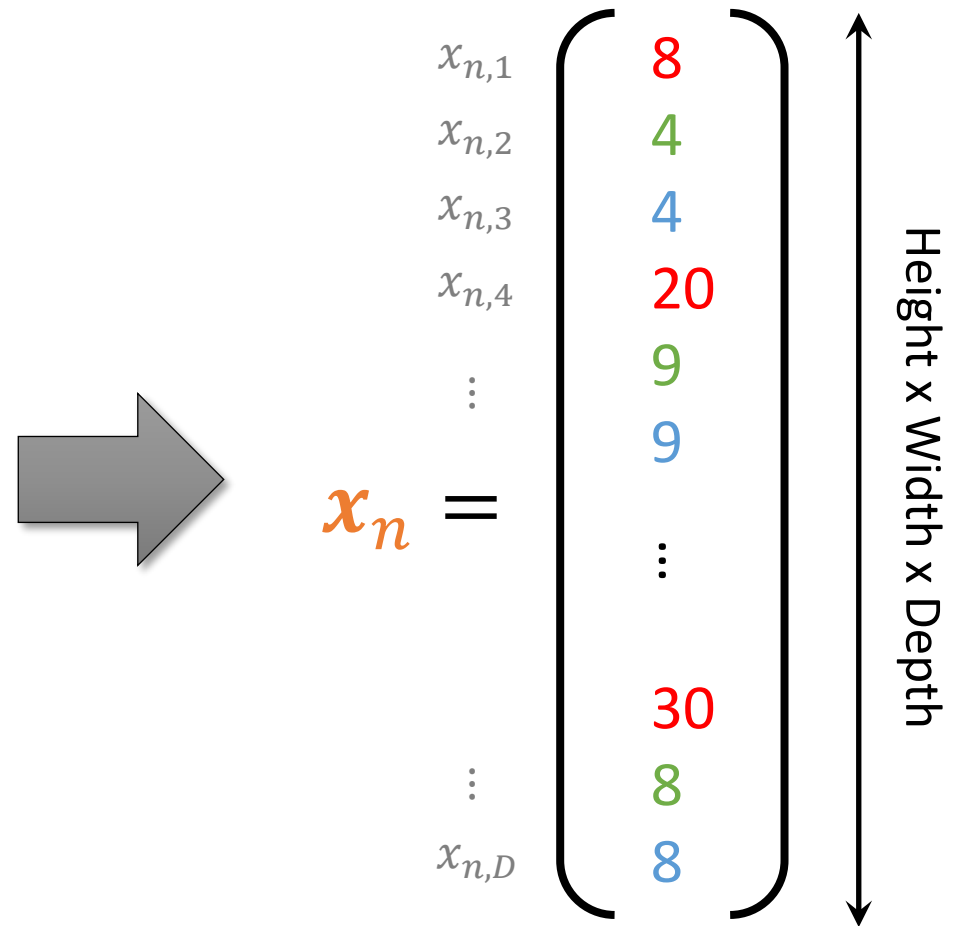
- $x_n$  je sloupcový vektor o rozměru  $D$
- $w$  je matice vah klasifikátoru s rozměry  $K \times D$
- $b$  je sloupcový vektor biasů klasifikátoru s rozměrem  $K$

} parametry klasifikátoru

# Reprezentace RGB obrázku jako vektoru



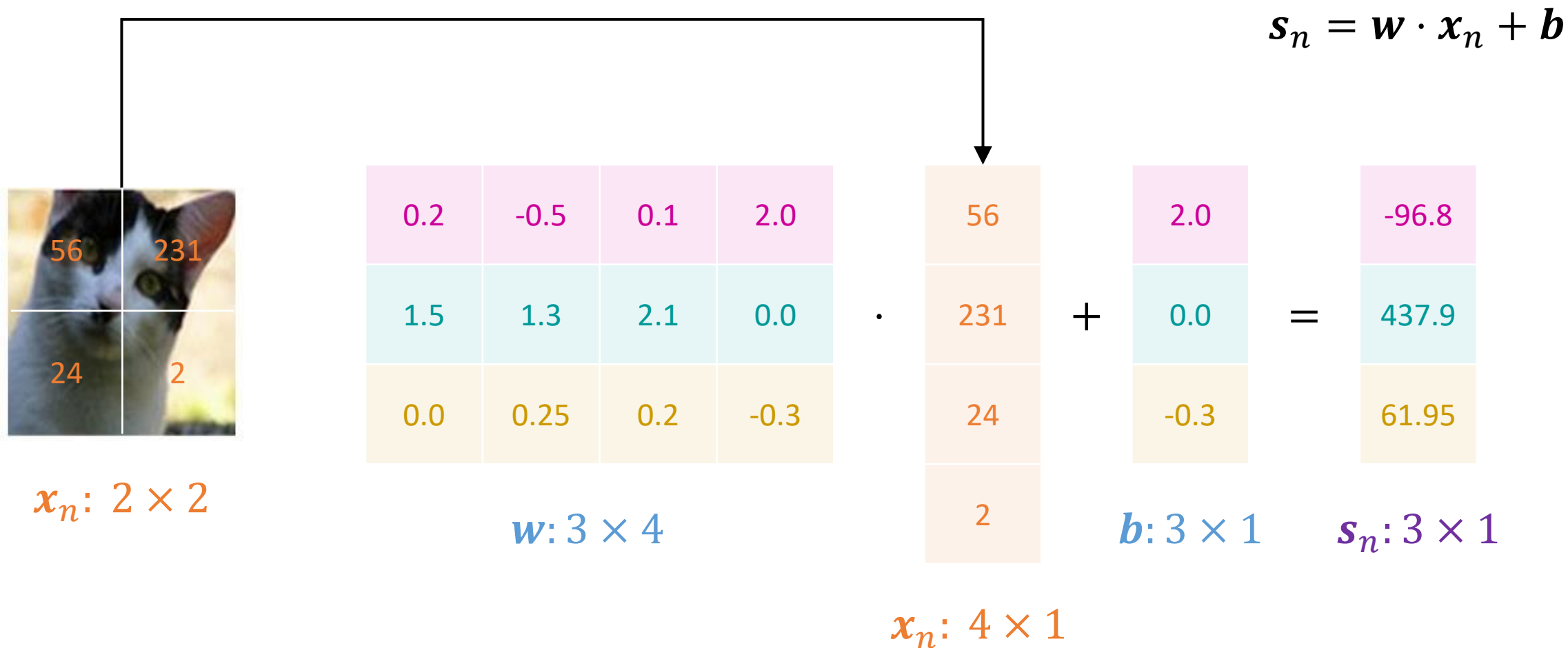
Tensor tvaru (výška, šířka, hloubka) (HWC formát)



Vektor délky  $D = \text{výška} \times \text{šířka} \times \text{hloubka}$



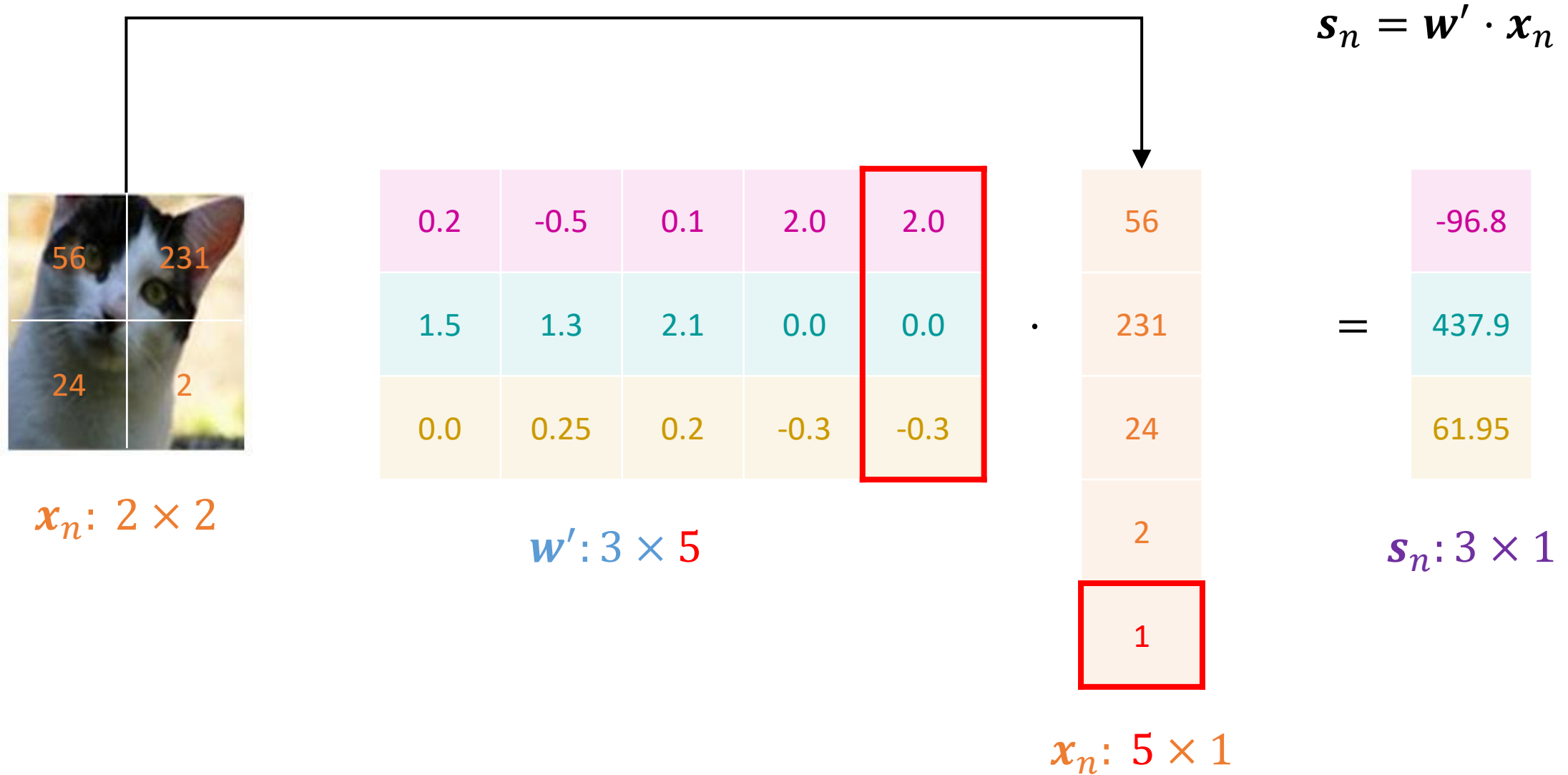
# Lineární predikce skóre: příklad



příklad: <https://cs231n.github.io/linear-classify/>

[https://web.eecs.umich.edu/~justincj/slides/eecs498/WI2022/598\\_WI2022\\_lecture03.pdf](https://web.eecs.umich.edu/~justincj/slides/eecs498/WI2022/598_WI2022_lecture03.pdf)

# Lineární predikce skóre: příklad bez biasu



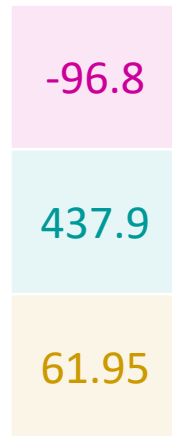
# Linearita predikcí\*: $f(\alpha \cdot x) = \alpha \cdot f(x)$

$$s_n = w' \cdot x_n$$

$$s_m = w' \cdot x_m = w' \cdot (0.5 \cdot x_n) = 0.5 \cdot s_n$$



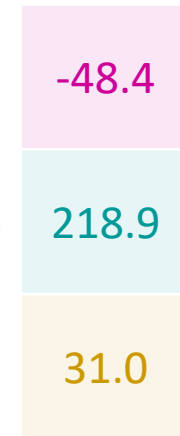
$x_n$



$s_n$



$x_m = 0.5 \cdot x_n$



$s_m$

\*Neplatí pro afinní model s biasem

# Geometrická interpretace

skalární součin

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

$x = [x_1; x_2]$  ... příznakový vektor obrázku

$w = [w_1; w_2]$  ... normálový vektor dělicí přímky

přímka prochází počátkem  $\rightarrow$  posun  $b = 0$

btw MATLABovská notace

$$[u; v] = \begin{bmatrix} u \\ v \end{bmatrix}$$

# Geometrická interpretace

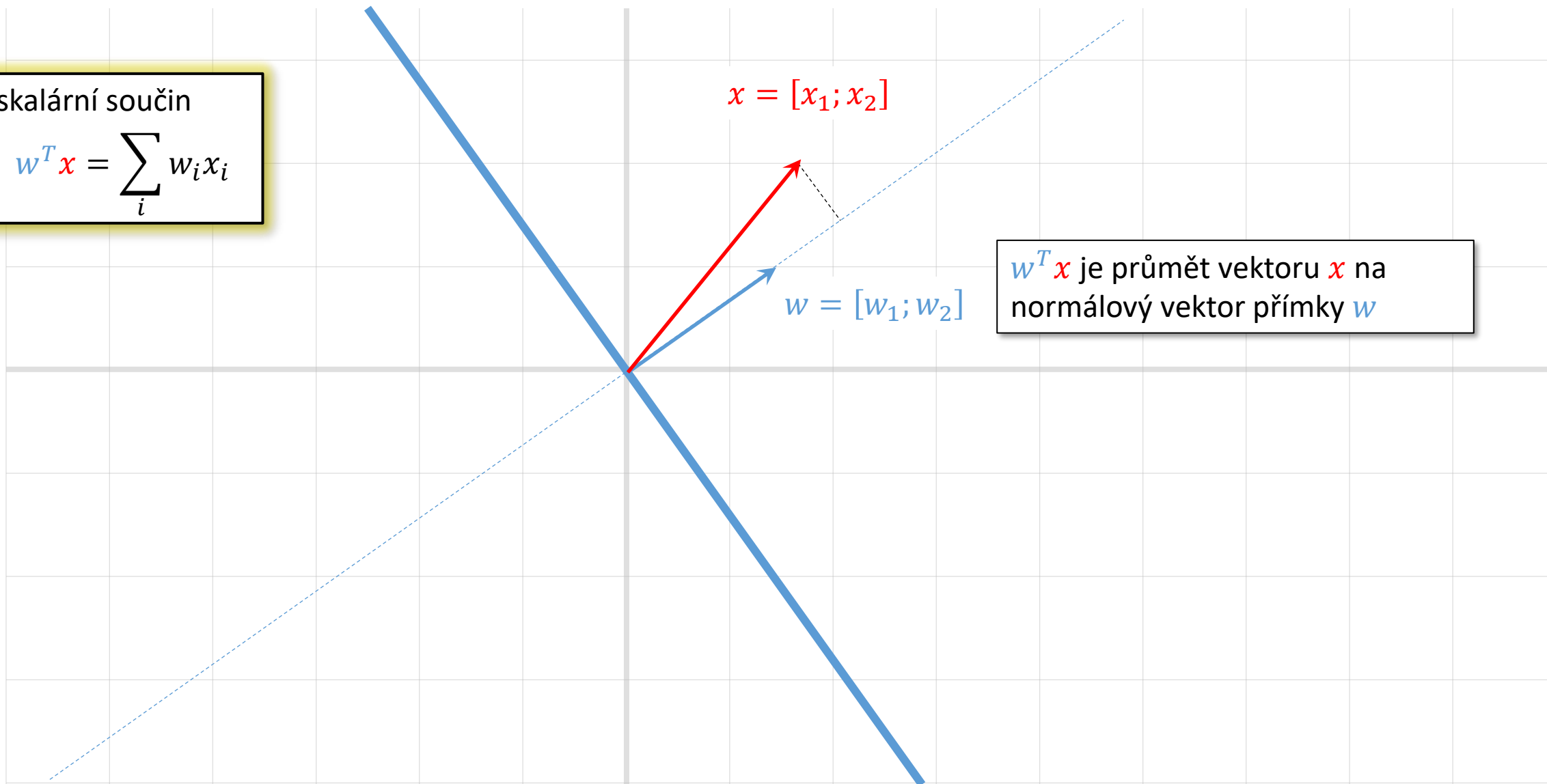
skalární součin

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

$$x = [x_1; x_2]$$

$$w = [w_1; w_2]$$

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



# Geometrická interpretace

skalární součin

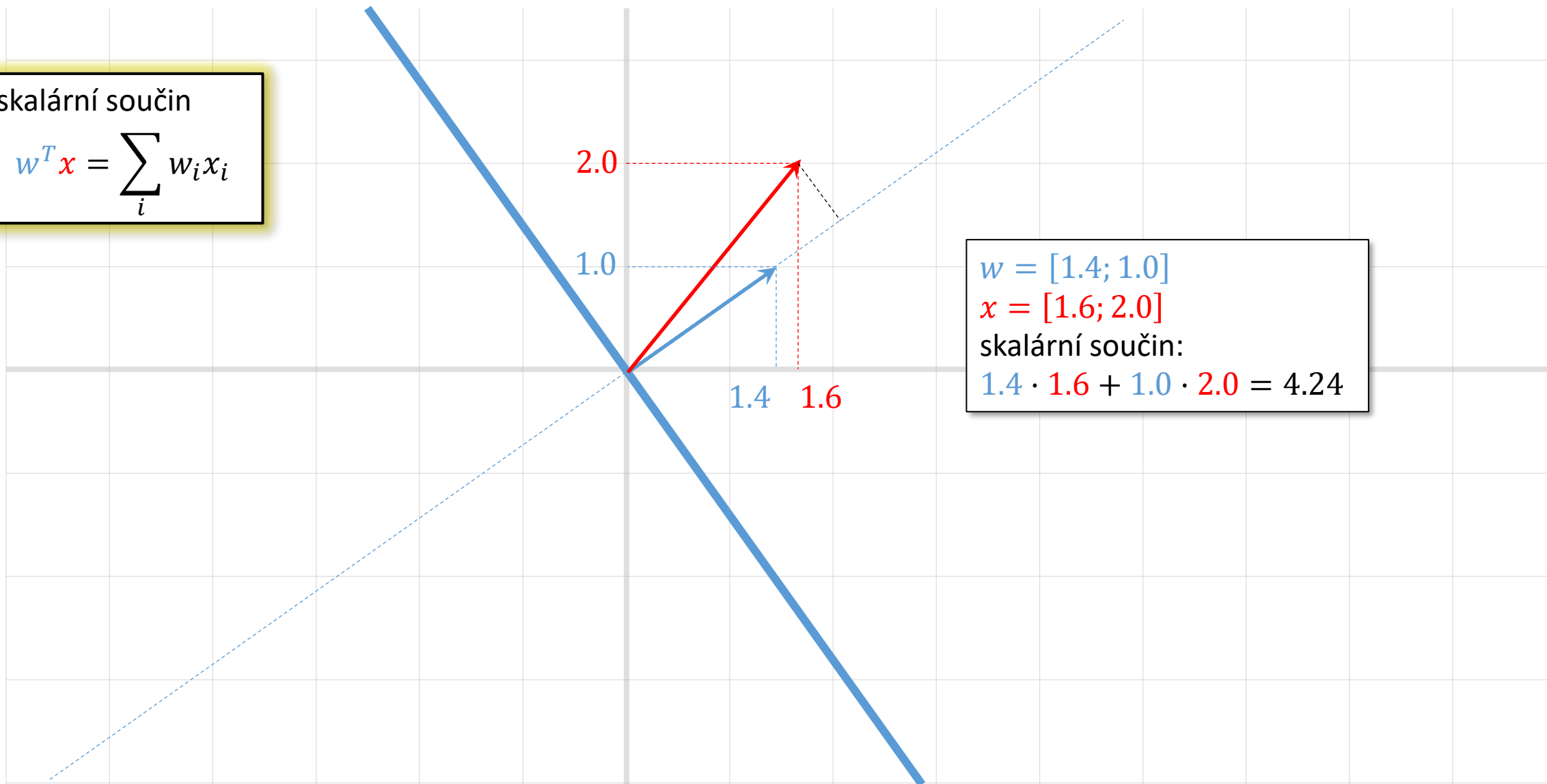
$$w^T x = \sum_i w_i x_i$$

$$w = [1.4; 1.0]$$

$$x = [1.6; 2.0]$$

skalární součin:

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



# Geometrická interpretace

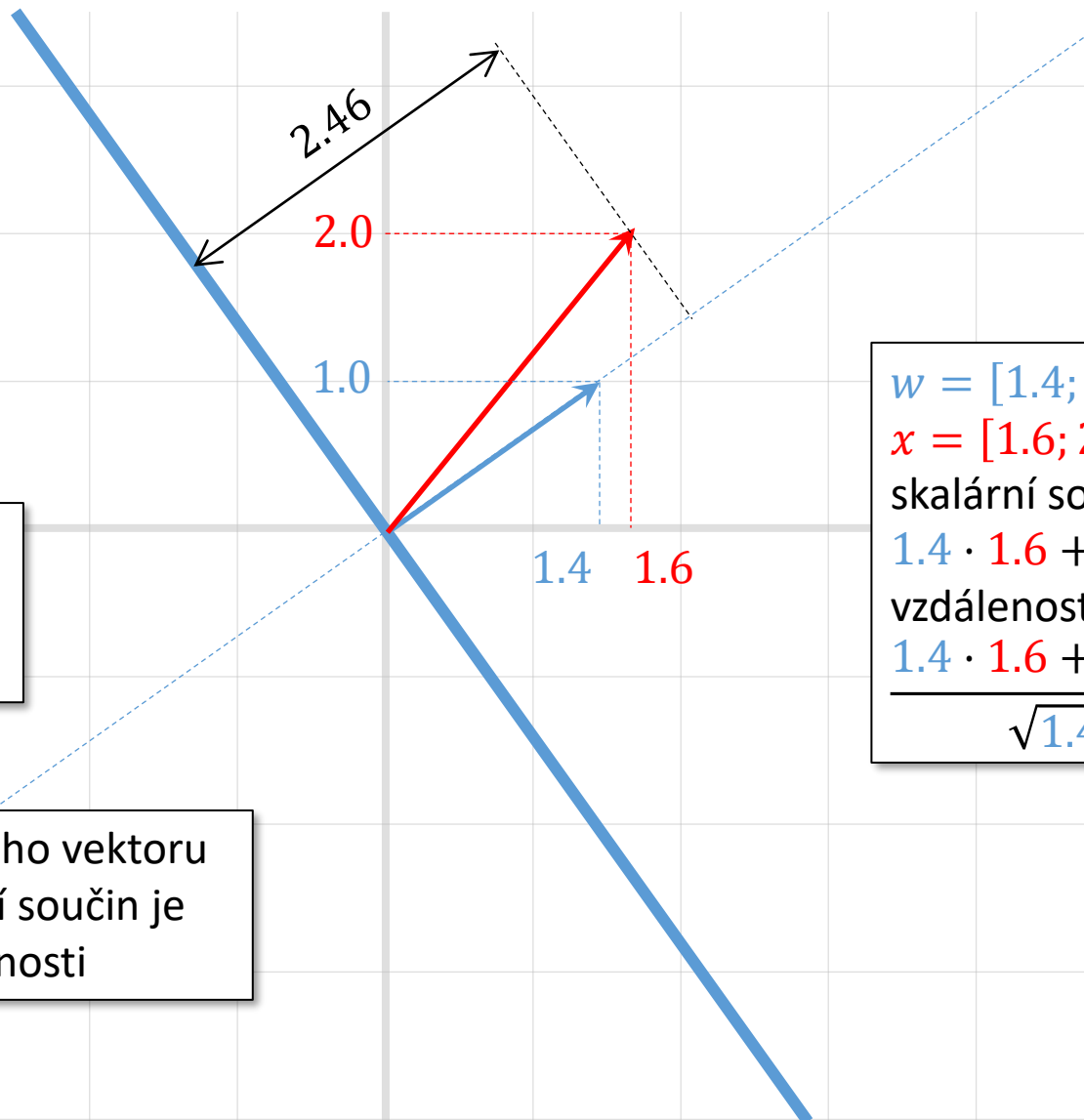
skalární součin

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

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:

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

2.46

# Geometrická interpretace

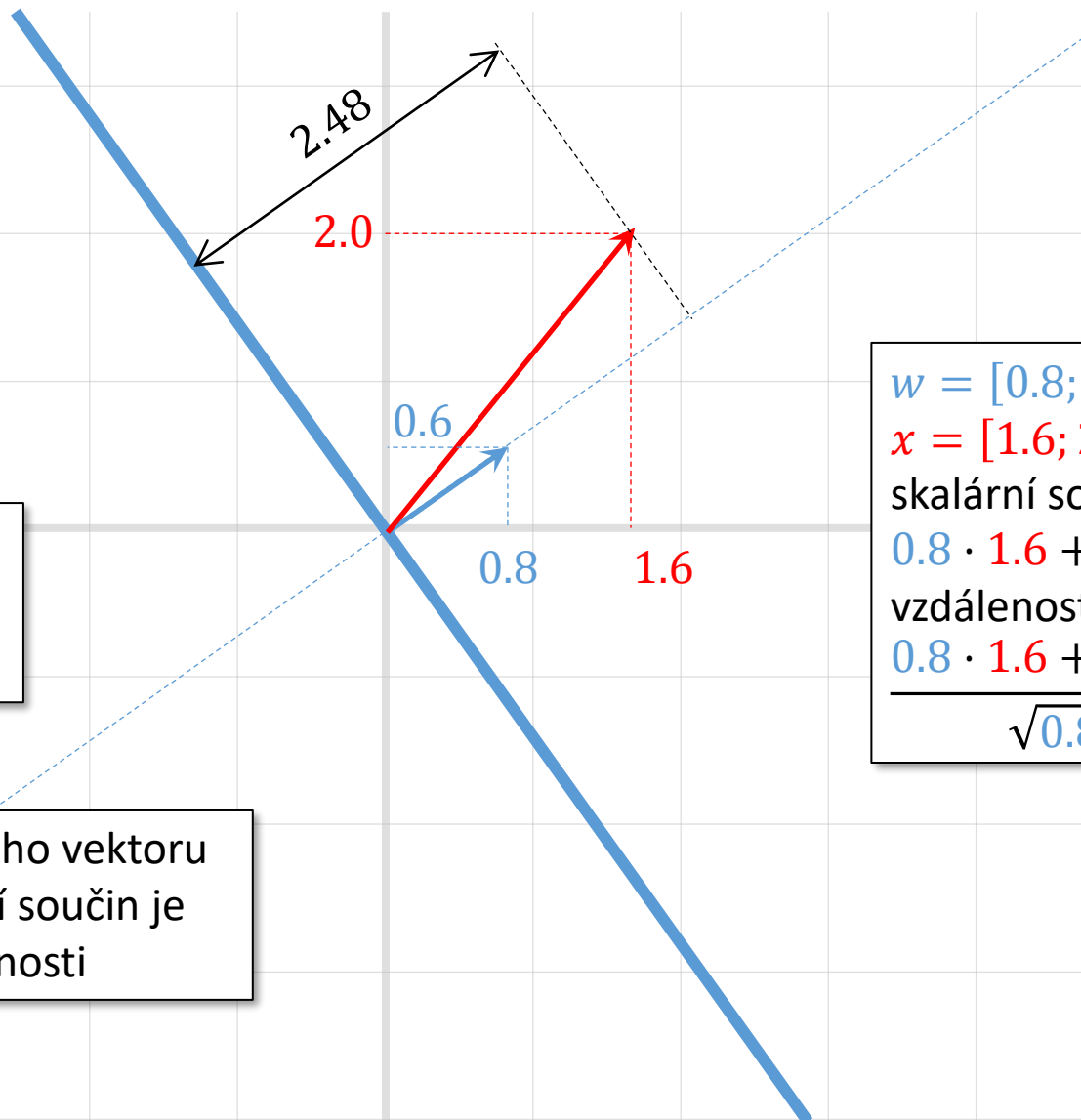
skalární součin

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

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:

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

2.48

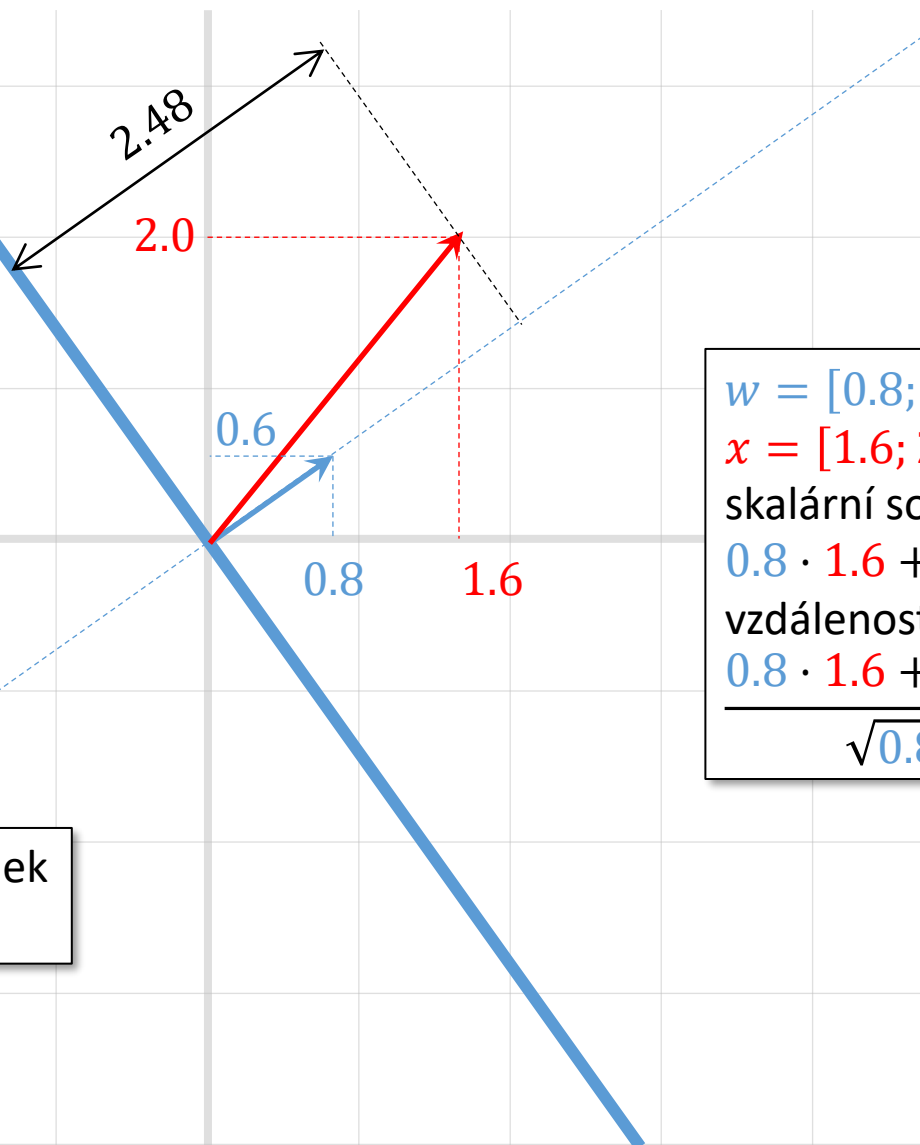


# Geometrická interpretace

skalární součin

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

pro body „nad“ přímkou bude výsledek  
vždy kladný



$$w = [0.8; 0.6]$$

$$x = [1.6; 2.0]$$

skalární součin: (zaokrouhlování)

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

vzdálenost:

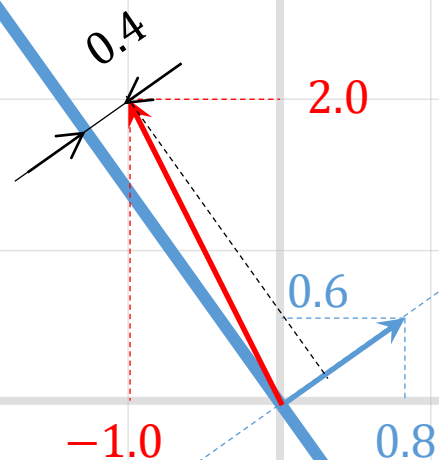
$$\frac{0.8 \cdot 1.6 + 0.6 \cdot 2.0 + 0.0}{\sqrt{0.8^2 + 0.6^2}} = 2.48$$

2.48

# Geometrická interpretace

skalární součin

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



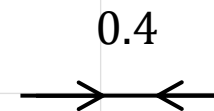
$$w = [0.8; 0.6]$$

$$x = [-1.0; 2.0]$$

skalární součin:

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

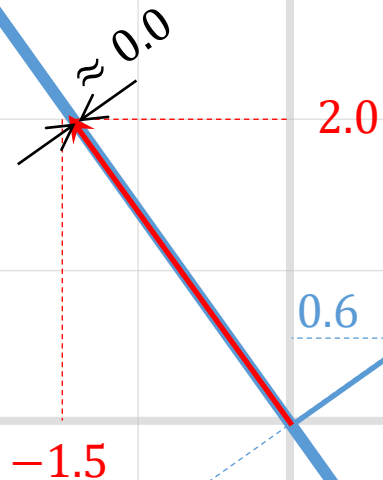
pro body „nad“ přímkou bude výsledek  
vždy kladný



# 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:

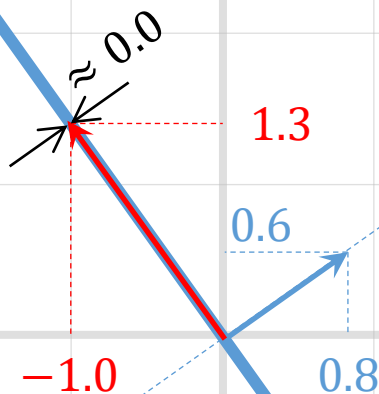
$$0.8 \cdot -1.5 + 0.6 \cdot 2.0 = 0.0$$

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:

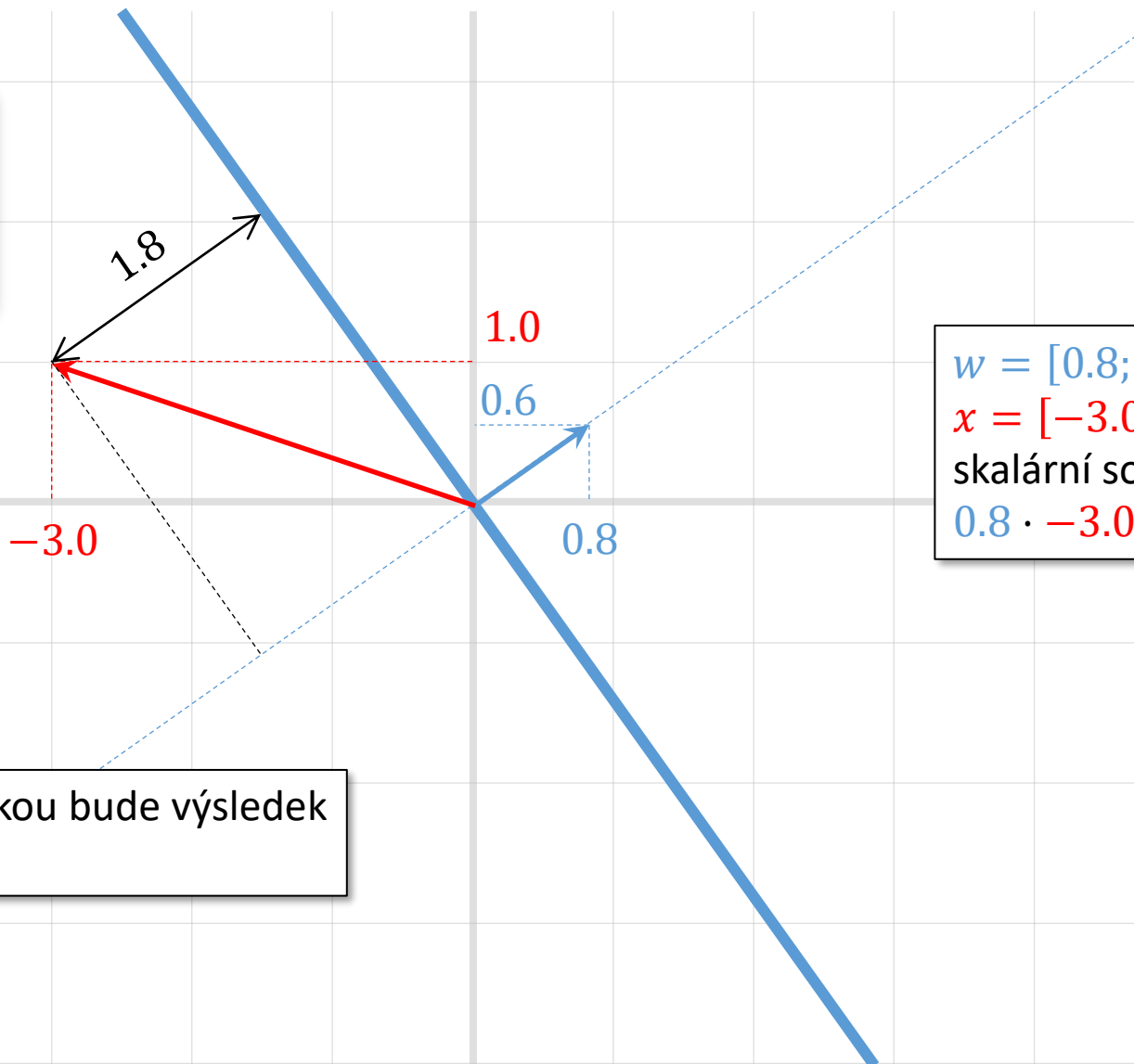
$$0.8 \cdot -1.0 + 0.6 \cdot 1.3 \approx 0.0$$

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:

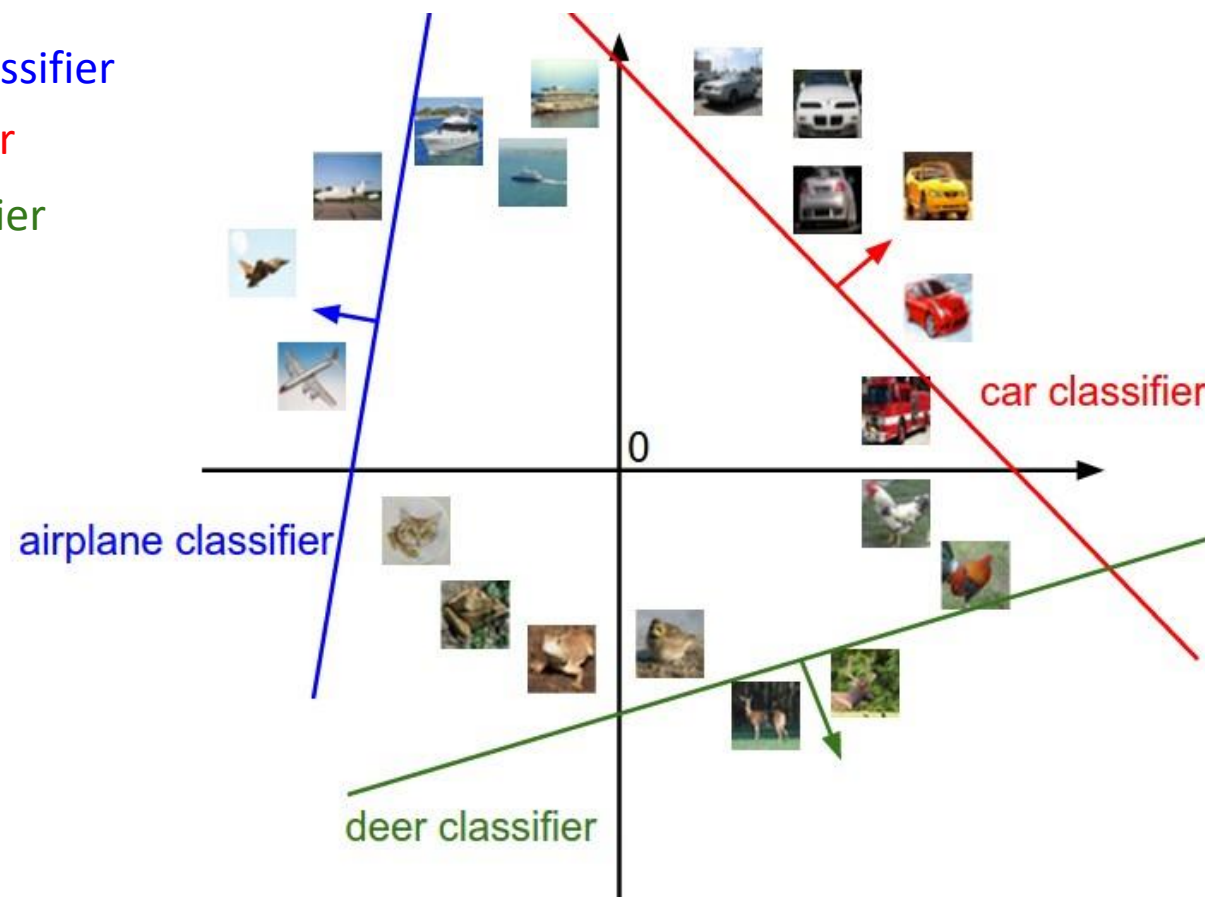
$$0.8 \cdot -3.0 + 0.6 \cdot 1.0 = -1.8$$

pro body „pod“ přímkou bude výsledek  
vždy záporný

# Geometrická interpretace

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

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

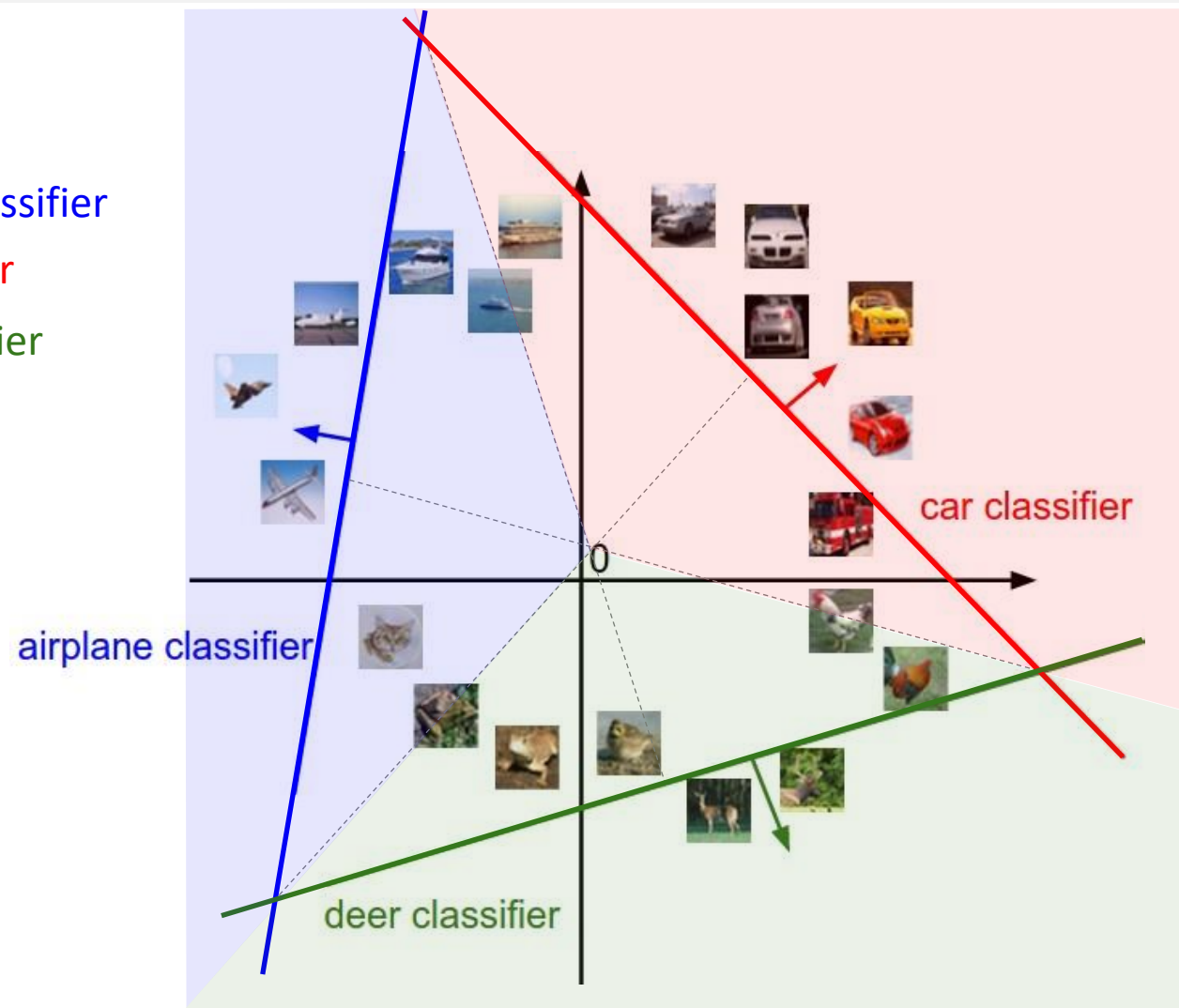


# Geometrická interpretace

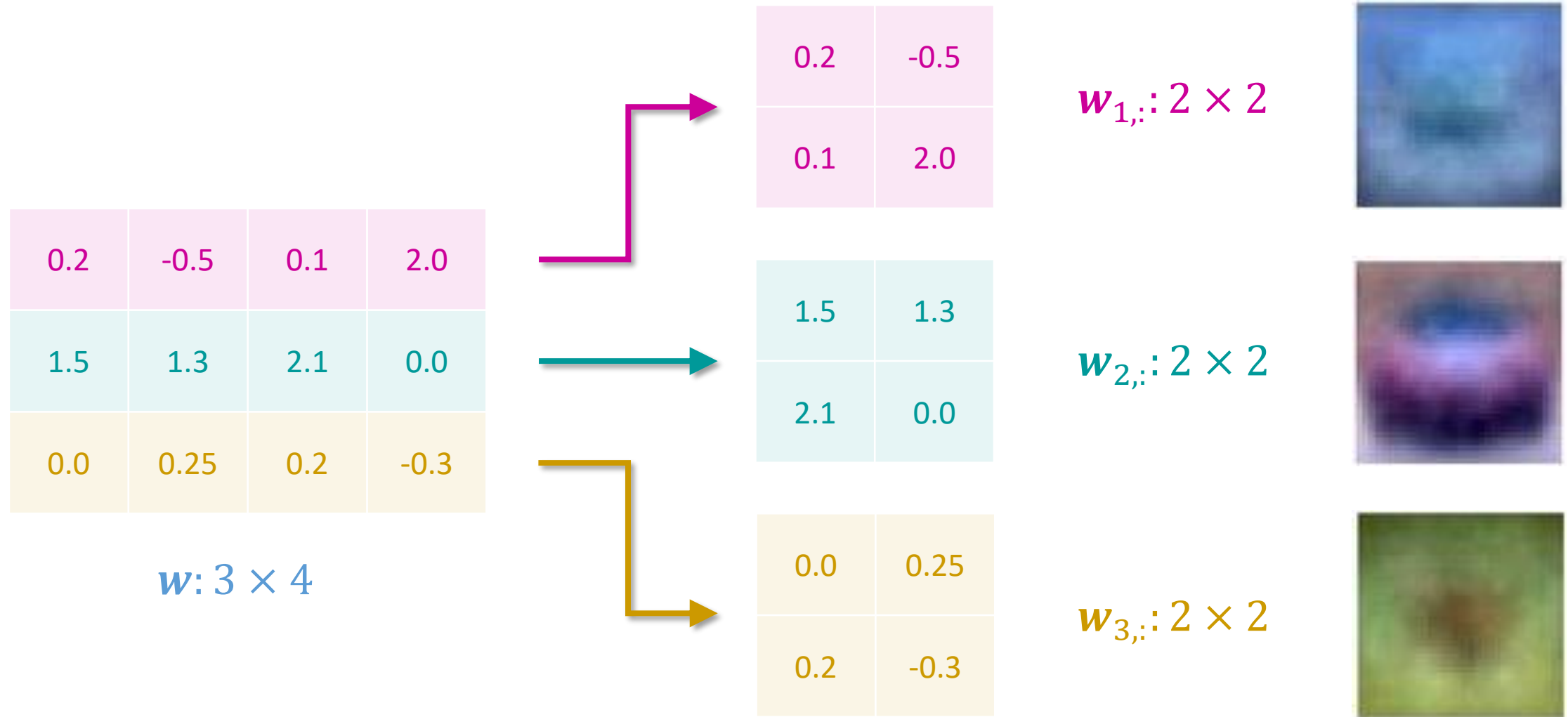
$$\mathbf{w} = \begin{bmatrix} \mathbf{w}_{1,1} & \mathbf{w}_{1,2} & \dots & \mathbf{w}_{1,D} \\ \mathbf{w}_{2,1} & \mathbf{w}_{2,2} & \dots & \mathbf{w}_{2,D} \\ \mathbf{w}_{3,1} & \mathbf{w}_{3,2} & \dots & \mathbf{w}_{3,D} \end{bmatrix}$$

airplane classifier  
car classifier  
deer classifier

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



# Vizuální interpretace: párování vzorů (template matching)



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  $\mathbf{x} \cdot \mathbf{w} + b$  a toto skóre, jak dále uvidíme, chceme maximalizovat (pro správnou třídu)
- Kdy bude skóre maximální?

$$\begin{aligned}\mathbf{w}^* &= \underset{\mathbf{w}}{\operatorname{argmax}} \mathbf{x} \cdot \mathbf{w} + b && \# \text{ bias zanedbáme a přepíšeme skalární součin} \\ &= \underset{\mathbf{w}}{\operatorname{argmax}} \|\mathbf{x}\| \cdot \|\mathbf{w}\| \cdot \cos \phi(\mathbf{x}, \mathbf{w}) && \# \|\mathbf{x}\| \text{ je konst., } \|\mathbf{w}\| \text{ ovlivňuje pouze škálu, nikoliv podobu} \\ &= \underset{\mathbf{w}}{\operatorname{argmax}} \cos \phi(\mathbf{x}, \mathbf{w}) && \# \text{ kosinus má maximum v nule: } \cos(0) = 1 \\ &= \alpha \cdot \mathbf{x} && \# \text{ úhel } \phi(\mathbf{x}, \mathbf{w}) \text{ mezi } \mathbf{x} \text{ a } \mathbf{w} \text{ bude nula právě když } \mathbf{w} = \alpha \cdot \mathbf{x}\end{aligned}$$

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

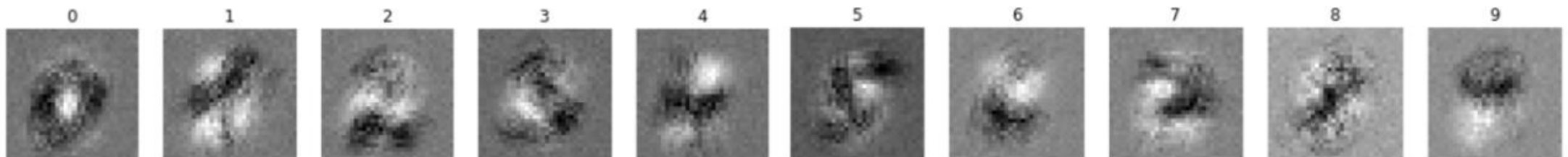
# Vizuální interpretace: párování vzorů (template matching)

## Dataset CIFAR-10



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

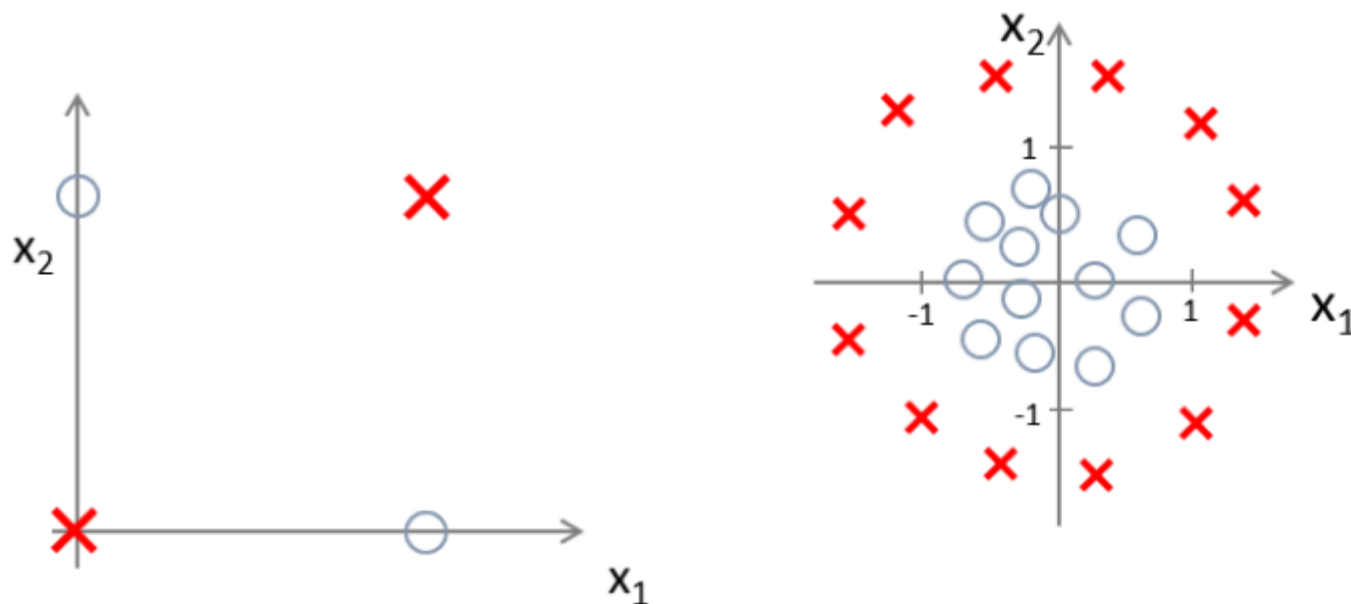
## Dataset MNIST (invertované: černá = vysoká hodnota, bílá = nízká hodnota)



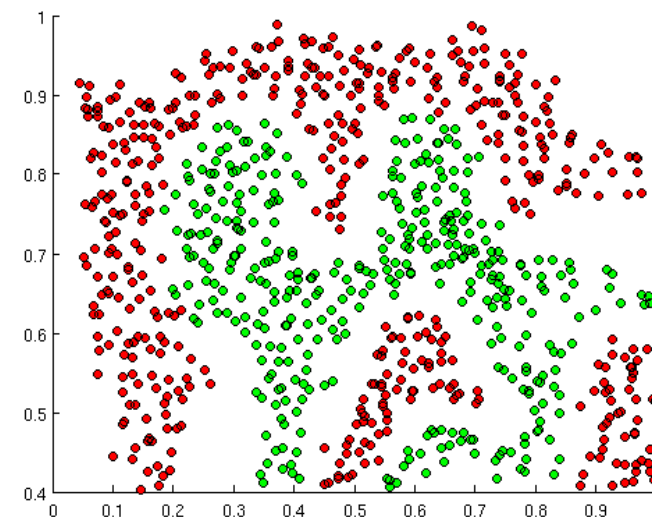
obrázek: USU, přednáška 7

# Lineárně neseparovatelné případy

XOR



obrázky: USU, přednáška 9



obrázek:

<http://openclassroom.stanford.edu/...>

# Návrh a trénování lineárního klasifikátoru

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  $\hat{y}_n \in \{1, \dots, K\}$

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

- Pro každý obrázek  $\mathbf{x}_n$  přitom známe správnou odpověď  $y_n \in \{1, \dots, K\}$  (target)
- Celkem máme dataset  $N$  obrázků a tedy i párů  $(\mathbf{x}_n, y_n)$

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

$$a_n = \mathbb{1}(\hat{y}_n = y_n) \quad \leftarrow \text{Pro jeden obrázek}$$

$$a = \frac{1}{N} \sum_{n=1}^N a_n \quad \leftarrow \text{Pro celý dataset}$$

- Čím větší přesnost  $a$ , tím lépe

# Nepřesnost klasifikace (misclassification rate)

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

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

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

- 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  $(\mathbf{x}_n, \mathbf{y}_n)$  v trénovacím datasetu  $\mathbf{X}$  spočteme

$$l_n = L_n(\mathbf{x}_n, \mathbf{y}_n, \mathbf{w}, \mathbf{b})$$

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

$$L_n(\mathbf{x}_n, \mathbf{y}_n, \mathbf{w}, \mathbf{b}) = \llbracket \arg\max_k \mathbf{w} \cdot \mathbf{x}_n + \mathbf{b} \neq \mathbf{y}_n \rrbracket$$

Iverson bracket notation

$$\llbracket p \rrbracket = \begin{cases} 1 & p \text{ je pravdivá} \\ 0 & \text{jinak} \end{cases}$$

- A zprůměrujeme přes celý dataset

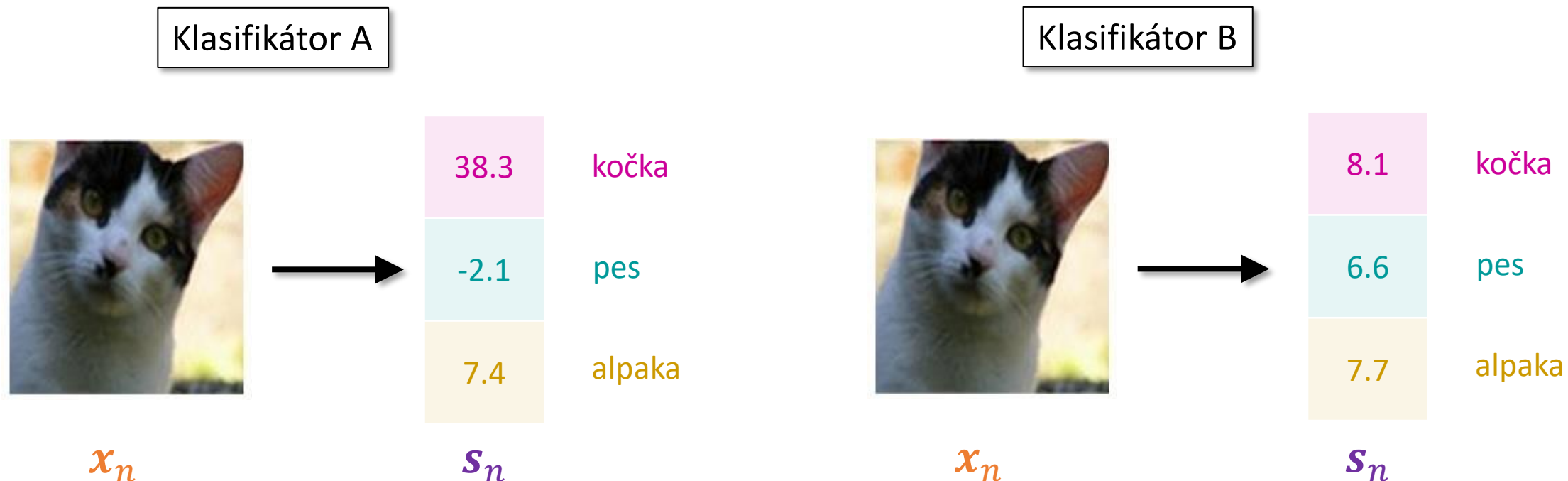
$$L(\mathbf{X}, \mathbf{w}, \mathbf{b}) = \frac{1}{N} \sum_{n=1}^N L_n(\mathbf{x}_n, \mathbf{y}_n, \mathbf{w}, \mathbf{b})$$

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



# Význam skóre

- Který klasifikátor je lepší?



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

# Multiclass cross entropy

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

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

Pro jeden obrázek

- kde

$$\mathbf{p}_n = [p_{n,1}, \dots, p_{n,K}]^T \quad \dots \text{cílové rozdělení (ground truth / target)}$$

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

jsou vektory, na které nahlížíme jako na diskrétní pravděpodobnostní rozdělení

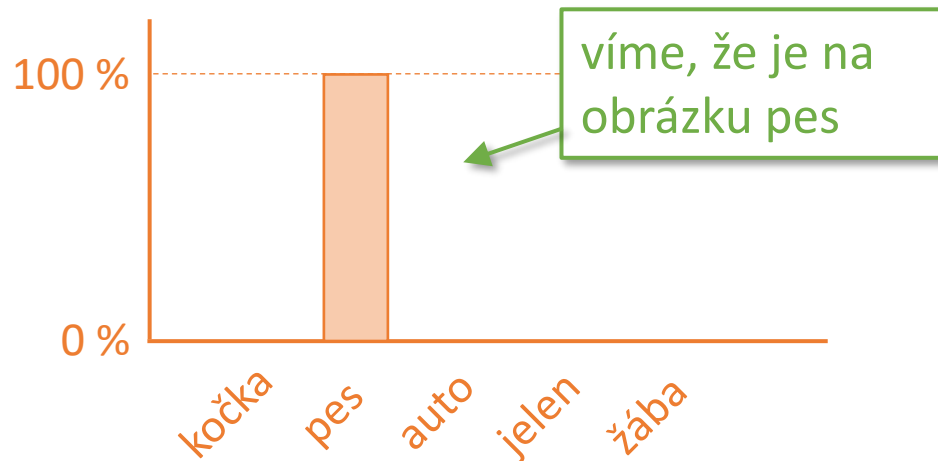
→ cross entropy = rozdíl mezi dvěma pravděpodobnostními rozděleními

# Multiclass cross entropy

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

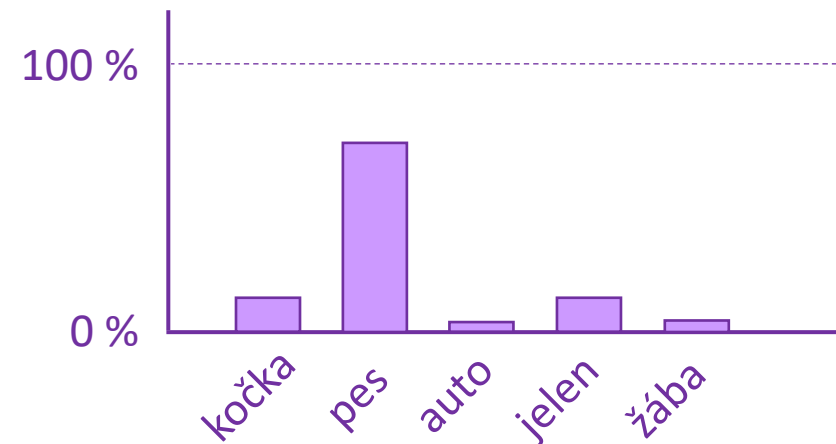
$$\mathbf{p}_n = [p_{n,1}, \dots, p_{n,K}]^T$$

cílové rozdělení (ground truth / target)



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

výstup (predikce) klasifikátoru



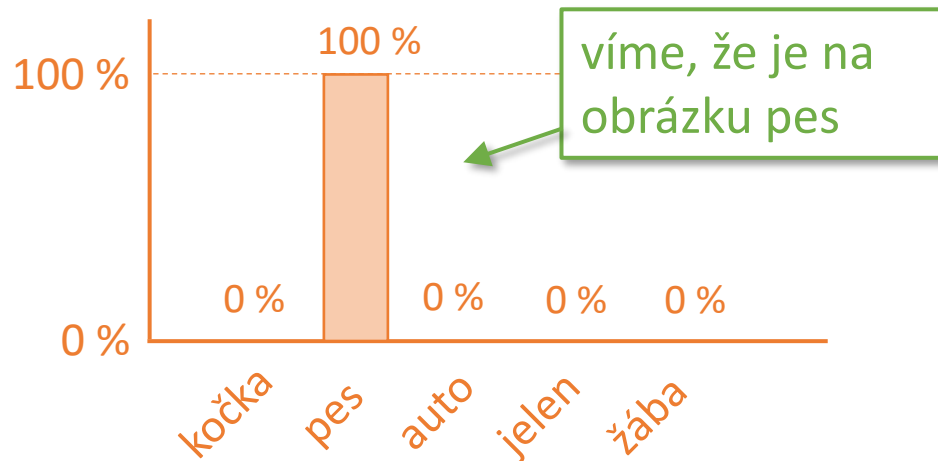
# Multiclass cross entropy



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

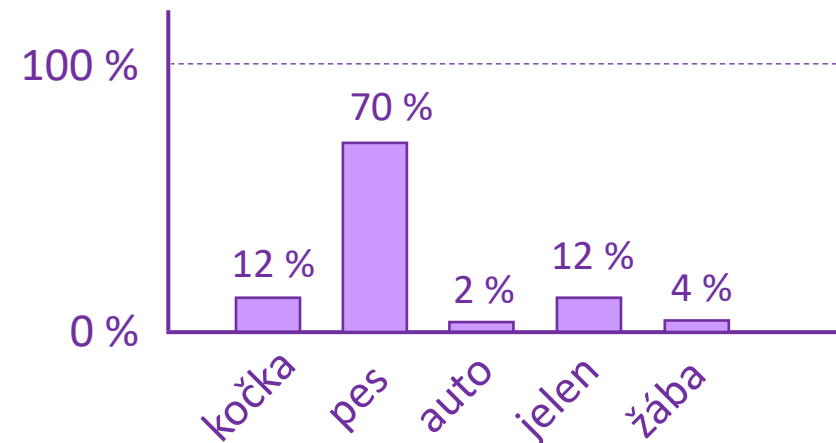
$$\mathbf{p}_n = [p_{n,1}, \dots, p_{n,K}]^T$$

cílové rozdělení (ground truth / target)



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

výstup (predikce) klasifikátoru



# Multiclass cross entropy

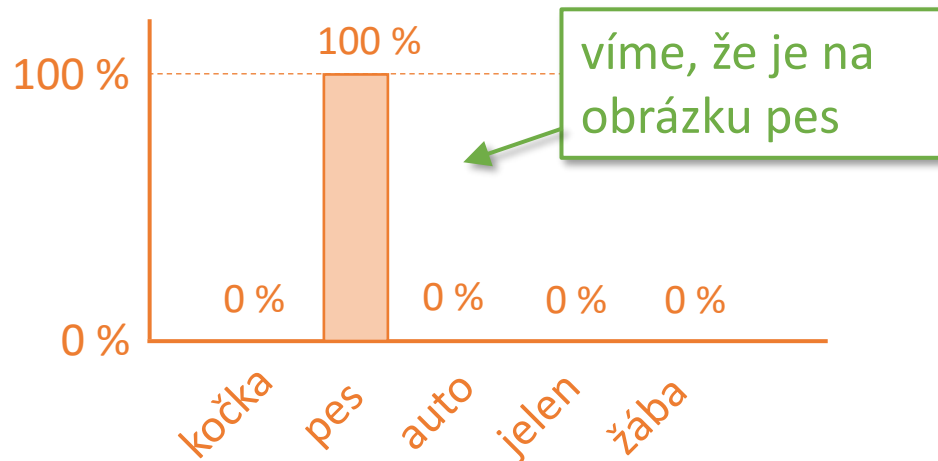


$$l_n = -0 \cdot \log 0.19 - 1 \cdot \log 0.23 - 0 \cdot \log 0.18 - 0 \cdot \log 0.20 - 0 \cdot \log 0.20 \\ = 1.470$$

loss je vyšší

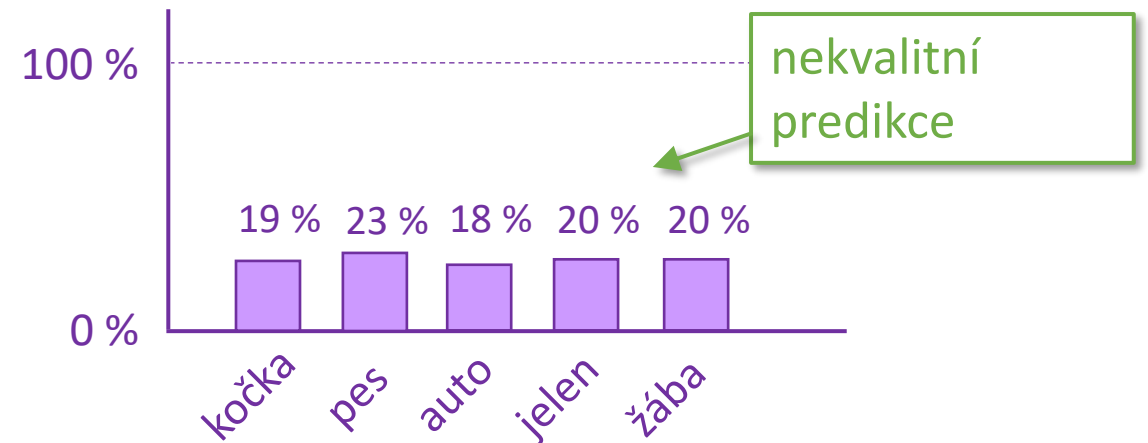
$$\mathbf{p}_n = [p_{n,1}, \dots, p_{n,K}]^T$$

cílové rozdělení (ground truth / target)



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

výstup (predikce) klasifikátoru



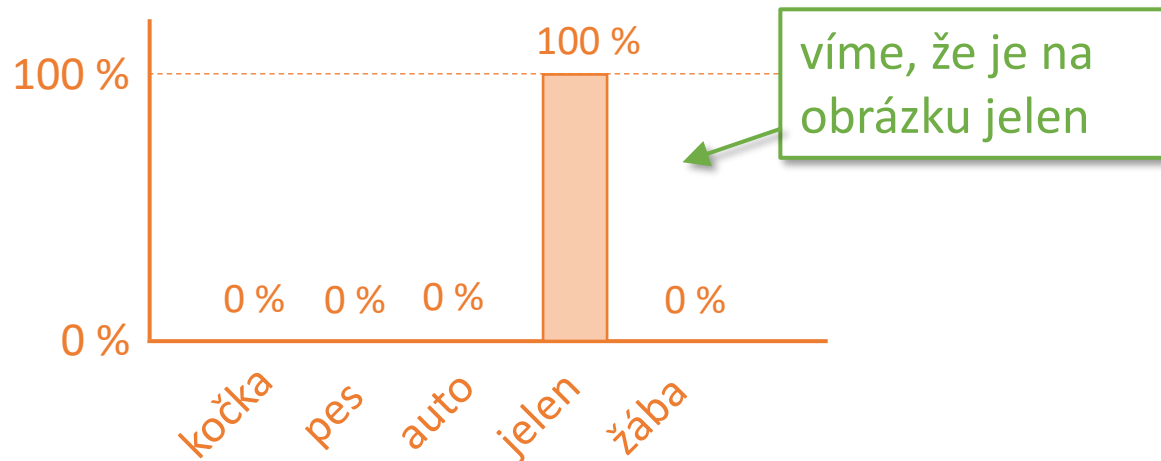
# Multiclass cross entropy



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

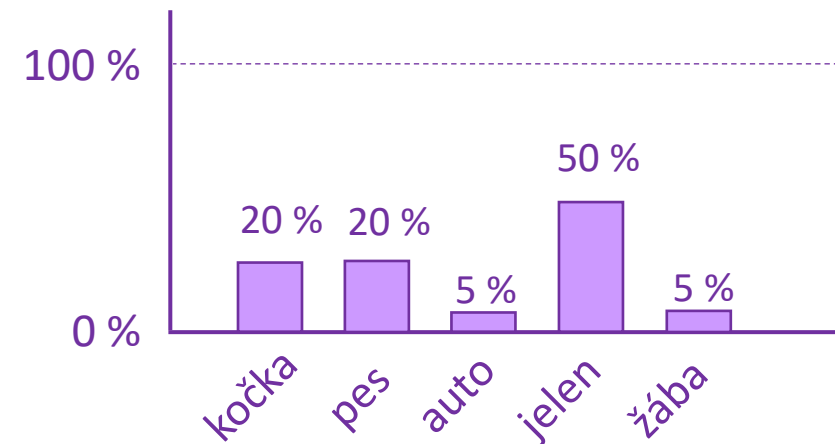
$$\mathbf{p}_n = [p_{n,1}, \dots, p_{n,K}]^T$$

cílové rozdělení (ground truth / target)



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

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  $K = 5 \rightarrow$  požadované rozdělení je

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

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

# Převod výstupních skóre modelu na rozdělení: softmax

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

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

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

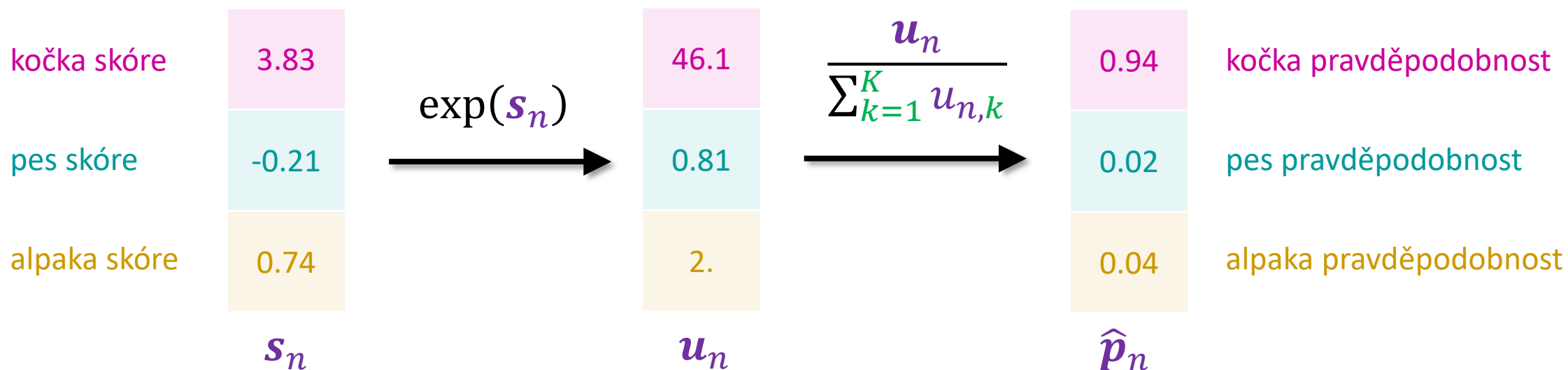
$$\hat{\mathbf{p}}_n = [\hat{p}_{n,1}, \dots, \hat{p}_{n,K}]^T, \quad 0 \leq \hat{p}_{n,k} \leq 1, \quad \sum_{k=1}^K \hat{p}_{n,k} = 1$$

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



# Softmax: příklad

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



# Softmax + cross entropy

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

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

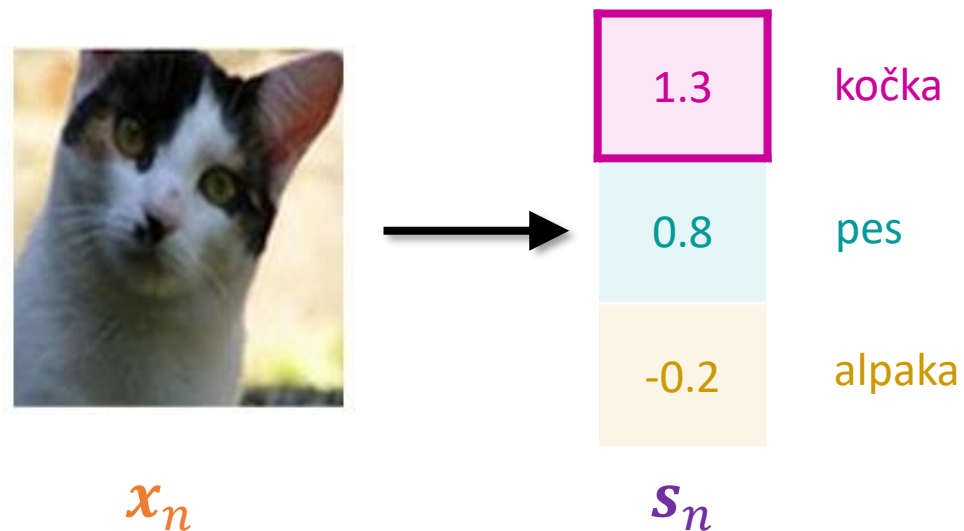
což je zápis, jenž najdeme např. v [poznámkách cs231n](#)

- 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

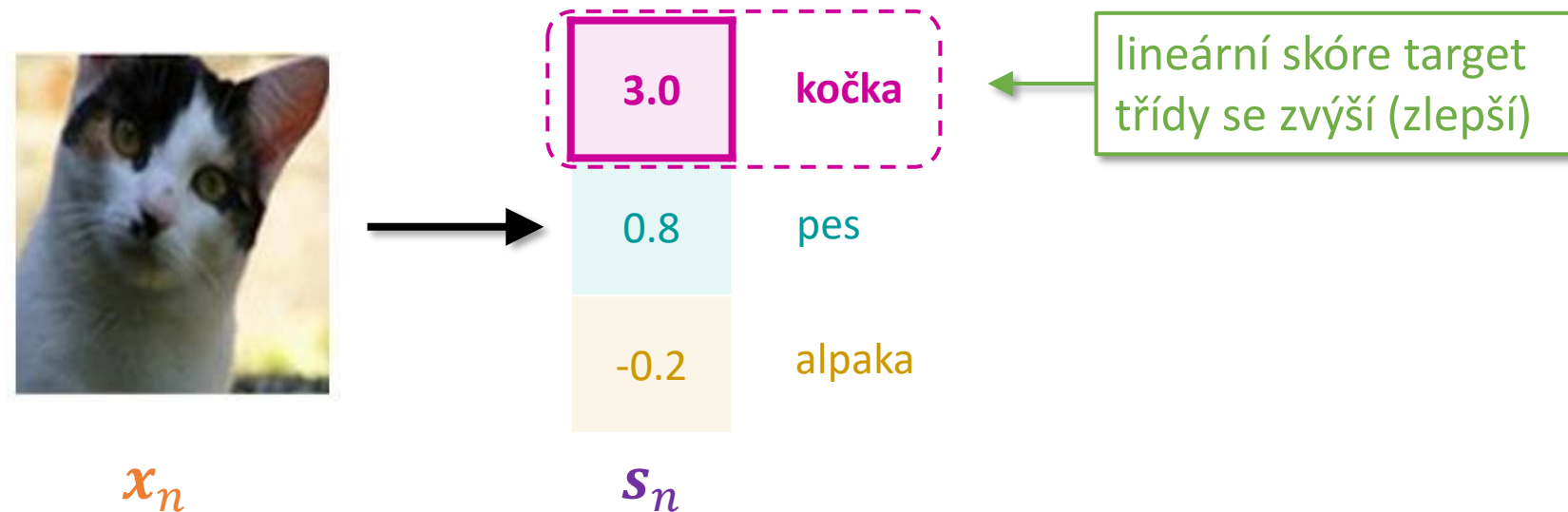
# Citlivost cross entropy na změnu skóre



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

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

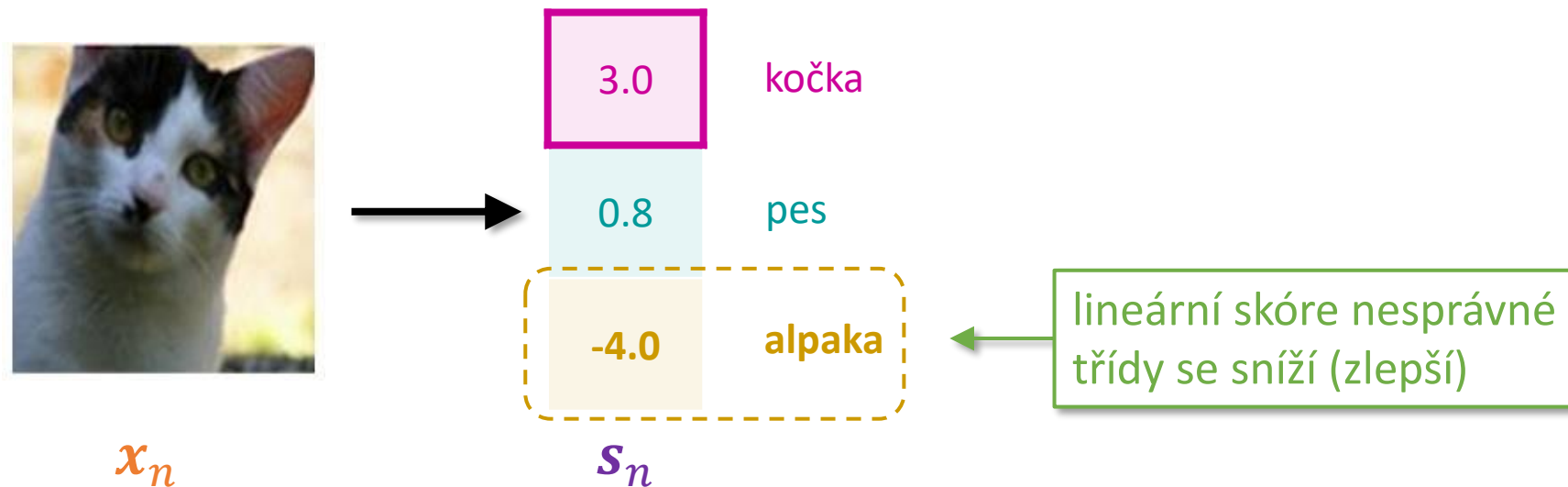
# Citlivost cross entropy na změnu skóre



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

softmax cross entropy se zmenší (zlepší)

# Citlivost cross entropy na změnu skóre



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

softmax cross entropy se opět zmenší (zlepší)

# 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)


L2 regularizace

---

# Vliv přeškálování parametrů

$$\mathbf{s}_n = \mathbf{w} \cdot \mathbf{x}_n$$
$$\hat{\mathbf{p}}_n = \frac{e^{\mathbf{s}_n}}{\sum_{k=1}^K e^{s_{n,k}}}$$

Matice parametrů  
vynásobená 10x



$$\begin{bmatrix} 0.21 \\ 0.09 \\ -0.40 \end{bmatrix} = \begin{bmatrix} 0.016 & -0.002 & 0.003 \\ 0.003 & -0.006 & 0.006 \\ -0.004 & -0.006 & -0.008 \end{bmatrix} \begin{bmatrix} 10 \\ 20 \\ 30 \end{bmatrix}$$
$$\begin{bmatrix} 0.41 \\ 0.36 \\ 0.22 \end{bmatrix} = \text{Softmax} \left( \begin{bmatrix} 0.21 \\ 0.09 \\ -0.40 \end{bmatrix} \right)$$

$$y_n = 0$$

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

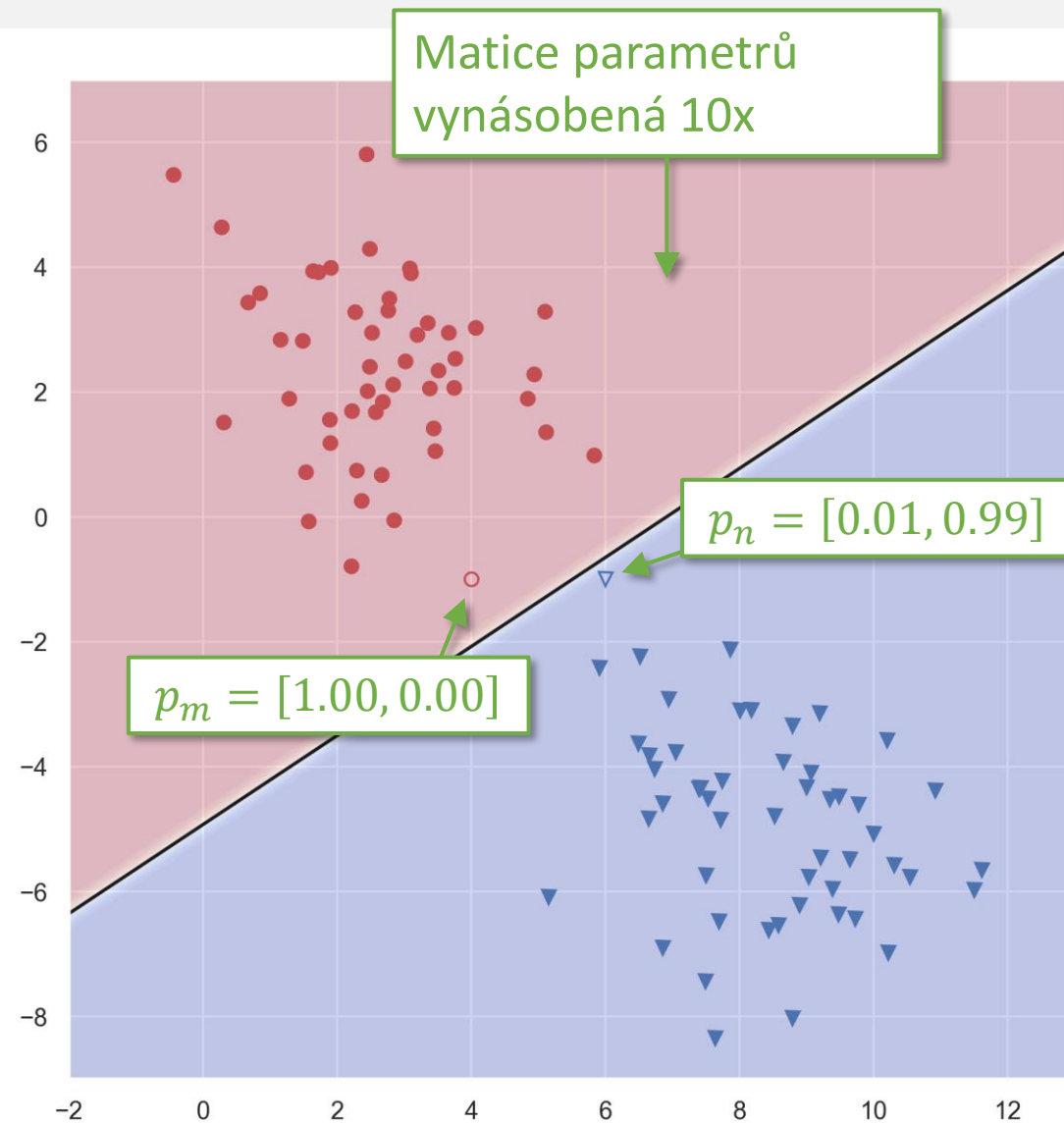
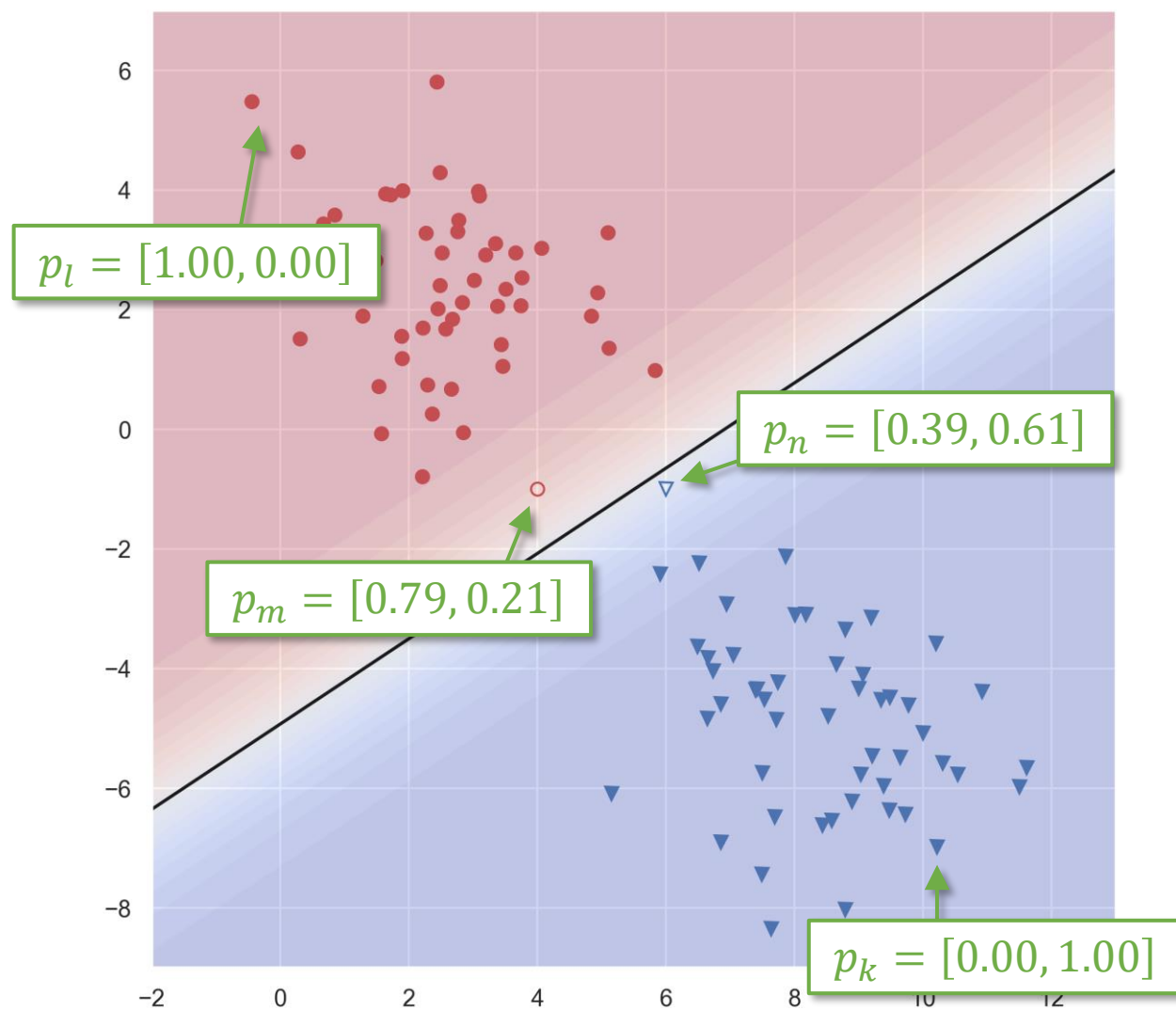
$$\begin{bmatrix} 2.05 \\ 0.90 \\ -4.00 \end{bmatrix} = \begin{bmatrix} 0.16 & -0.02 & 0.03 \\ 0.03 & -0.06 & 0.06 \\ -0.04 & -0.06 & -0.08 \end{bmatrix} \begin{bmatrix} 10 \\ 20 \\ 30 \end{bmatrix}$$
$$\begin{bmatrix} 0.76 \\ 0.24 \\ 0.00 \end{bmatrix} = \text{Softmax} \left( \begin{bmatrix} 2.05 \\ 0.90 \\ -4.00 \end{bmatrix} \right)$$

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

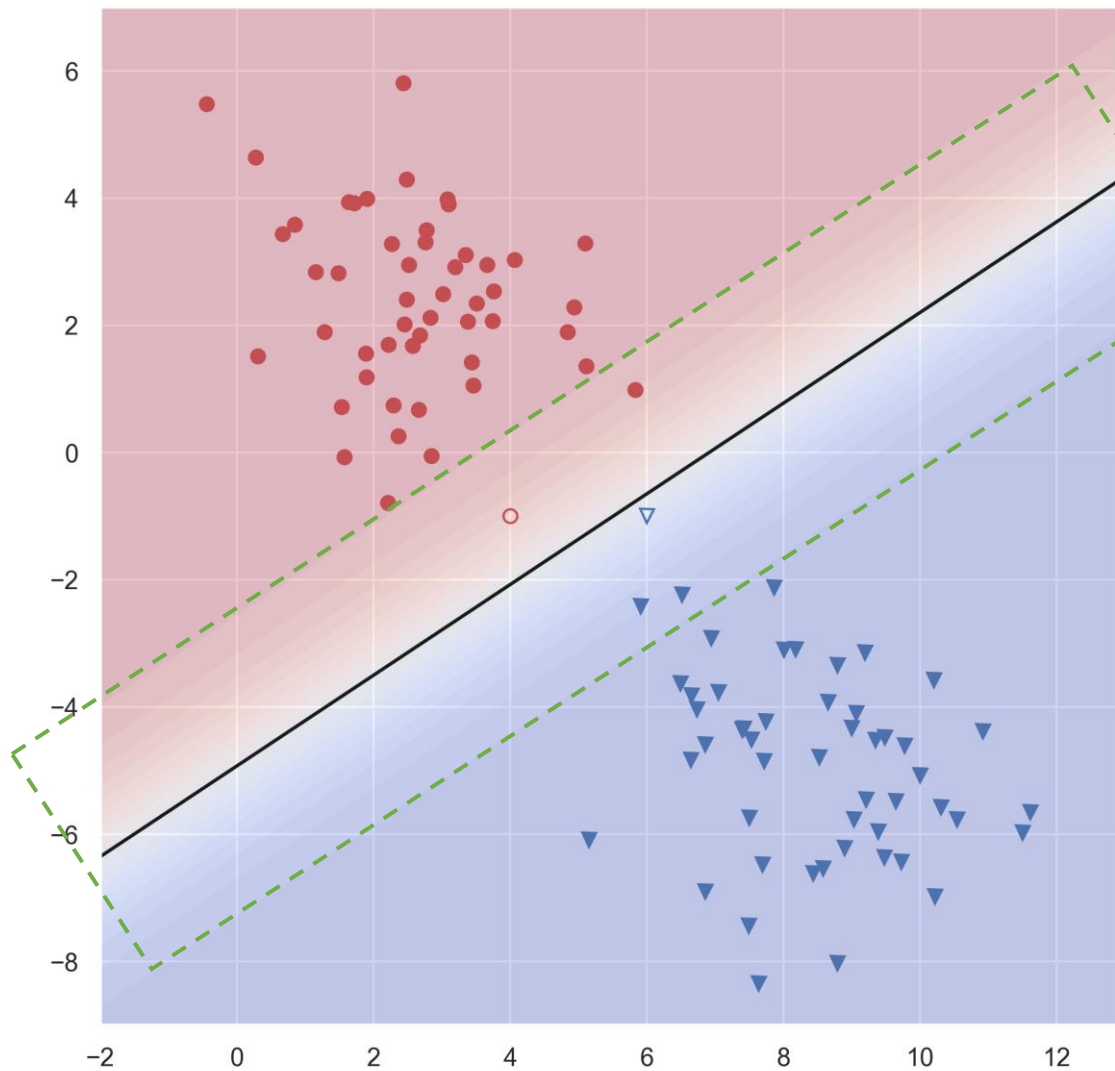
- Přeškálováním parametrů se zvýrazní rozdíly, ale nezmění znaménko skóre (logitů) ani pořadí pravděpodobností na výstupu, tj. klasifikátor predikuje stále stejně
- Hodnota kritéria (lossu) se ale přitom zmenší



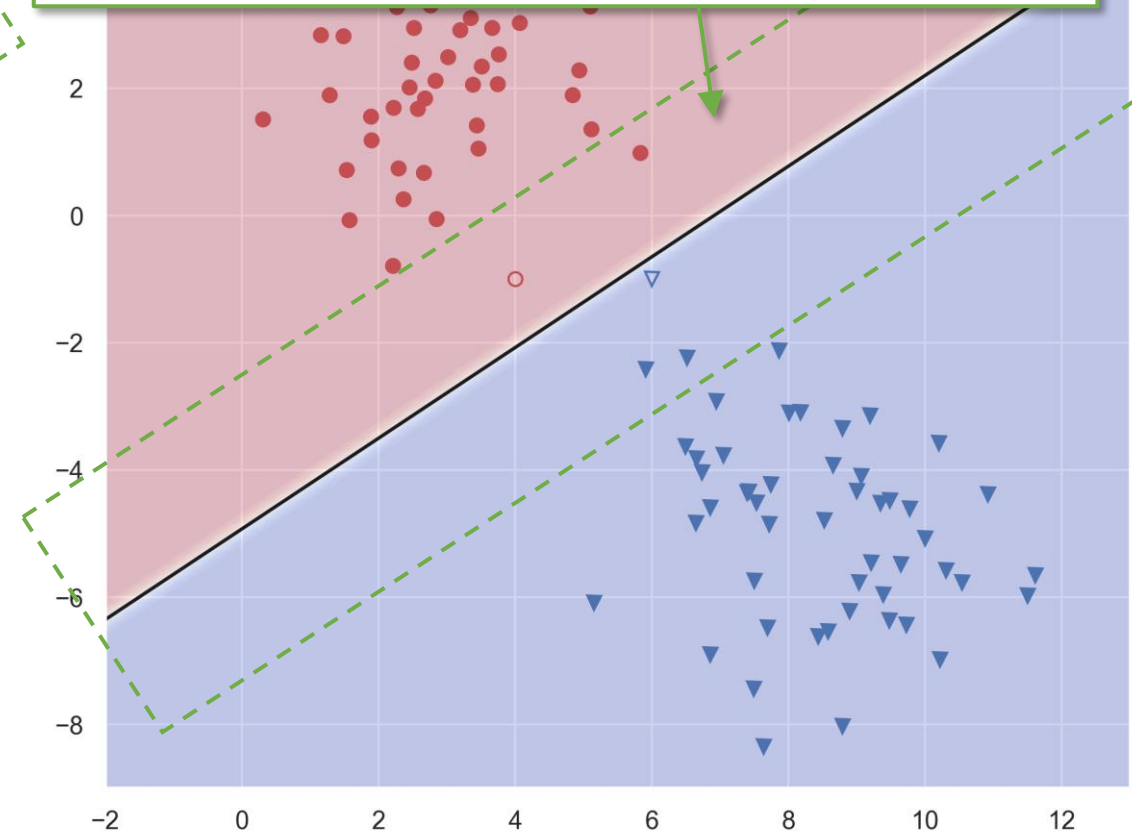
# Vliv přeškálování parametrů



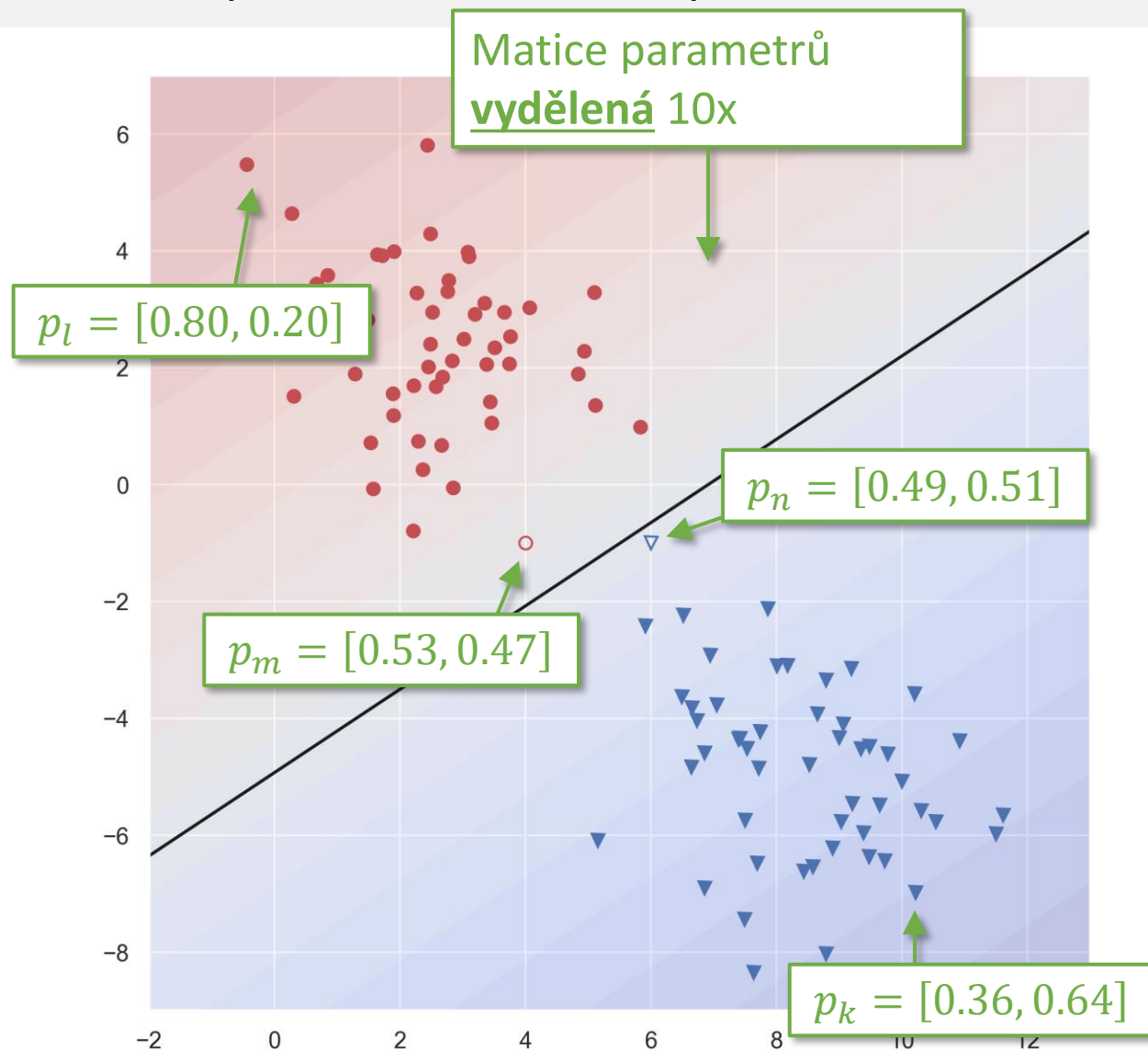
# Vliv přeskálování parametrů



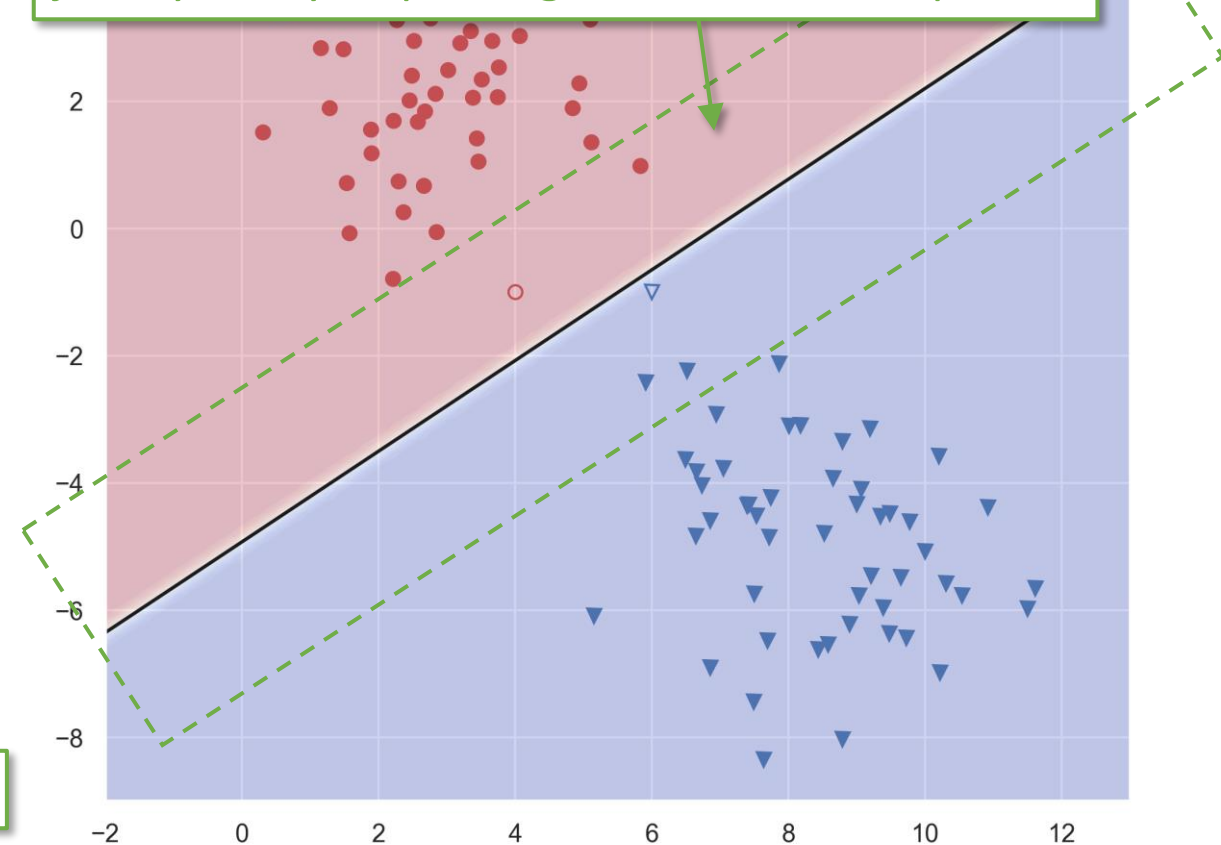
Přílišné “sebevědomí” v oblastech, kde jsou data řídká. Malá změna na vstupu potom znamená velké změny v predikcích (**vysoká variance**) a klasifikátor je tzv. **přeučten (overfit)**. Jen o trochu jiná data (např. test set) znamenají jiné výsledky = špatná generalizační schopnost.



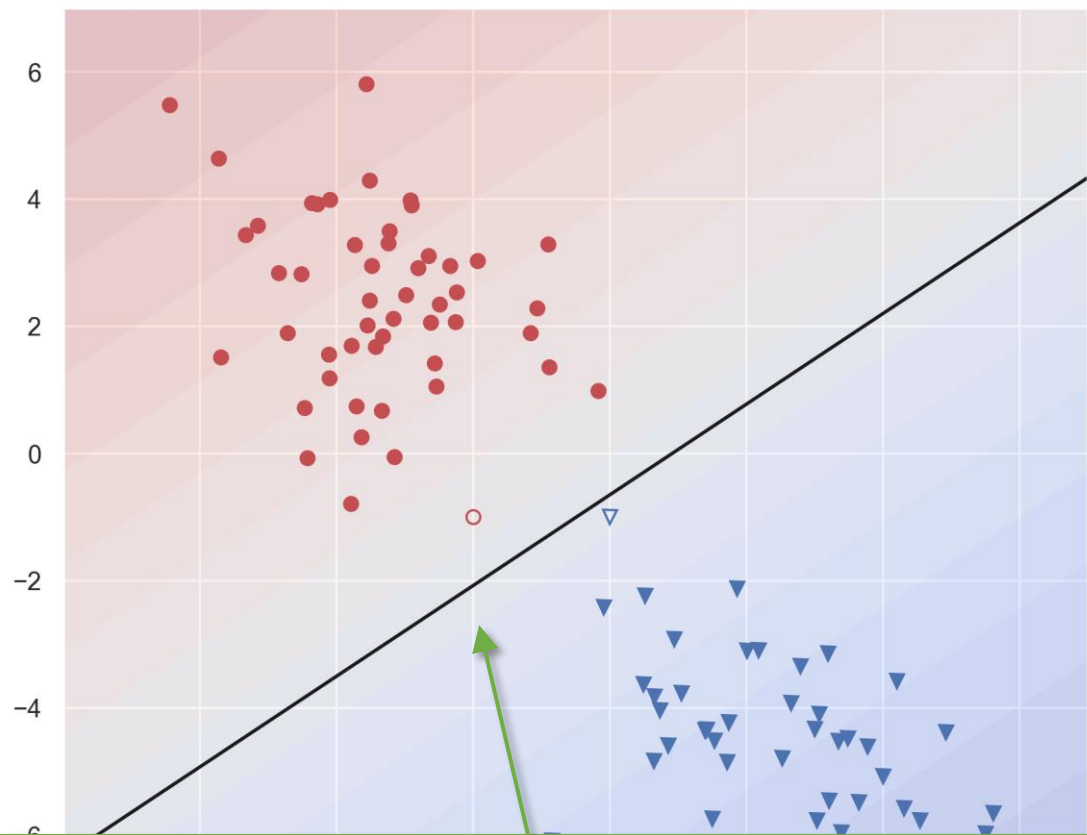
# Vliv přeškálování parametrů



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

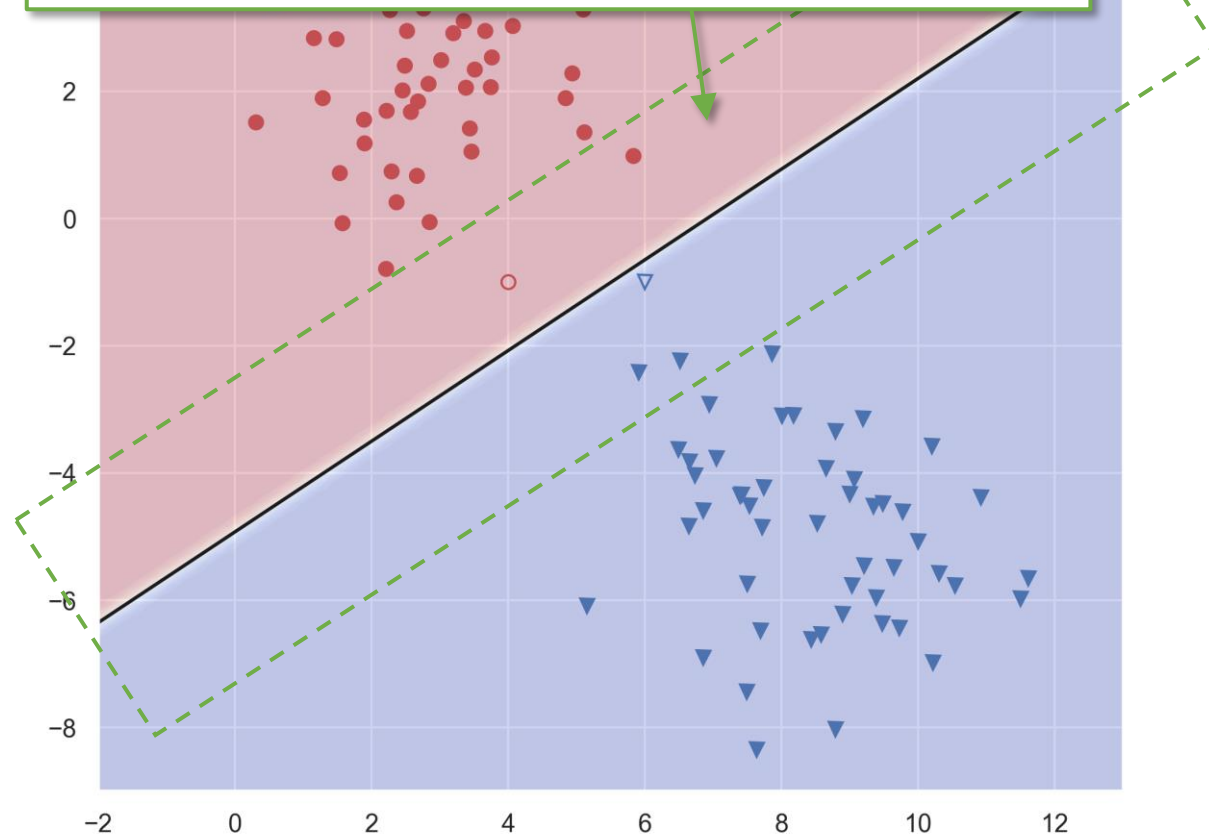


# Vliv přeskálování parametrů



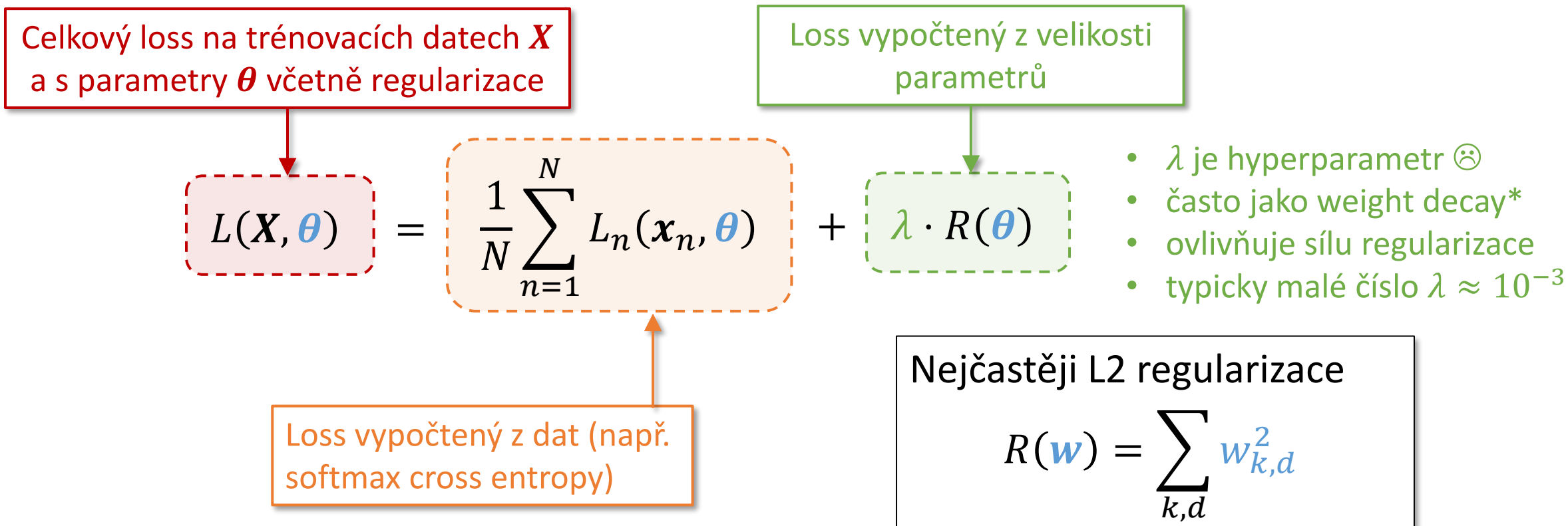
Přílišná “opatrnost” i v oblastech s reprezentativními daty znamená malé změny na výstupu i při velkých rozdílech ve vstupech (**vysoký bias**). Klasifikátor je tzv. **nedoučen (underfit)** a není schopen modelovat okrajové případy.

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



# Škálování vah aditivní regularizací

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



# Celkové kritérium včetně aditivní regularizace

- Včetně aditivní regularizace tedy budeme chybovost klasifikátoru posuzovat jako

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

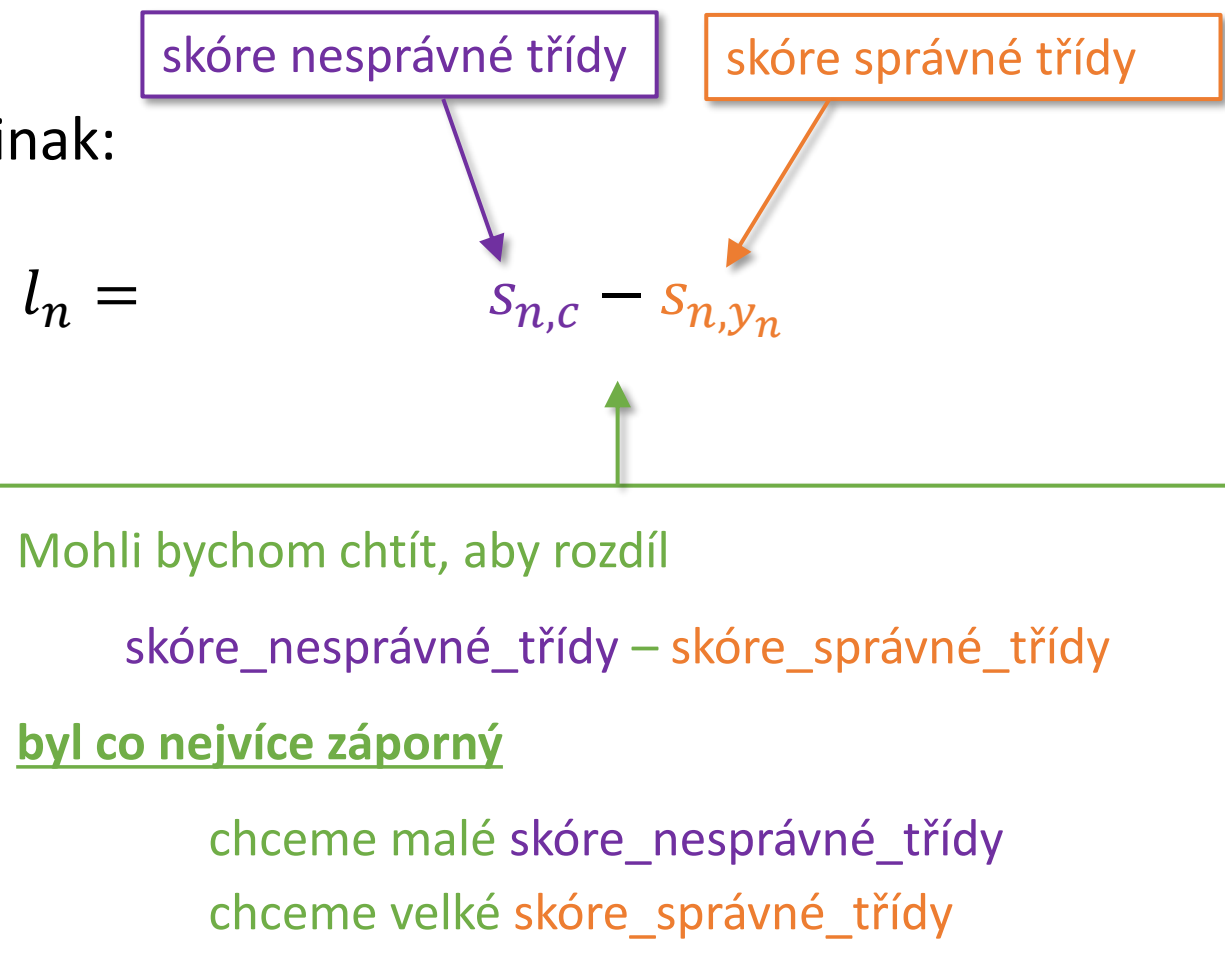
# Klasifikační kritérium

---

Support Vector Machine (SVM)

# Weston-Watkins multiclass hinge loss

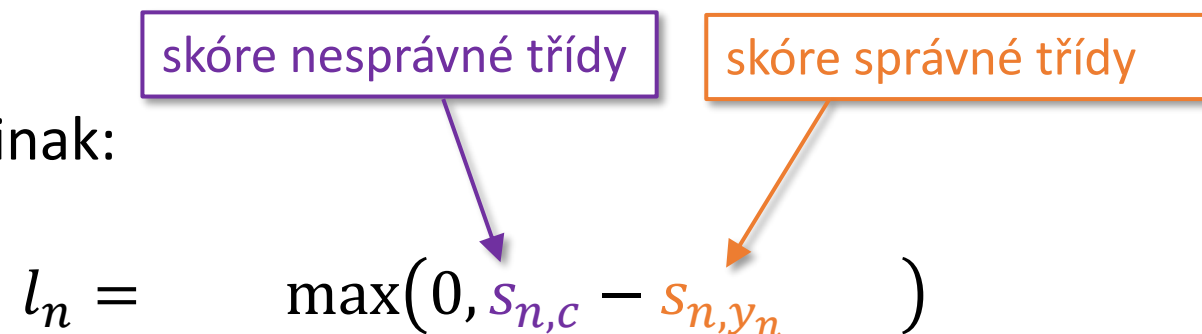
- Zkusme měřit data loss jinak:





# Weston-Watkins multiclass hinge loss

- Zkusme měřit data loss jinak:



The diagram illustrates the Weston-Watkins multiclass hinge loss formula. At the top, there are two boxes: a purple one labeled 'skóre nesprávné třídy' (scores of incorrect classes) and an orange one labeled 'skóre správné třídy' (scores of correct classes). A purple arrow points from the first box to the term  $s_{n,c}$  in the formula, and an orange arrow points from the second box to the term  $s_{n,y_n}$ . The formula itself is  $l_n = \max(0, s_{n,c} - s_{n,y_n})$ . A green arrow points from a text box below to the entire formula.

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

V rámci prevence overfitu nám ale bude stačit, když

$$\text{skóre\_správné\_třídy} > \text{skóre\_nesprávné\_třídy}$$

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

- Zkusme měřit data loss jinak:

The diagram illustrates the Weston-Watkins multiclass hinge loss formula. At the top, there are two boxes: a purple box on the left labeled "skóre nesprávné třídy" (scores of incorrect classes) and an orange box on the right labeled "skóre správné třídy" (scores of correct classes). A purple arrow points from the purple box to the variable  $s_{n,c}$  in the formula, and an orange arrow points from the orange box to the variable  $s_{n,y_n}$ . The formula itself is  $l_n = \max(0, s_{n,c} - s_{n,y_n} + \Delta)$ . A green arrow points from a text box below to the expression  $s_{n,c} - s_{n,y_n} + \Delta$  in the formula.

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

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

$$\text{skóre\_správné\_třídy} > \text{skóre\_nesprávné\_třídy} + \Delta$$

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:

The diagram illustrates the Weston-Watkins multiclass hinge loss formula. At the top, two boxes are present: a purple box on the left labeled "skóre nesprávné třídy" (scores of incorrect classes) and an orange box on the right labeled "skóre správné třídy" (scores of correct classes). A purple arrow points from the purple box to the variable  $s_{n,c}$  in the formula, and an orange arrow points from the orange box to the variable  $s_{n,y_n}$ . The formula itself is  $l_n = \max(0, s_{n,c} - s_{n,y_n} + 1)$ . Below the formula, a green box contains text explaining the margin: "Margin  $\Delta$  se obvykle nastavuje na hodnotu 1, lze ho totiž nahradit přeškálováním skóre - a to lze "šťelovat" regularizací." A green arrow points from this box to the constant "1" in the formula.

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

Margin  $\Delta$  se obvykle nastavuje na hodnotu 1, lze ho totiž nahradit přeškálováním skóre - a to lze "šťelovat" regularizací.

# Weston-Watkins multiclass hinge loss

- Zkusme měřit data loss jinak:

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

skóre nesprávné třídy

skóre správné třídy

Podmínka dostatečného rozdílu by měla platit pro všechny nesprávné třídy

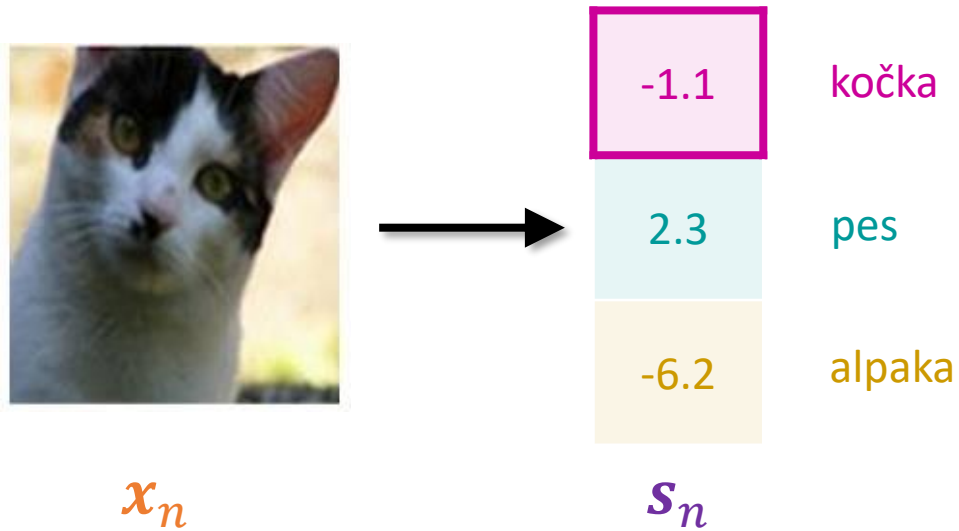
# Weston-Watkins multiclass hinge loss

- Zkusme měřit data loss jinak:

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

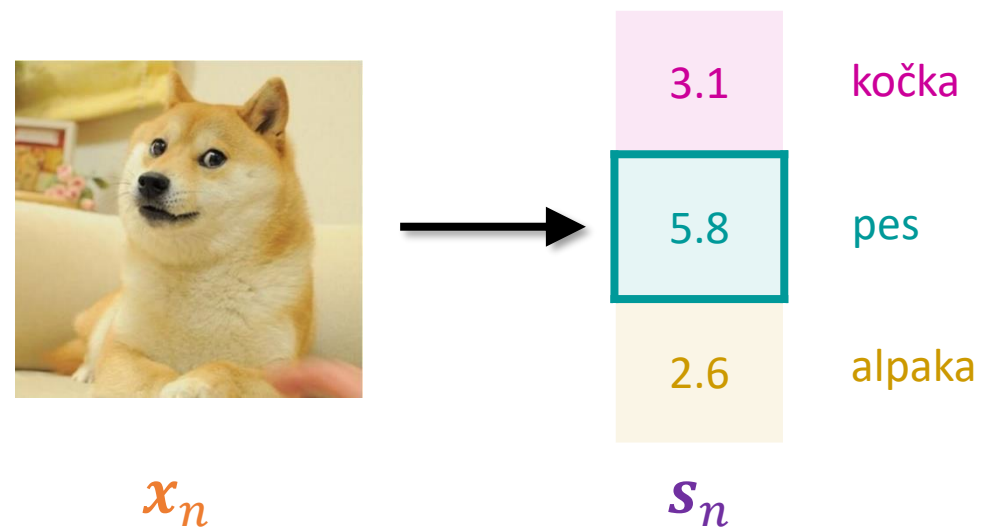
finální podoba hinge loss

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



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

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

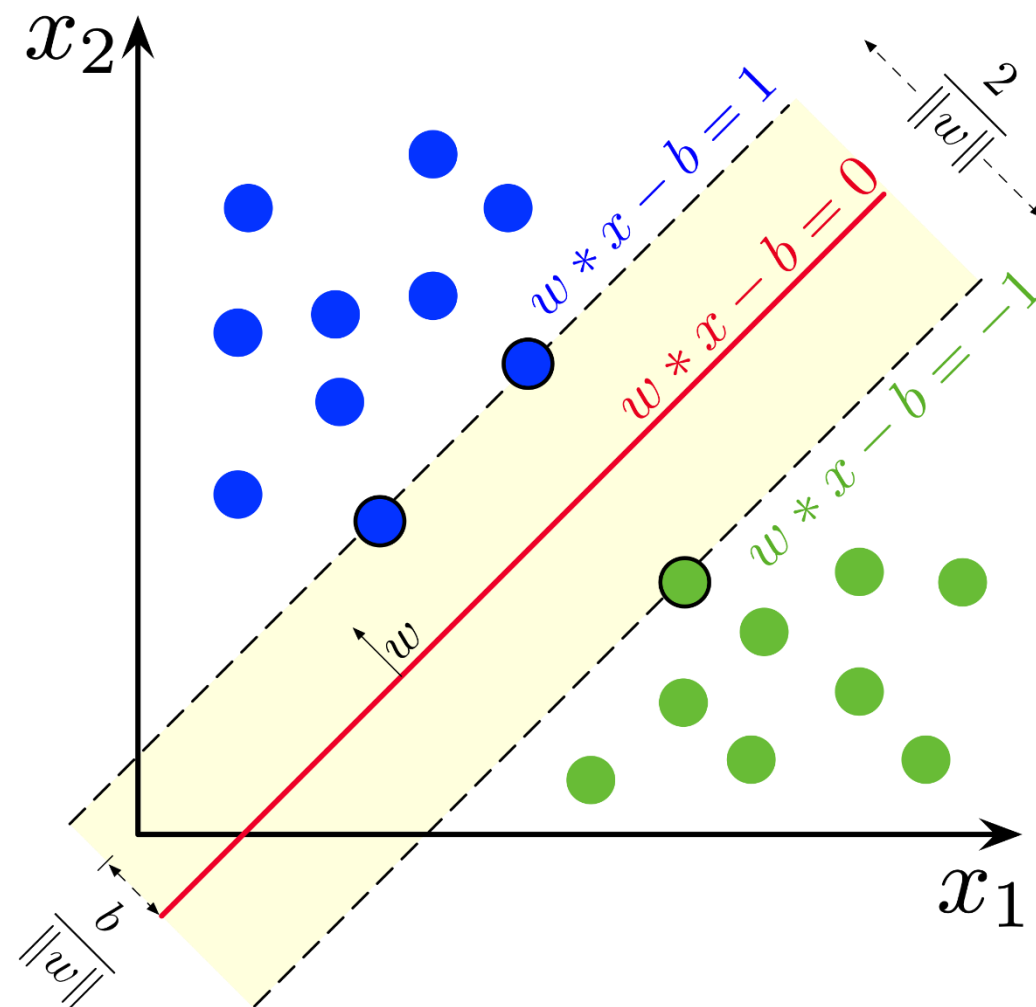


$$l_n = \sum_{c \in \{\text{kočka}, \text{alpaka}\}} \max(0, s_{n,c} - s_{n,y_n} + 1)$$

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

# Hard Margin Support Vector Machine (SVM)

- Multiclass hinge loss se snaží, aby každý bod byl přemapován na vzdálenost alespoň  $\Delta$  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  $\rightarrow$  pokud jsou třídy lineárně separovatelné, vždy najde optimální řešení

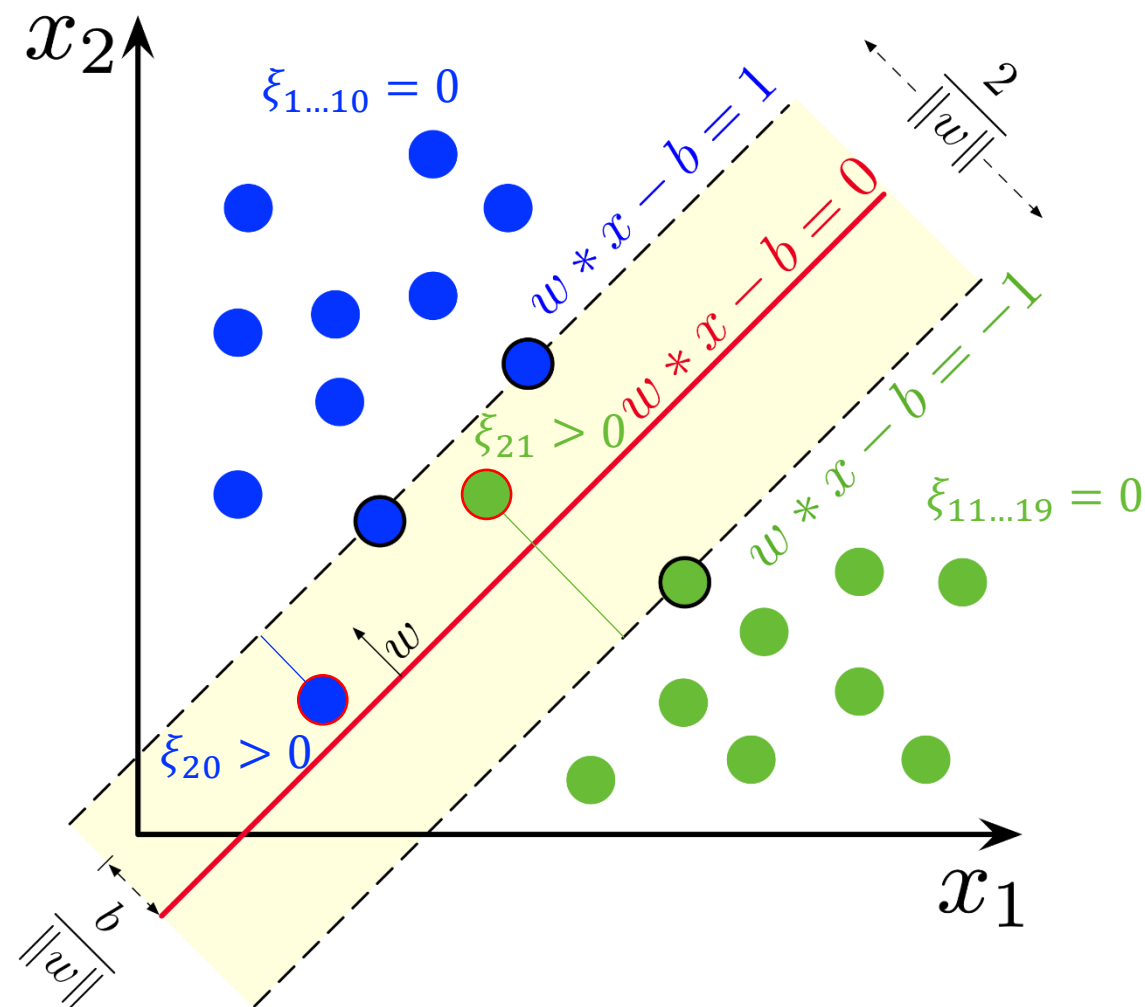


# Soft Margin Support Vector Machine (SVM)

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

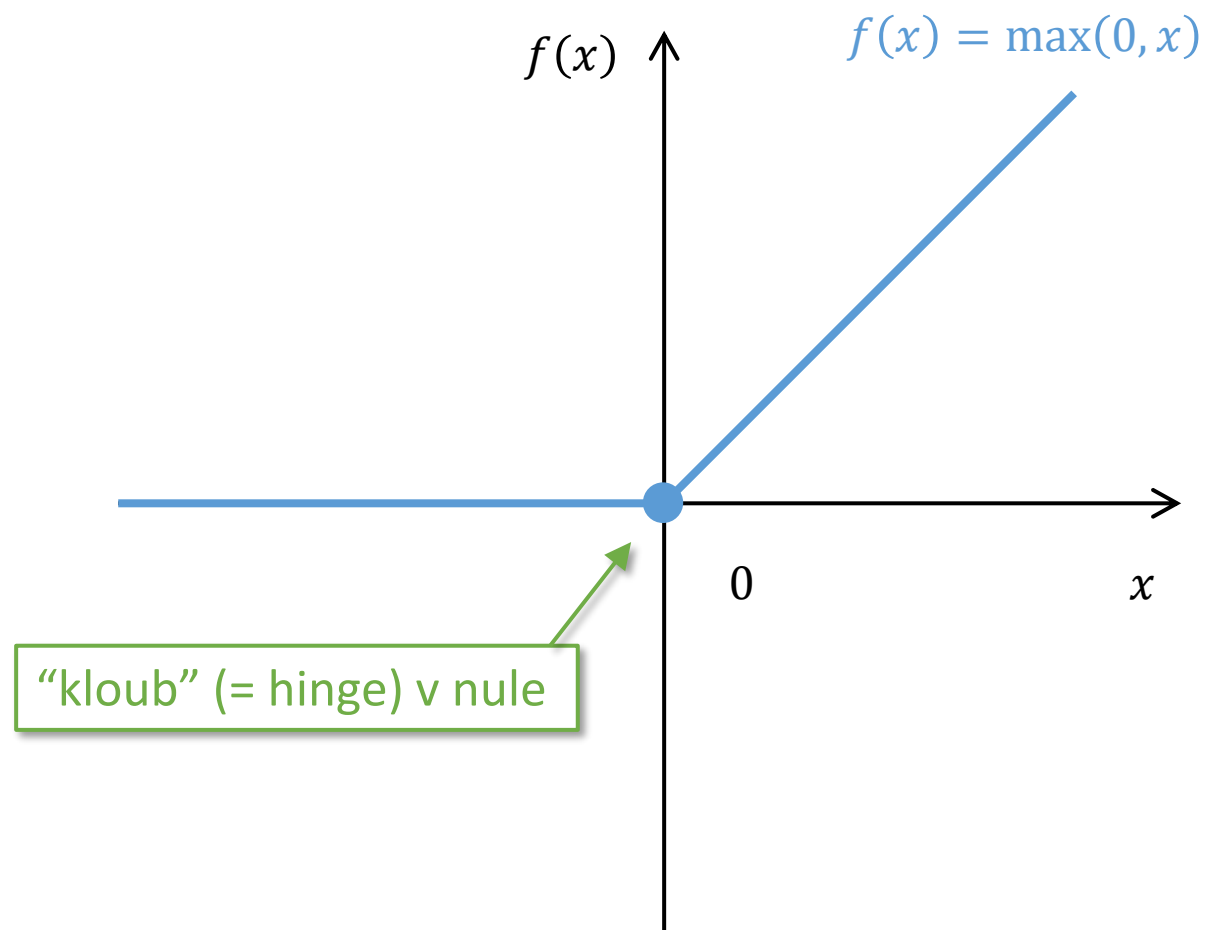
$$\xi_n = \max(0, s_{n,c} - s_{n,y_n} + 1)$$

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

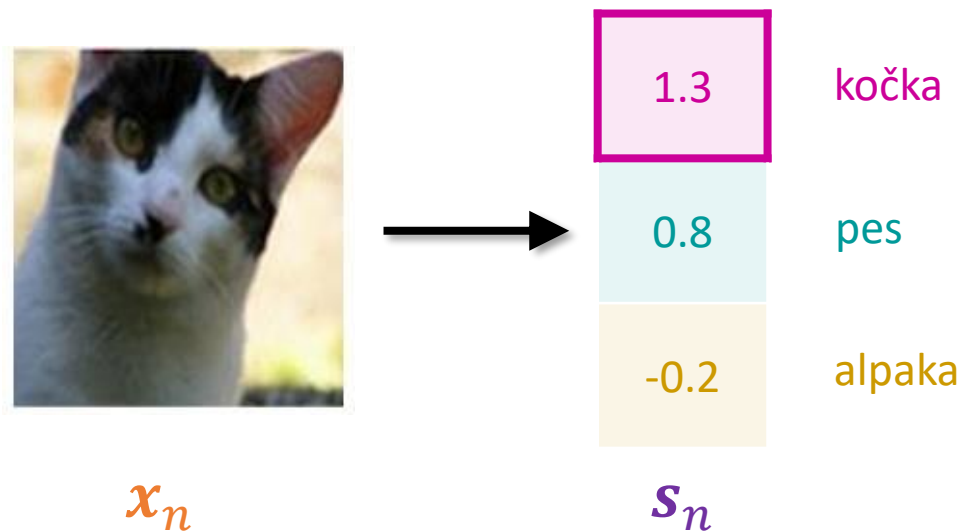




# Proč název hinge loss?

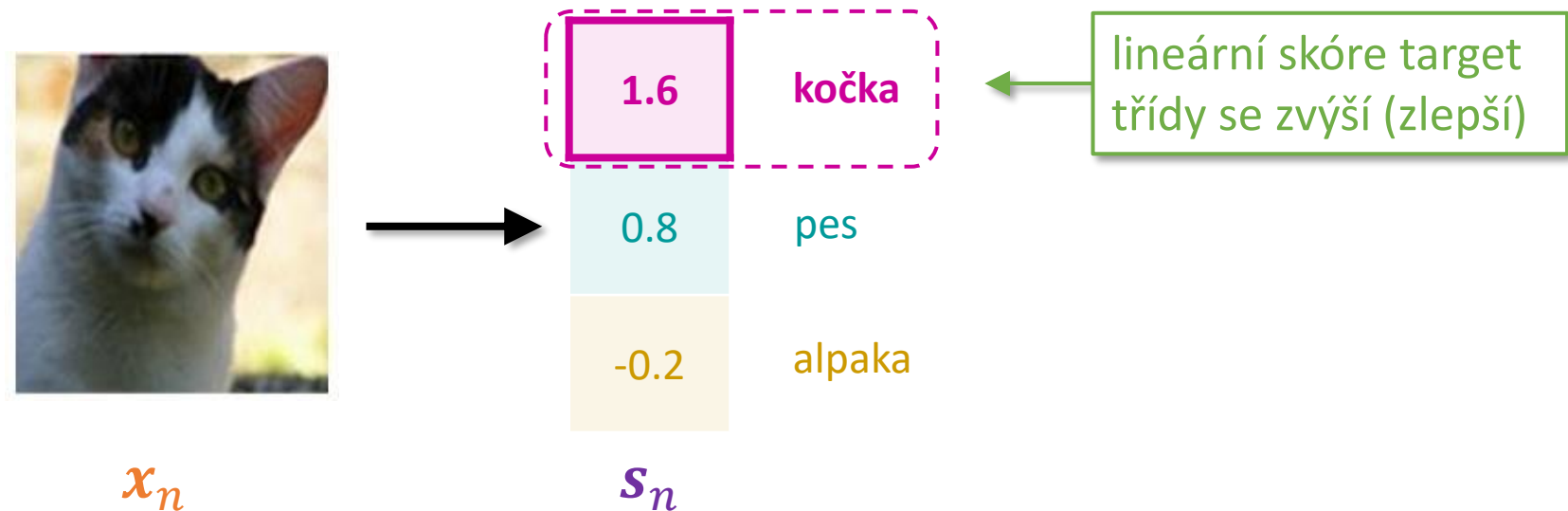


# Citlivost hinge lossu na změnu skóre



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

# Citlivost hinge lossu na změnu skóre

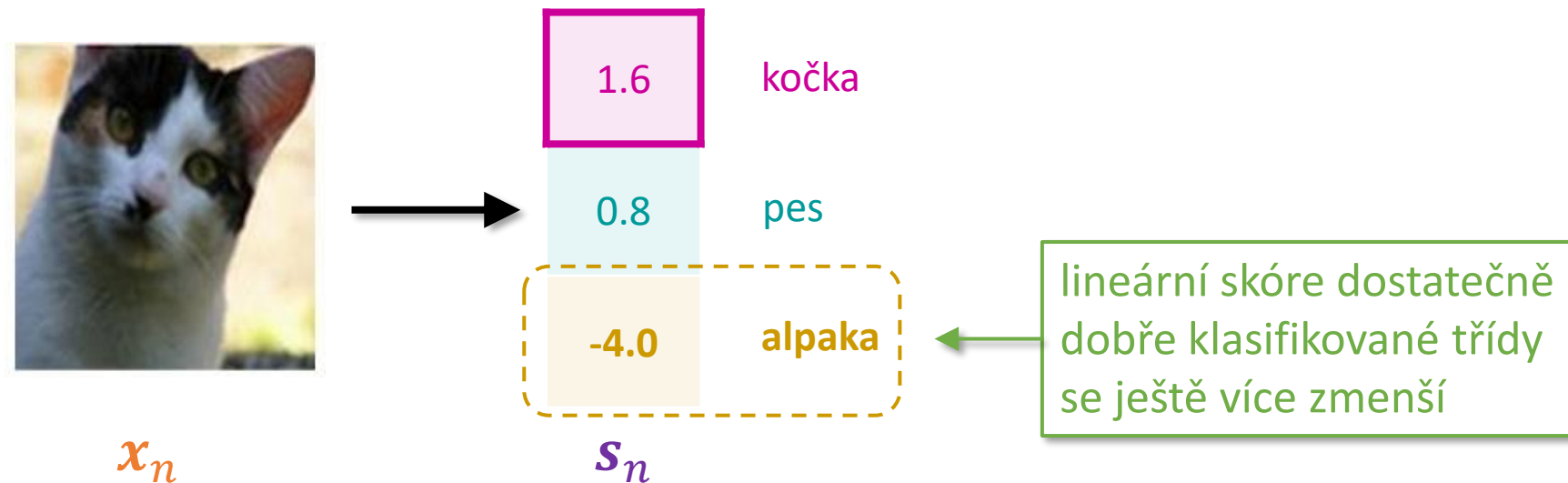


$$\begin{aligned} l_n &= \max(0, 0.8 - 1.6 + 1) + \max(0, -0.2 - 1.6 + 1) \\ &= \max(0, 0.2) + \max(0, -0.8) \\ &= 0.2 + 0 \\ &= 0.2 \end{aligned}$$

hinge loss se zmenší (zlepší)

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

# Citlivost hinge lossu na změnu skóre



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

← hinge loss už se nesníží

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

# 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”

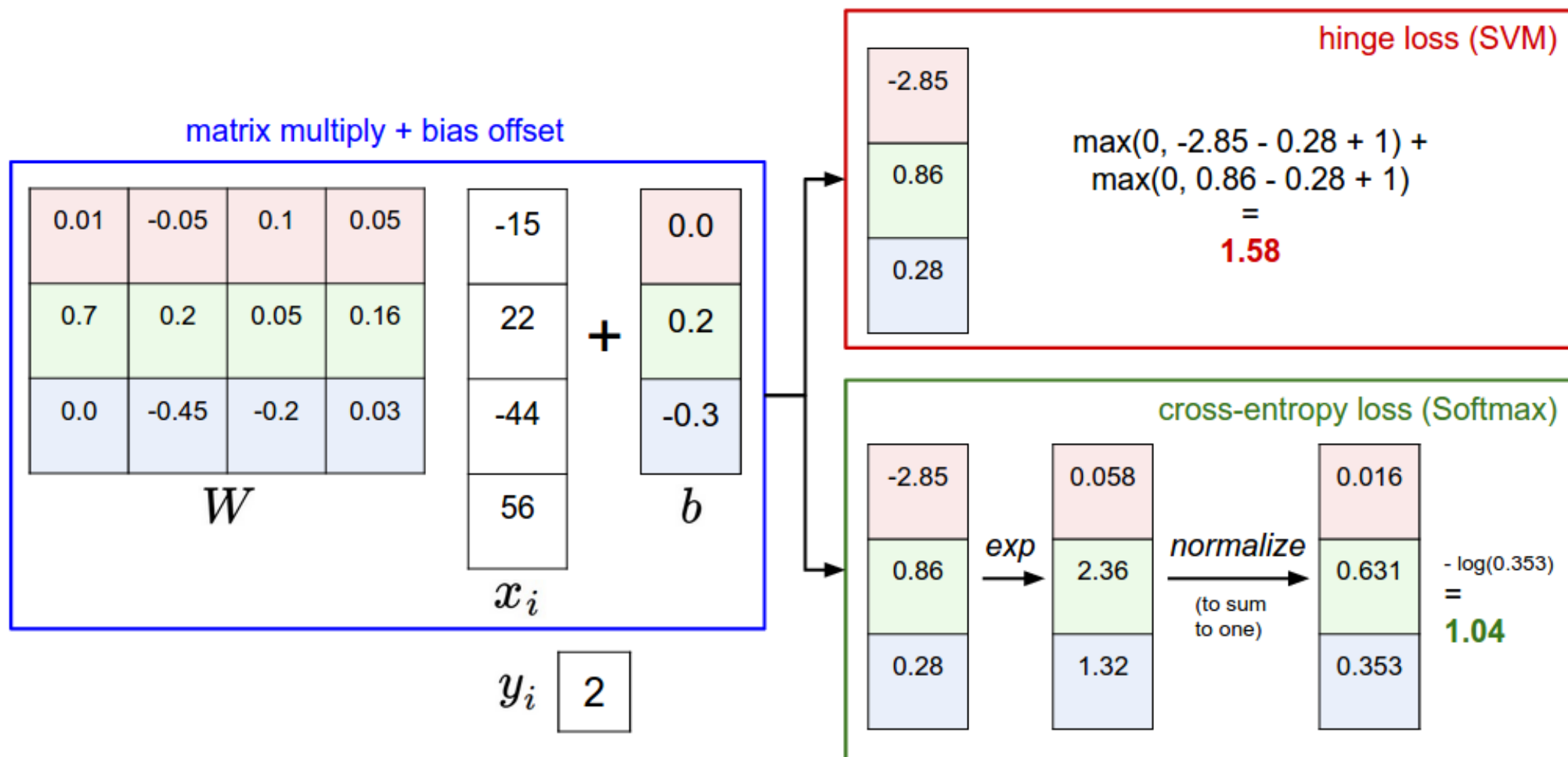
neární skóre dostatečně  
obře klasifikované třídy  
e ještě více zmenší

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

# Závěr

---

# Lineární model a klasifikační loss



# 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 klasifikátoru

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

