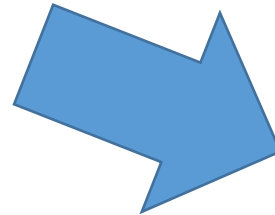
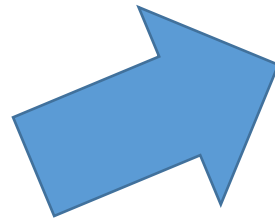


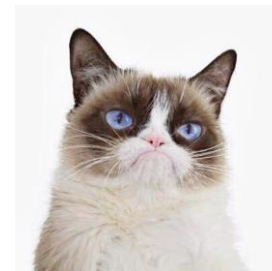
Aplikace neuronových sítí

Lineární klasifikace, Softmax, SVM

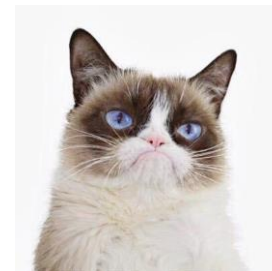
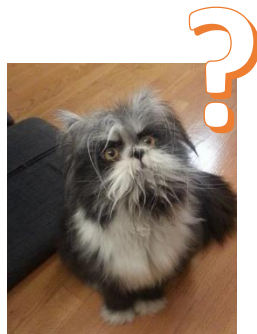
Problém klasifikace



Vzorové obrázky neboli trénovací data



Predikce na neznámém obrázku



Metoda nejbližšího souseda

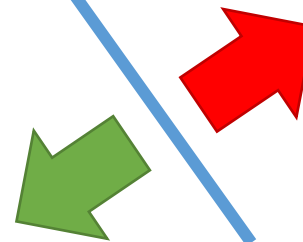
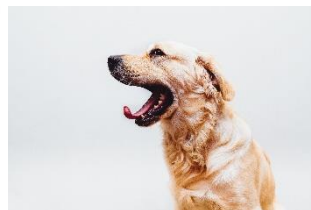
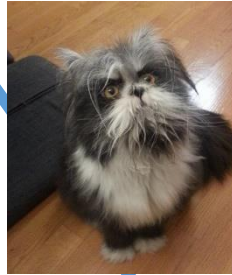
nejkratší vzdálenost je ke kočce → na obrázku je kočka



spočítáme vzdálenosti ke všem

Lineární klasifikace

obrázek je na straně koček → na obrázku je kočka



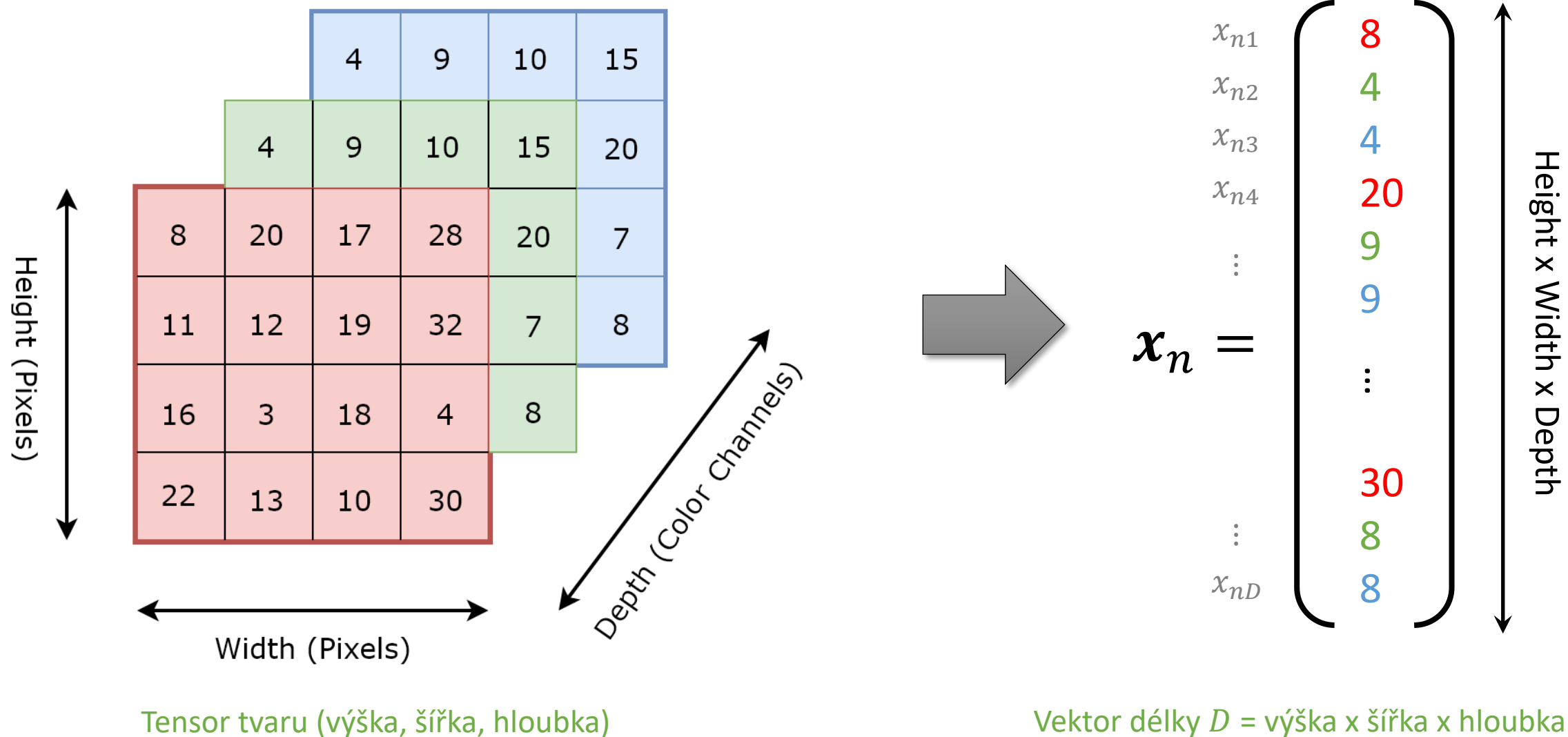
prostor přímkou (rovinou) rozdělíme
na dvě poloviny; na jedné straně
jsou psi, na druhé kočky

Jak vybereme optimální lineární klasifikátor?

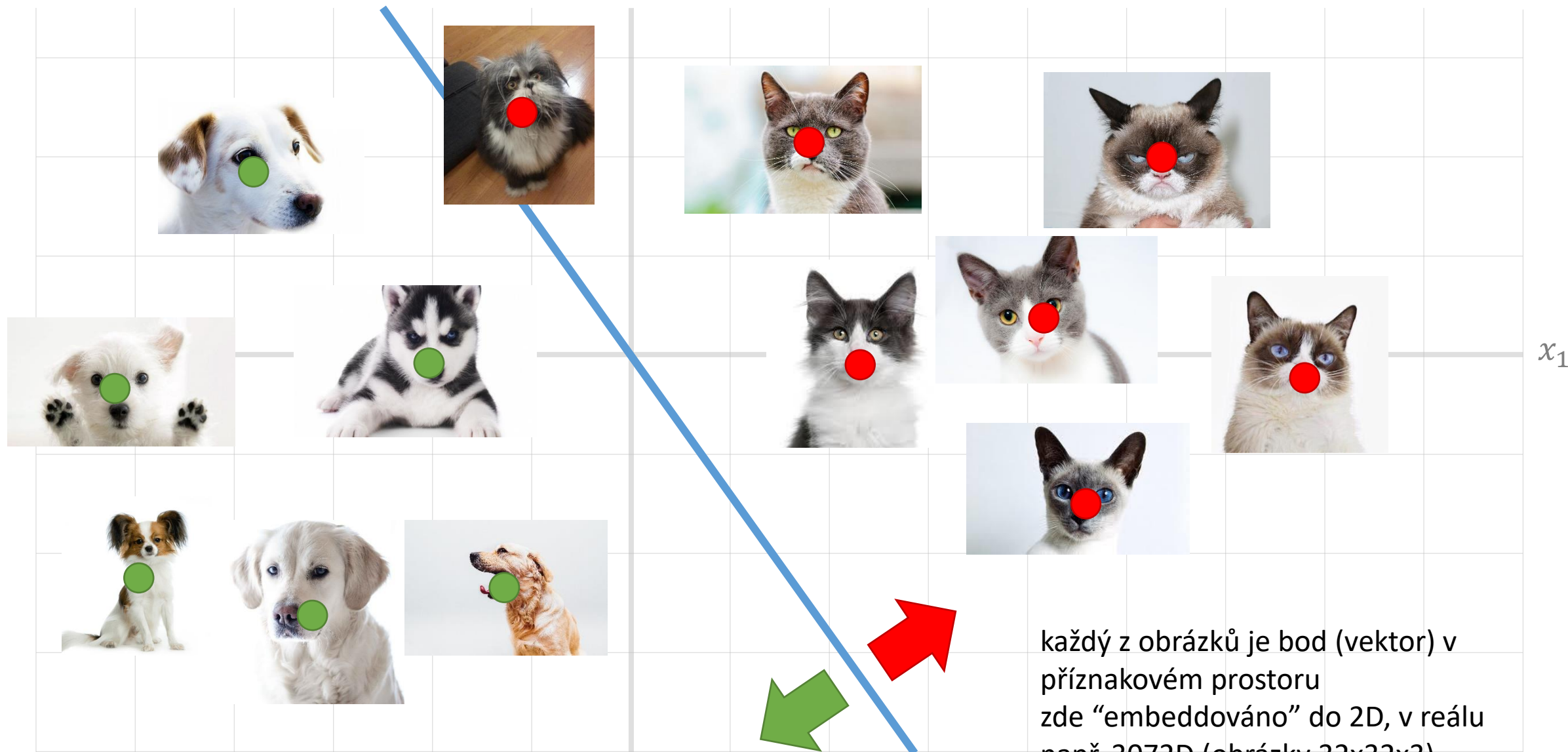
USU v kostce:

1. Vytvoříme diskriminativní klasifikátor s upravitelnými parametry
2. Kvantifikujeme jeho úspěšnost klasifikace nějakým kritériem
3. Upravujeme parametry a poznamenáváme si jejich výsledek (hodnotu kritéria)
4. Jako nejlepší vybereme takové parametry, které optimalizují zvolené kritérium

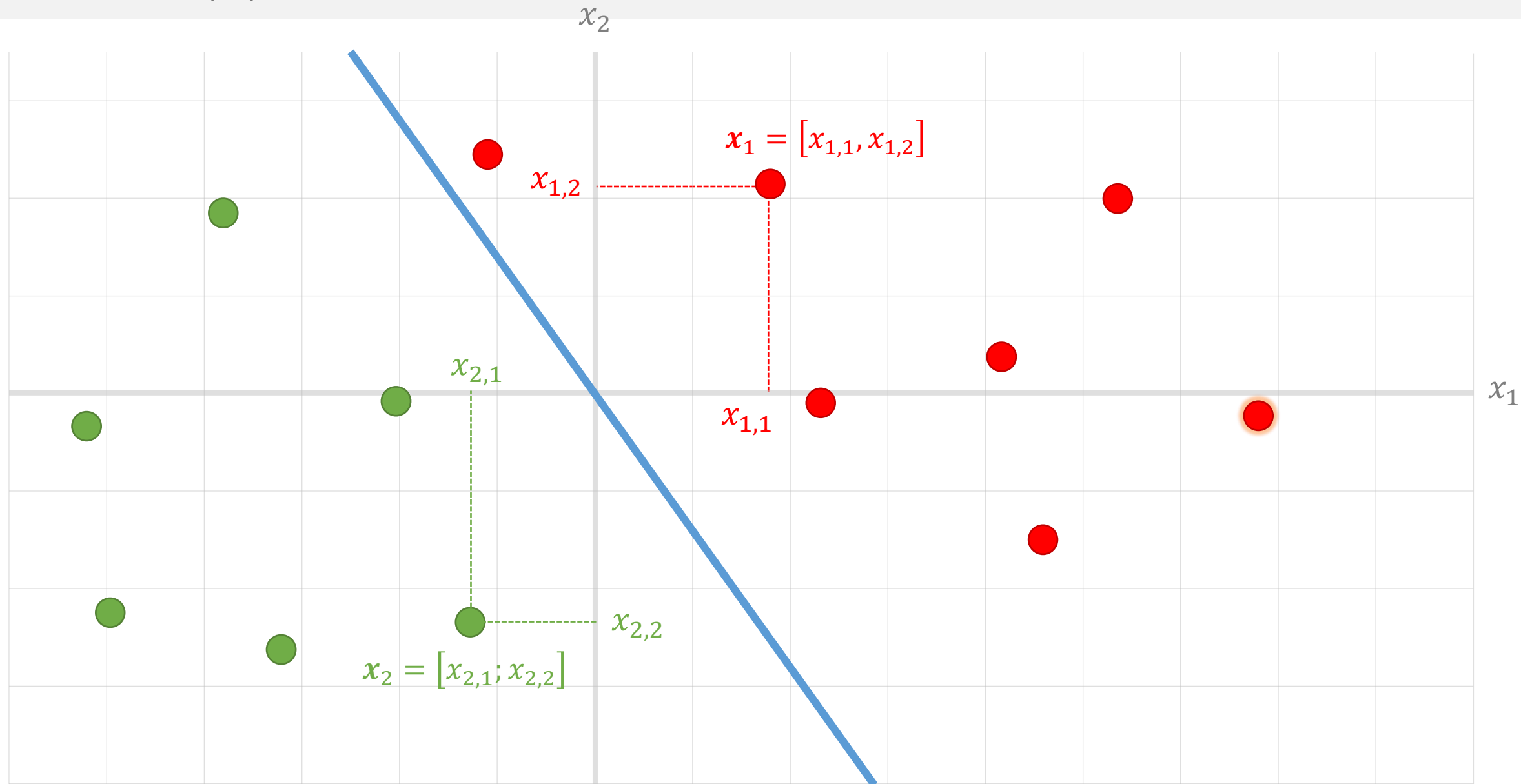
Reprezentace RGB obrázku jako vektoru



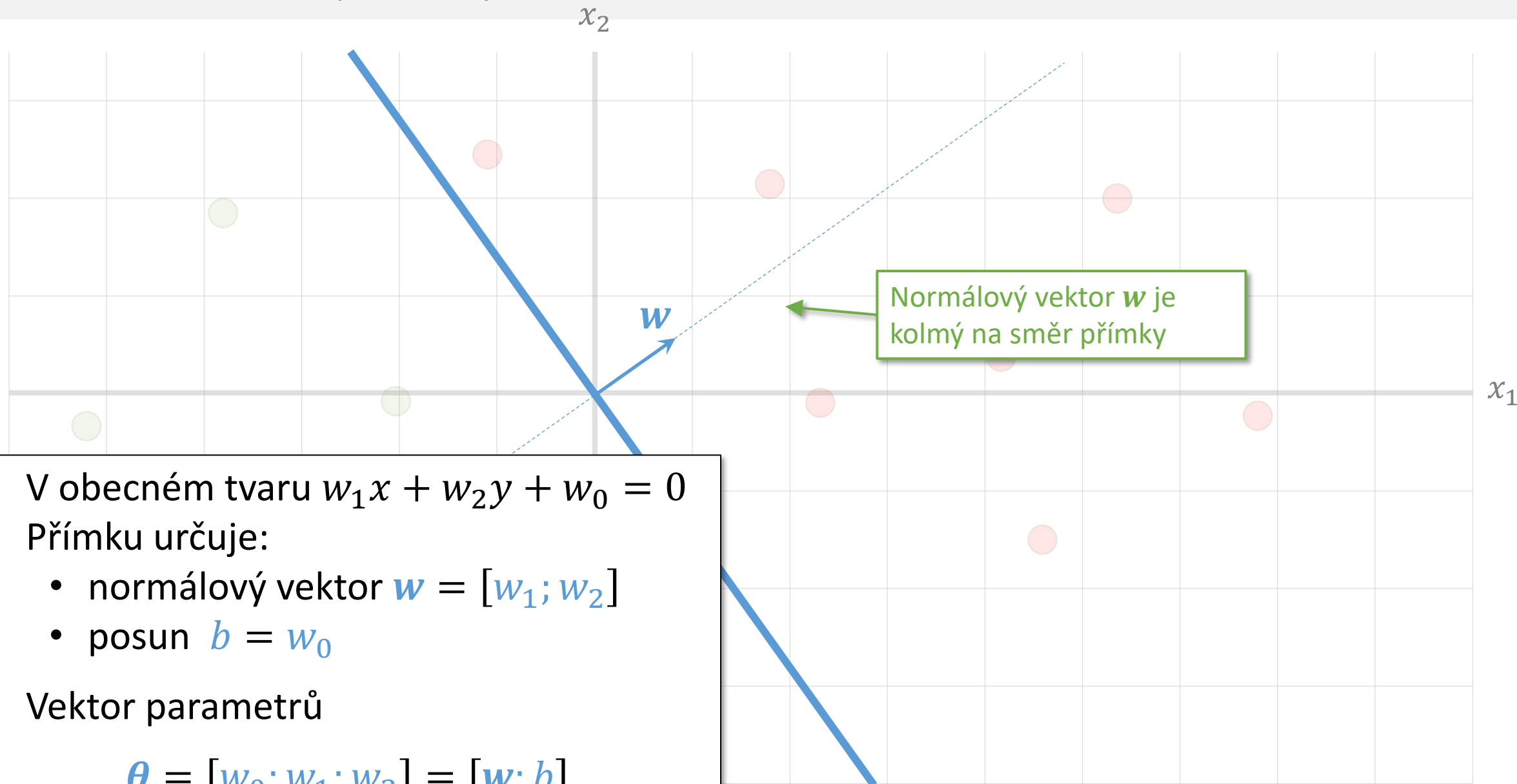
Příznakový prostor

 x_2 

Příznakový prostor



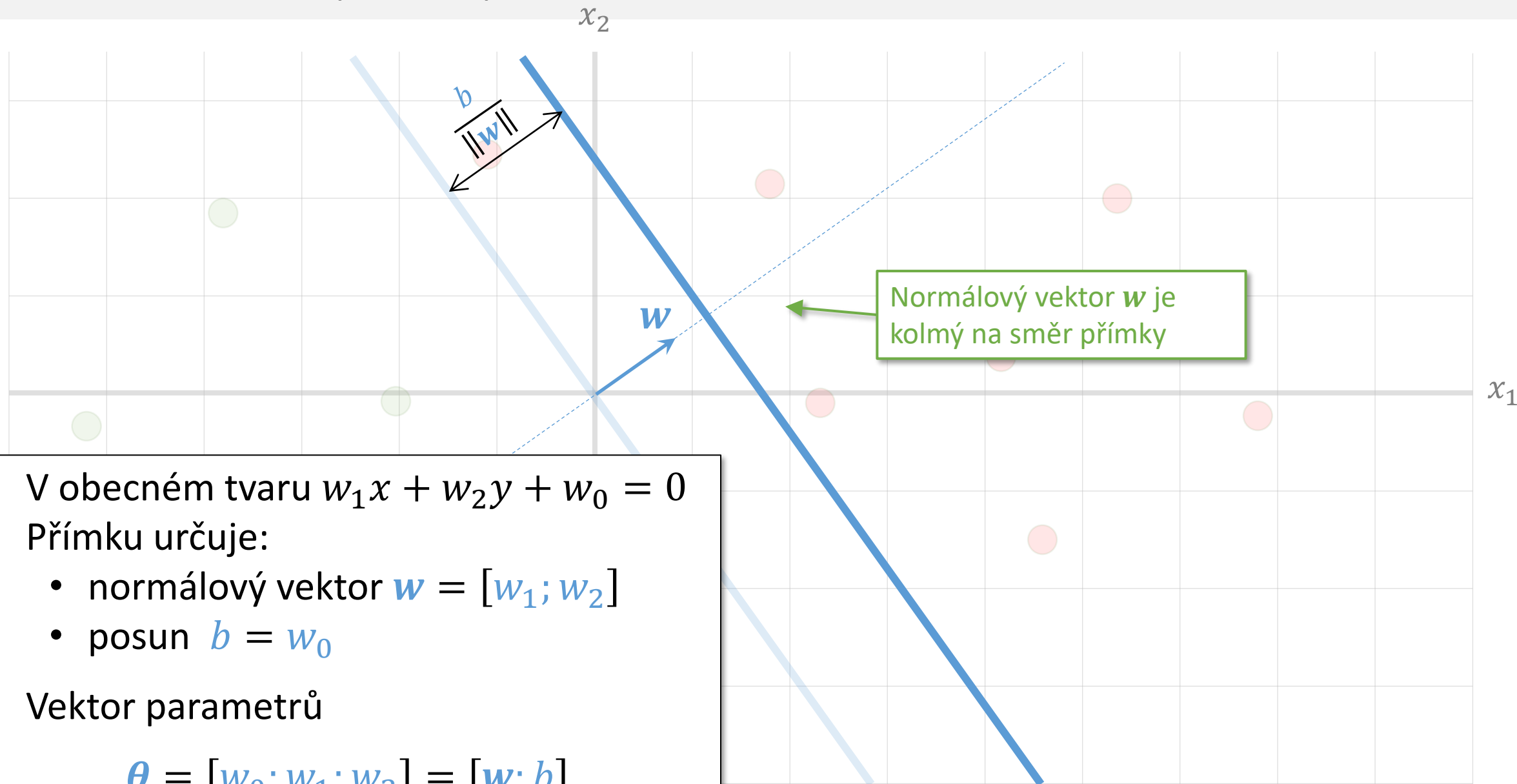
Parametrizace přímky



- V obecném tvaru $w_1x + w_2y + w_0 = 0$
- Přímku určuje:
 - normálový vektor $w = [w_1; w_2]$
 - posun $b = w_0$
- Vektor parametrů

$$\theta = [w_0; w_1; w_2] = [w; b]$$

Parametrizace přímky



- V obecném tvaru $w_1x + w_2y + w_0 = 0$
- Přímku určuje:
 - normálový vektor $w = [w_1; w_2]$
 - posun $b = w_0$
- Vektor parametrů

$$\theta = [w_0; w_1; w_2] = [w; b]$$

Na jaké straně bod je? Skalární součin vektorů

skalární součin

$$\mathbf{w}^T \mathbf{x} = \sum_i w_i x_i$$

vzdálenost bodu od přímky

$$\begin{aligned} d(\mathbf{w}, \mathbf{x}) &= \frac{\mathbf{w}^T \mathbf{x} + b}{\|\mathbf{w}\|} \\ &= \frac{w_1 x_1 + \dots + w_D x_D + b}{\sqrt{w_1^2 + \dots + w_D^2}} \end{aligned}$$

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

$$\mathbf{w} = [w_1; w_2], b = 0$$

$$\mathbf{x}_n = [x_{n1}; x_{n2}]$$

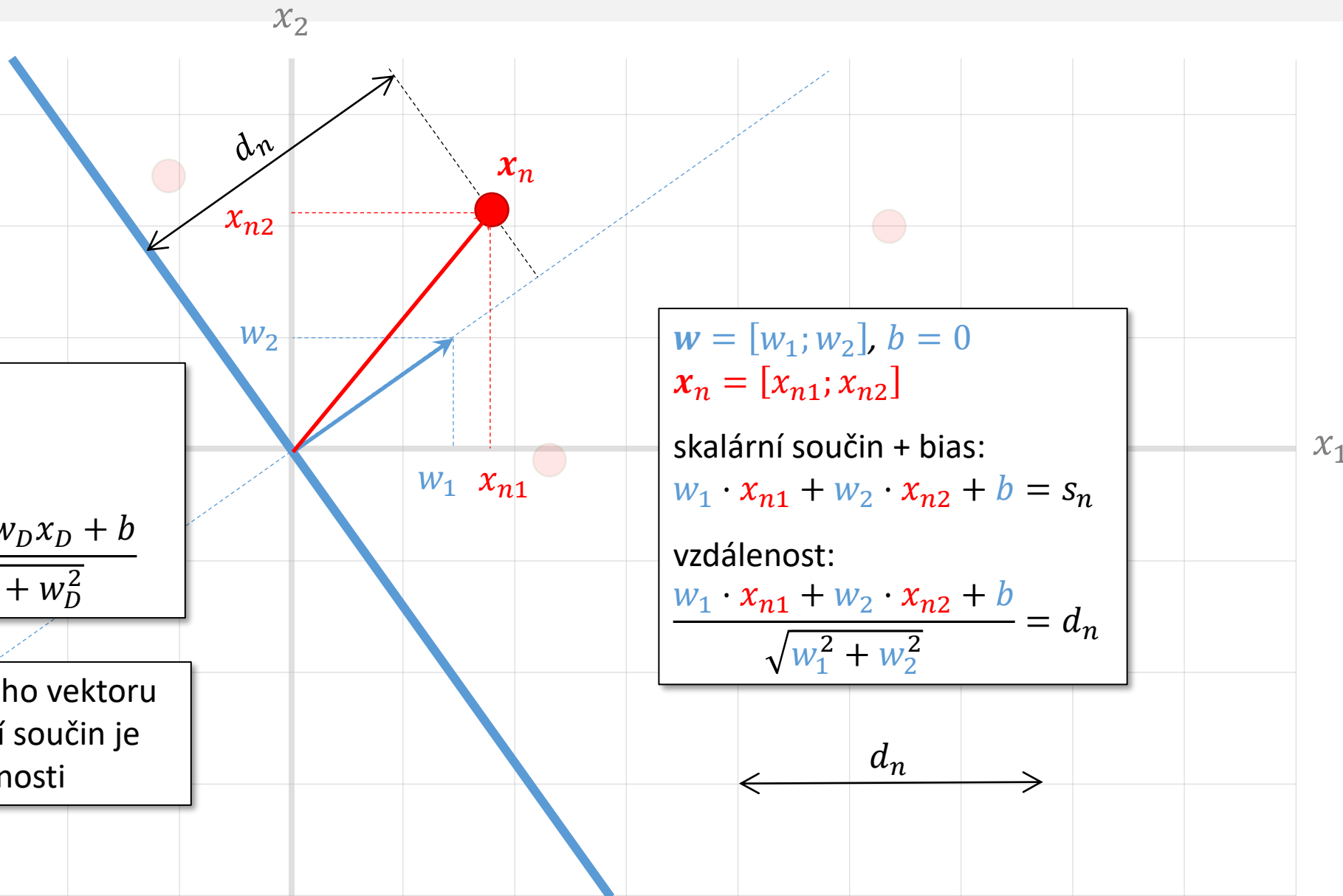
skalární součin + bias:

$$w_1 \cdot x_{n1} + w_2 \cdot x_{n2} + b = s_n$$

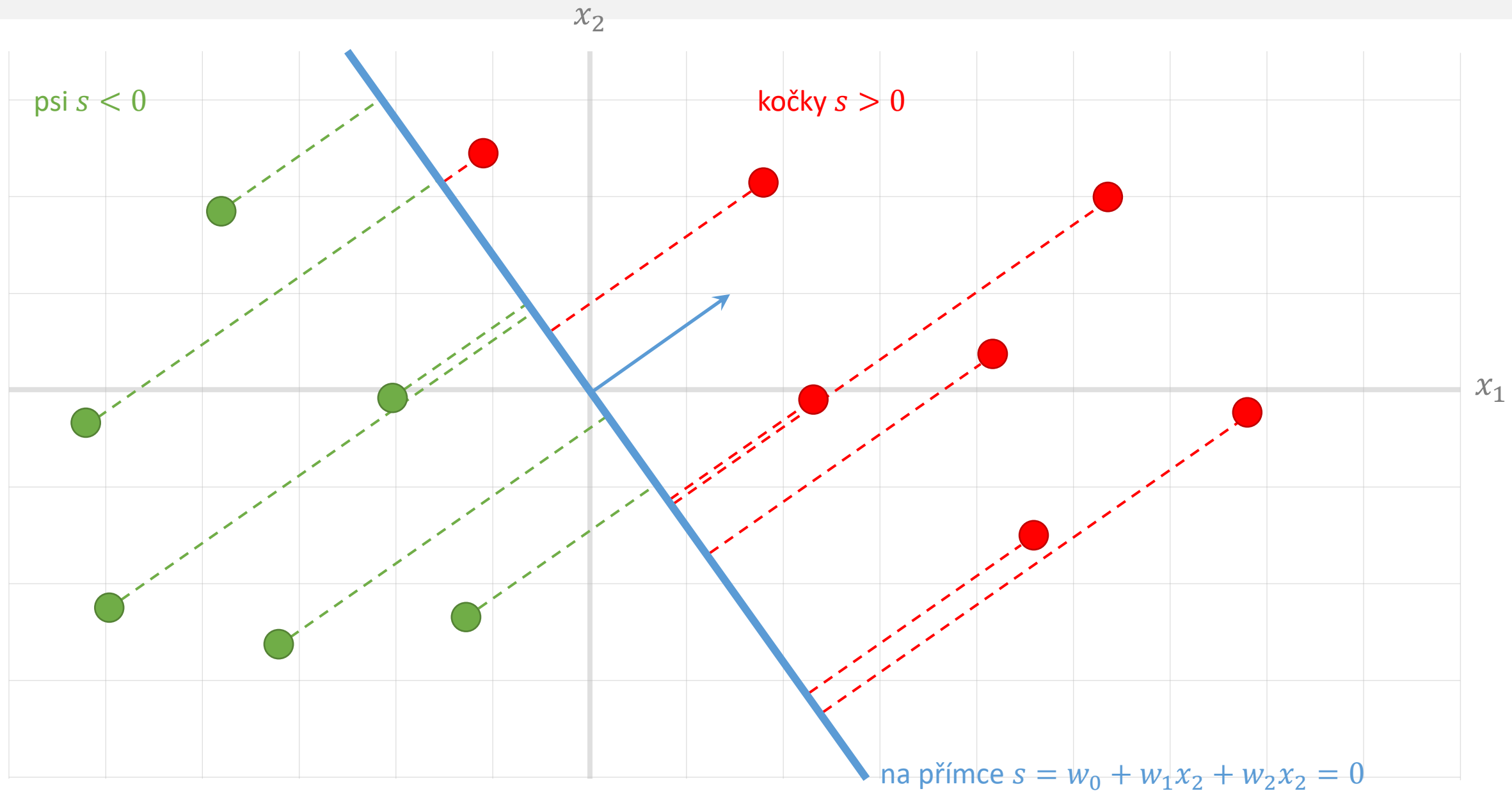
vzdálenost:

$$\frac{w_1 \cdot x_{n1} + w_2 \cdot x_{n2} + b}{\sqrt{w_1^2 + w_2^2}} = d_n$$

$$\longleftrightarrow d_n$$

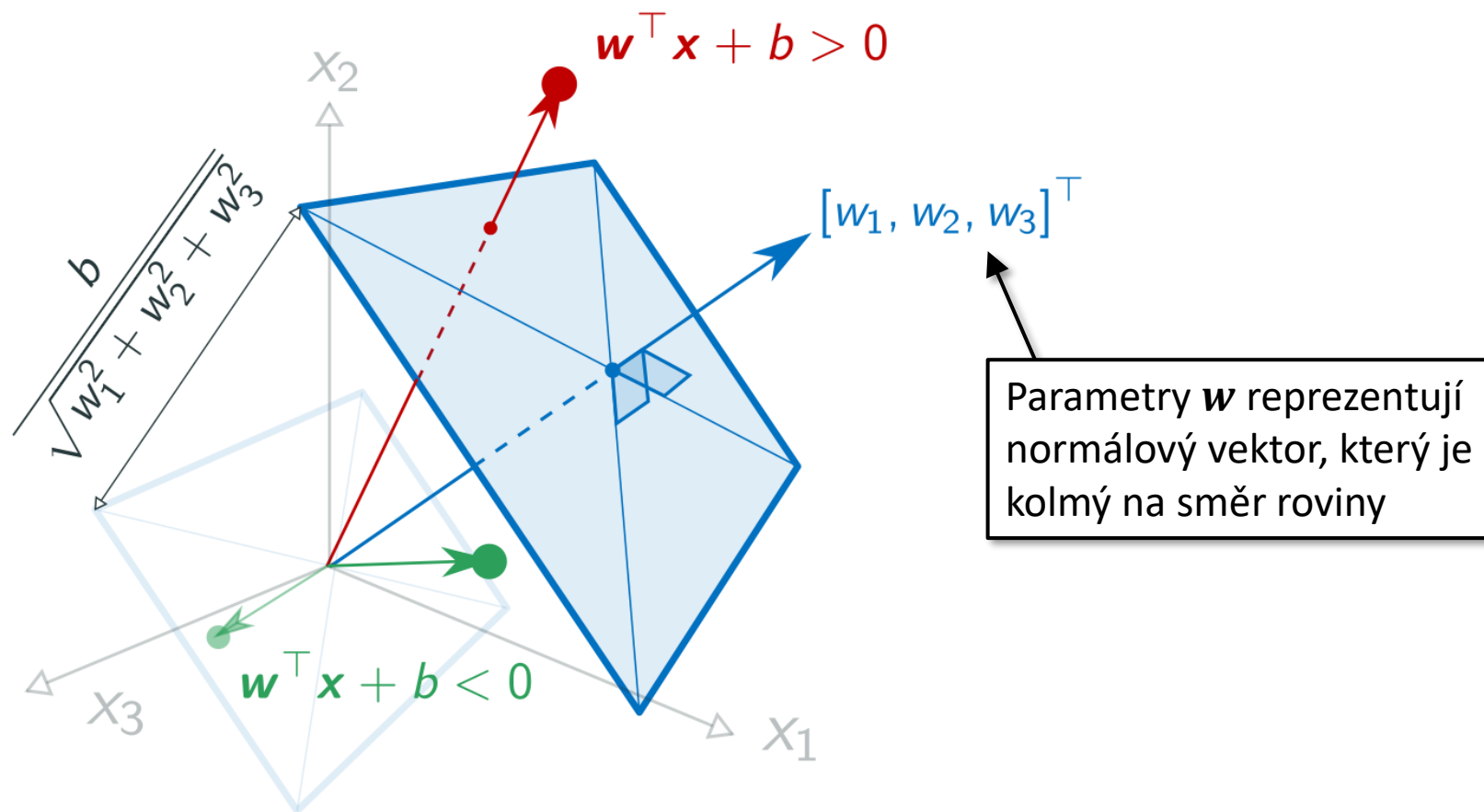


Ideální klasifikátor



Vícerozměrný prostor

Ve více rozměrech přímku nahrazuje rovina



Rozšíření z binární ($C = 2$) na multiclass klasifikaci ($C > 2$)

1. One-vs-rest (one-vs-all)

- C samostatných klasifikátorů, z nichž každý diskriminuje jednu ze tříd vůči ostatním
- např. kočka ("1") vs ostatní ("0"), pes ("1") vs ostatní ("0"), ...
- Pro $C = 10 \rightarrow 10$ klasifikátorů
- Vyhrává třída s nejvyšším skóre/pravděpodobností

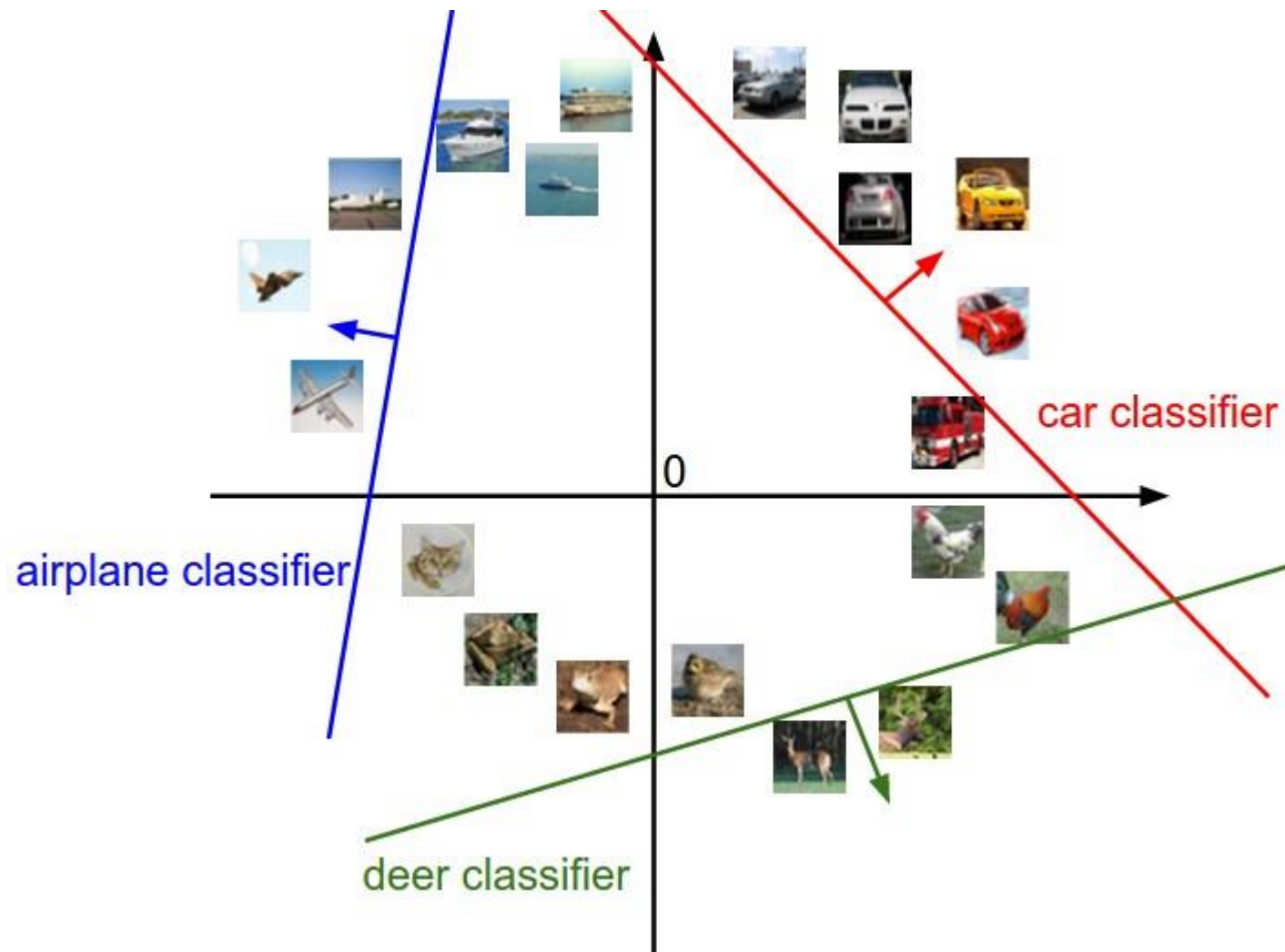
2. One-vs-one (all-vs-all)

- $C(C - 1)/2$ samostatných klasifikátorů pro každou dvojici tříd
- Např. kočka vs pes, kočka vs žába, pes vs žába, ...
- Pro $C = 10 \rightarrow 45$ klasifikátorů
- Pro $C = 1000$ (ImageNet) $\rightarrow 499500 \approx 0.5 \cdot 10^6$ klasifikátorů!
- Vyhrává třída s nejvyšším počtem "výher z duelů"

3. Reformulace úlohy

- jeden klasifikátor, ale výstupem bude C skóre pro každou třídu současně (paralelně)
- vyhrává třída s nejvyšším skóre/pravděpodobností
- \rightarrow tudy vede cesta!

Klasifikace do více tříd



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

Matice parametrů a lineární skóre

- Vektor vah \mathbf{w}_c a bias b_c nyní máme pro každou třídu $c = 1, \dots, C$

$$\mathbf{w}_c = [w_{c1}, \dots, w_{cD}]^\top$$

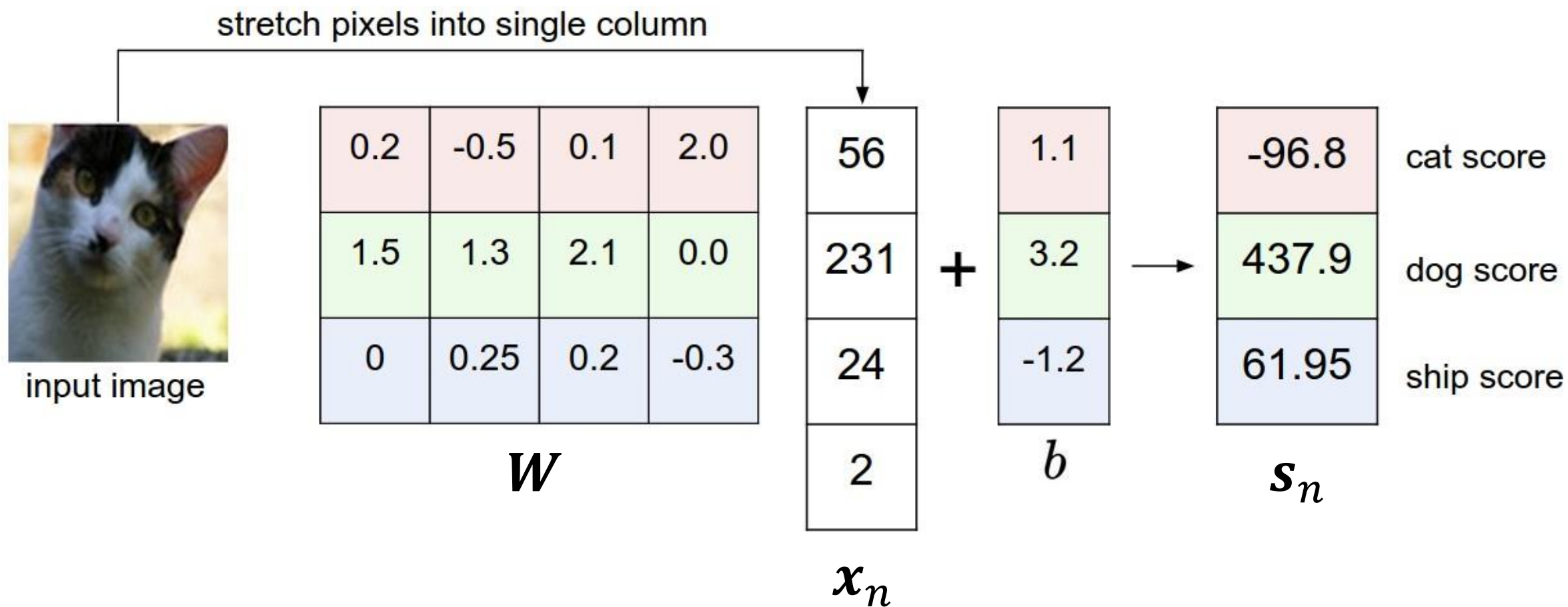
- Kompletní parametry klasifikátoru $\boldsymbol{\theta} = \{\mathbf{W}, \mathbf{b}\}$ jsou tedy matice vah a vektor biasů

$$\mathbf{W} = \begin{bmatrix} \mathbf{w}_1^\top \\ \vdots \\ \mathbf{w}_C^\top \end{bmatrix} = \begin{bmatrix} w_{11} & \dots & w_{1D} \\ \vdots & \ddots & \vdots \\ w_{C1} & \dots & w_{CD} \end{bmatrix} \quad \mathbf{b} = \begin{bmatrix} b_1 \\ \vdots \\ b_C \end{bmatrix}$$

- Lineární skóre (logity) vstupu \mathbf{x}_n dostaneme pro každou třídu jako

$$\mathbf{s}_n = \begin{bmatrix} s_{n1} \\ \vdots \\ s_{nC} \end{bmatrix} = \begin{bmatrix} w_{11} & \dots & w_{1D} \\ \vdots & \ddots & \vdots \\ w_{C1} & \dots & w_{CD} \end{bmatrix} \cdot \begin{bmatrix} x_{n1} \\ \vdots \\ x_{nC} \end{bmatrix} + \begin{bmatrix} b_1 \\ \vdots \\ b_C \end{bmatrix} = \mathbf{W}\mathbf{x}_n + \mathbf{b}$$

Matice parametrů a lineární skóre: příklad



Softmax

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

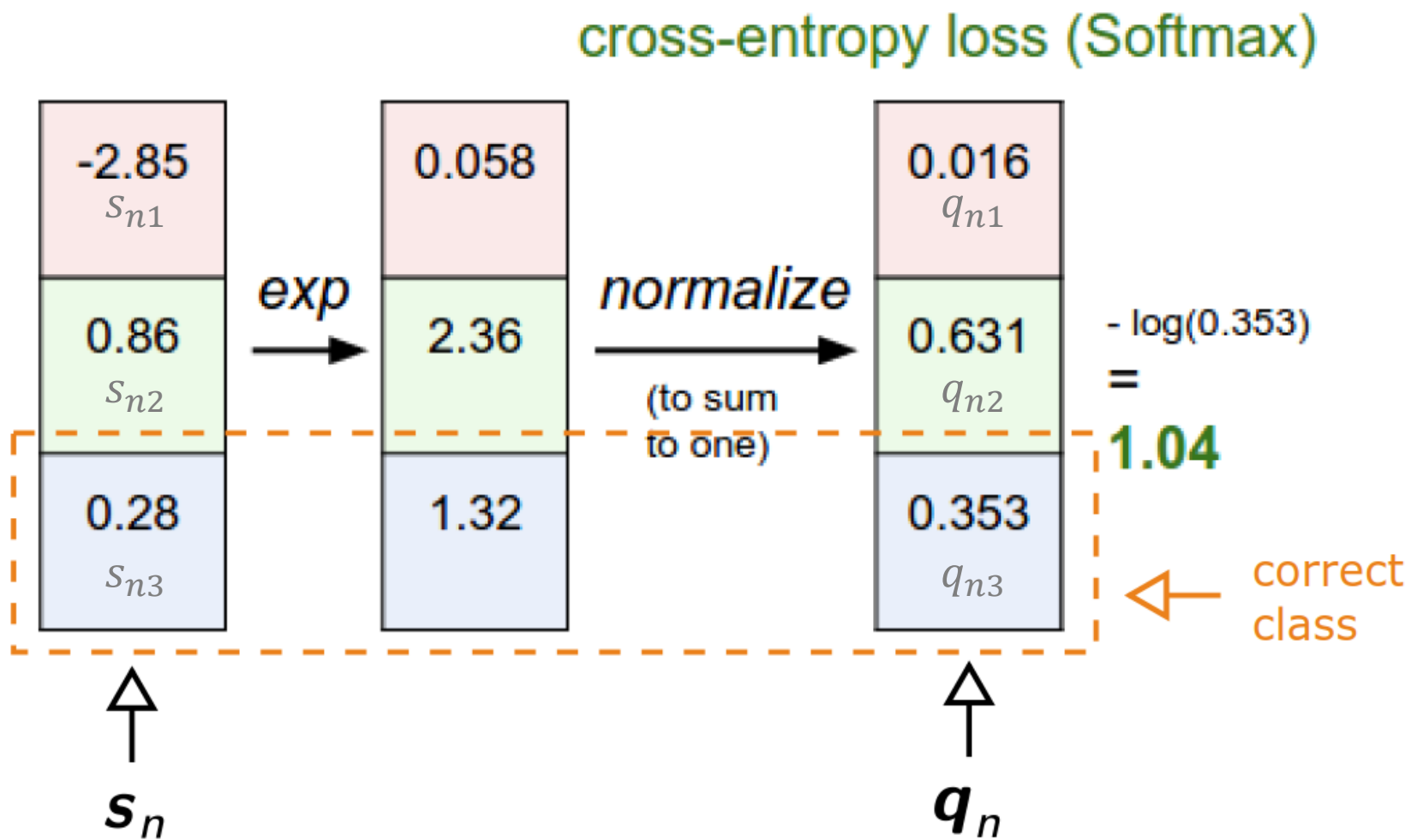
$$q_{nc} = P(\text{třída } c | \mathbf{x}_n) = \frac{e^{s_{nc}}}{\sum_{i=1}^C e^{s_{ni}}}$$

- Výstupem C -dimezionální vektor \mathbf{q}_n pravděpodobností jednotlivých tříd

$$\mathbf{q}_n = [q_{n1}, \dots, q_{nC}]^T, \quad 0 \leq q_{nc} \leq 1, \quad \sum_{c=1}^C q_{nc} = 1$$

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

Softmax příklad



Formulace úlohy trénování klasifikátoru

- Máme klasifikátor $f(\mathbf{x}_n; \mathbf{W}, \mathbf{b})$, který převezme obrázek \mathbf{x}_n a spočítá vektor \mathbf{q}_n , který vyjadřuje pravděpodobnosti jednotlivých tříd

$$\mathbf{s}_n = \mathbf{W}\mathbf{x}_n + \mathbf{b}$$

$$\mathbf{q}_n = f(\mathbf{x}_n; \mathbf{W}, \mathbf{b}) = \frac{e^{s_n}}{\sum_{c=1}^C e^{s_{nc}}}$$

- Na klasifikátor $f(\mathbf{x}_n; \mathbf{W}, \mathbf{b})$ nahlížíme jako na funkci, která má
 - vstup \mathbf{x}_n
 - a parametry \mathbf{W}, \mathbf{b}
- Úkolem nyní je nalézt optimální parametry $\boldsymbol{\theta} = \{\mathbf{W}, \mathbf{b}\}$ takové, se kterými funkce $f(\mathbf{x}_n; \boldsymbol{\theta})$ bude správně klasifikovat ideálně všechny obrázky \mathbf{x}_n , $n = 1, \dots, N$ z trénovací sady

Kritérium úspěšnosti klasifikátoru

- Pro každý obrázek \mathbf{x}_n známe správnou odpověď $y_n \in \{1, \dots, C\}$ (target)
- Predikce klasifikátoru $z_n \in \{1, \dots, C\}$ může být např. index max. prvku vektoru \mathbf{q}_n

$$z_n = \underset{c}{\operatorname{argmax}} \mathbf{q}_n$$

- Porovnáním predikcí a targetů spočítáme, jak dobře f klasifikuje

$$S_n = \mathbb{1}(z_n = y_n) \quad \leftarrow \text{Pro jeden obrázek}$$

$$S(\boldsymbol{\theta}) = \frac{1}{N} \sum_{n=1}^N S_n \quad \leftarrow \text{Pro všechny dohromady}$$

- Pokud $S(\boldsymbol{\theta}) > S(\boldsymbol{\theta}')$, víme, že parametry $\boldsymbol{\theta}$ jsou lepší než $\boldsymbol{\theta}'$

Formulace trénování jako minimalizace

- Ekvivalentně můžeme preferovat parametry, pro které nepřesnost je minimální

$$L_n = \mathbb{1}(z_n \neq y_n)$$

Pro jeden obrázek

$$L(\theta) = \frac{1}{N} \sum_{n=1}^N L_n$$

Pro všechny dohromady

- $L(\theta)$ se nazývá 0-1 loss a vyjadřuje, jak moc špatné parametry θ jsou
- Kritérium $L(\theta)$ kvantifikuje chybu modelu
- Úkolem je nalézt optimální parametry θ^* , které tuto chybu na vzorových datech X minimalizují, neboli



$$\theta^* = \arg \min_{\theta} L(\theta)$$



Multiclass cross entropy

- 0-1 loss je nevhodný, především proto, že není diferencovatelný → nelze použít standardní optimalizační techniky
- Logistická regrese definuje „lepší“ kritérium, tzv. **křížovou entropii**

$$L_n = - \sum_{c=1}^C p_{nc} \log q_{nc}$$

← Pro jeden obrázek

- kde

$\mathbf{p}_n = [p_{n1}, \dots, p_{nC}]^T$... cílové rozdělení (ground truth / target)

$\mathbf{q}_n = [q_{n1}, \dots, q_{nC}]^T$... výstupní pravd. (predikce) klasifikátoru

jsou vektory, na které nahlížíme jako na diskrétní rozdělení

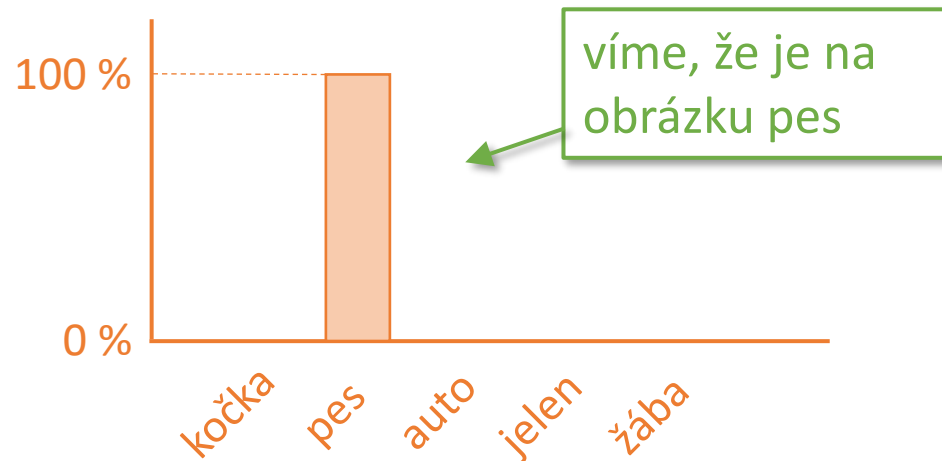
→ cross entropy = minimalizace rozdílu mezi dvěma rozděleními

Multiclass cross entropy

$$L_n = - \sum_{c=1}^C p_{nc} \log q_{nc}$$

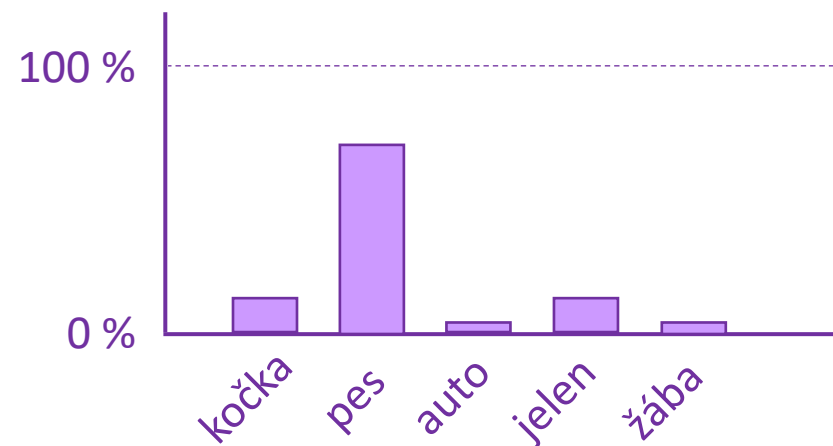
$$\mathbf{p}_n = [p_{n1}, \dots, p_{nC}]^T$$

cílové rozdělení (ground truth / target)



$$\mathbf{q}_n = [q_{n1}, \dots, q_{nC}]^T$$

výstup (predikce) klasifikátoru



One hot encoding


- Pro více tříd je y_n celé číslo, tj. $y_n \in \{1, \dots, C\}$
- Pokud $C = 5 \rightarrow$ požadované rozdělení pak je

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

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

Softmax + cross entropy

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

$$-L_n = \sum_{c=1}^c p_{nc} \log q_{nc} = \log q_{ny_n} = \log \frac{e^{s_{ny_n}}}{\sum_{c=1}^C e^{s_{nc}}}$$


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

- Pokud rozepíšeme logaritmus zlomku, dostaneme druhou častou variantu zápisu


$$L_n = -\log \frac{e^{s_{ny_n}}}{\sum_{c=1}^C e^{s_{nc}}} = -s_{ny_n} + \log \sum_{c=1}^C e^{s_{nc}}$$

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

Vliv přeškálování parametrů

$$\mathbf{s}_n = \mathbf{W}\mathbf{x}_n$$
$$\mathbf{q}_n = \frac{e^{\mathbf{s}_n}}{\sum_{c=1}^C e^{\mathbf{s}_{nc}}}$$

Matice parametrů
vynásobená 10x



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

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

- 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
- Aby nedocházelo k optimalizaci lossu pouhým škálováním velikosti parametrů, měli bychom nějak toto chování penalizovat

Regularizace

- Do kritéria (lossu) zavedeme člen penalizující velikost vah pomocí L2 normy

$$R(\mathbf{W}) = \|\mathbf{W}\|_2^2 = \sum_{c=1}^C \sum_{d=1}^D w_{cd}^2$$

- Celkově pak úloha je

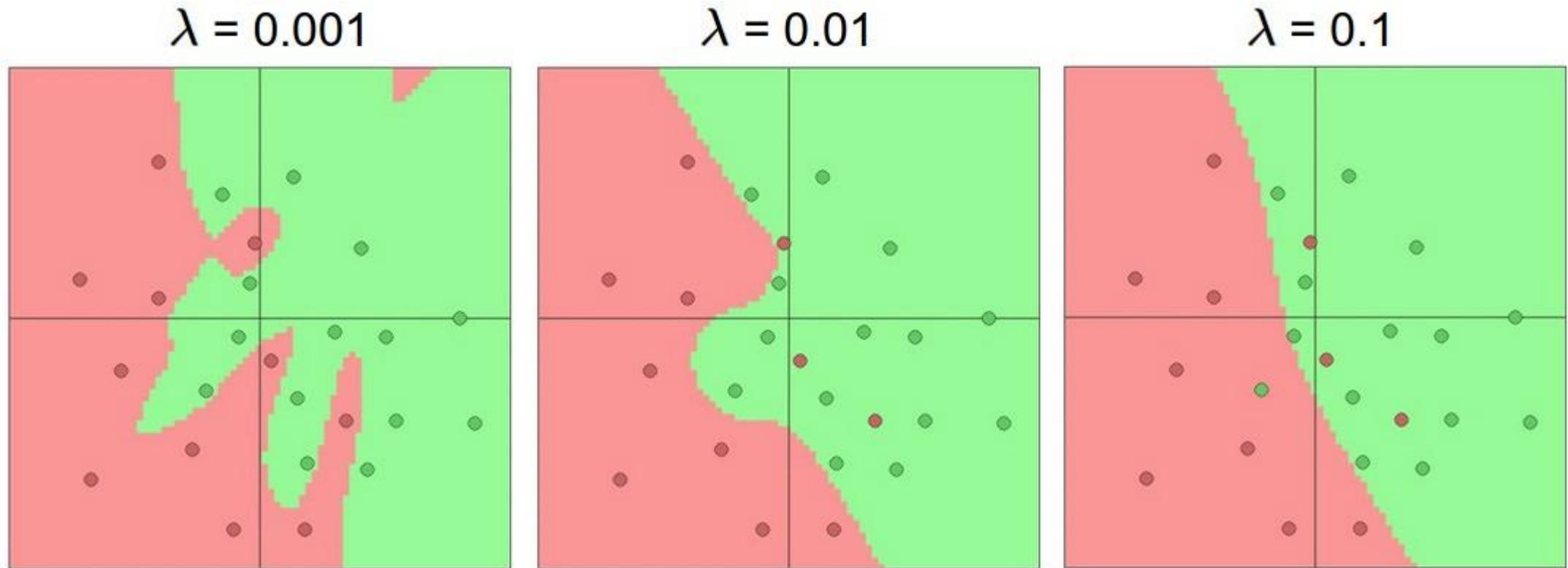
$$\boldsymbol{\theta}^* = \arg \min_{\boldsymbol{\theta}} L(\boldsymbol{\theta}) + \lambda R(\mathbf{W})$$

- $\lambda R(\mathbf{W})$ je regularizační člen
 - λ je hyperparametr, často označovaný jako weight decay
 - Obvykle aplikován pouze na váhy, na bias nikoliv

Regularizace: příklad

- Např. $\mathbf{x} = [1, 1, 1, 1]^T$ a dvojce různé parametry:
 - $\mathbf{w}_1 = [1, 0, 0, 0]^T$
 - $\mathbf{w}_2 = [0.25, 0.25, 0.25, 0.25]^T$
- Přestože $\mathbf{w}_1^T \mathbf{x} = \mathbf{w}_2^T \mathbf{x} = 1$, preferujeme \mathbf{w}_2
- Brání přeučení
 - \mathbf{w}_2 má menší normu
 - \mathbf{w}_1 sází všechno na jeden příznak, zatímco \mathbf{w}_2 důležitost rozkládá
 - normu $\|\mathbf{w}\|_2$ lze interpretovat jako apriorní pravděpodobnost
 - regularizace brání změnám \rightarrow trénování méně reaguje na změny

Vliv regularizace



<http://cs231n.github.io/neural-networks-1/>

Jak vybereme optimální lineární klasifikátor?

USU v kostce:

1. Vytvoříme diskriminativní klasifikátor s upravitelnými parametry
2. Kvantifikujeme jeho úspěšnost klasifikace nějakým kritériem
3. Upravujeme parametry a poznamenáváme si jejich výsledek (hodnotu kritéria)
4. Jako nejlepší vybereme takové parametry, které optimalizují zvolené kritérium

Minimalizace funkce

- Hledáme parametry klasifikátoru $\boldsymbol{\theta}^* = [\mathbf{w}^*; b^*]$, které na našem datasetu minimalizují loss $L(\boldsymbol{\theta})$ a zároveň regularizaci $\lambda R(\mathbf{W})$

$$\boldsymbol{\theta}^* = \arg \min_{\boldsymbol{\theta}} \sum_{n=1}^N L_n(\boldsymbol{\theta}) + \lambda R(\mathbf{W})$$

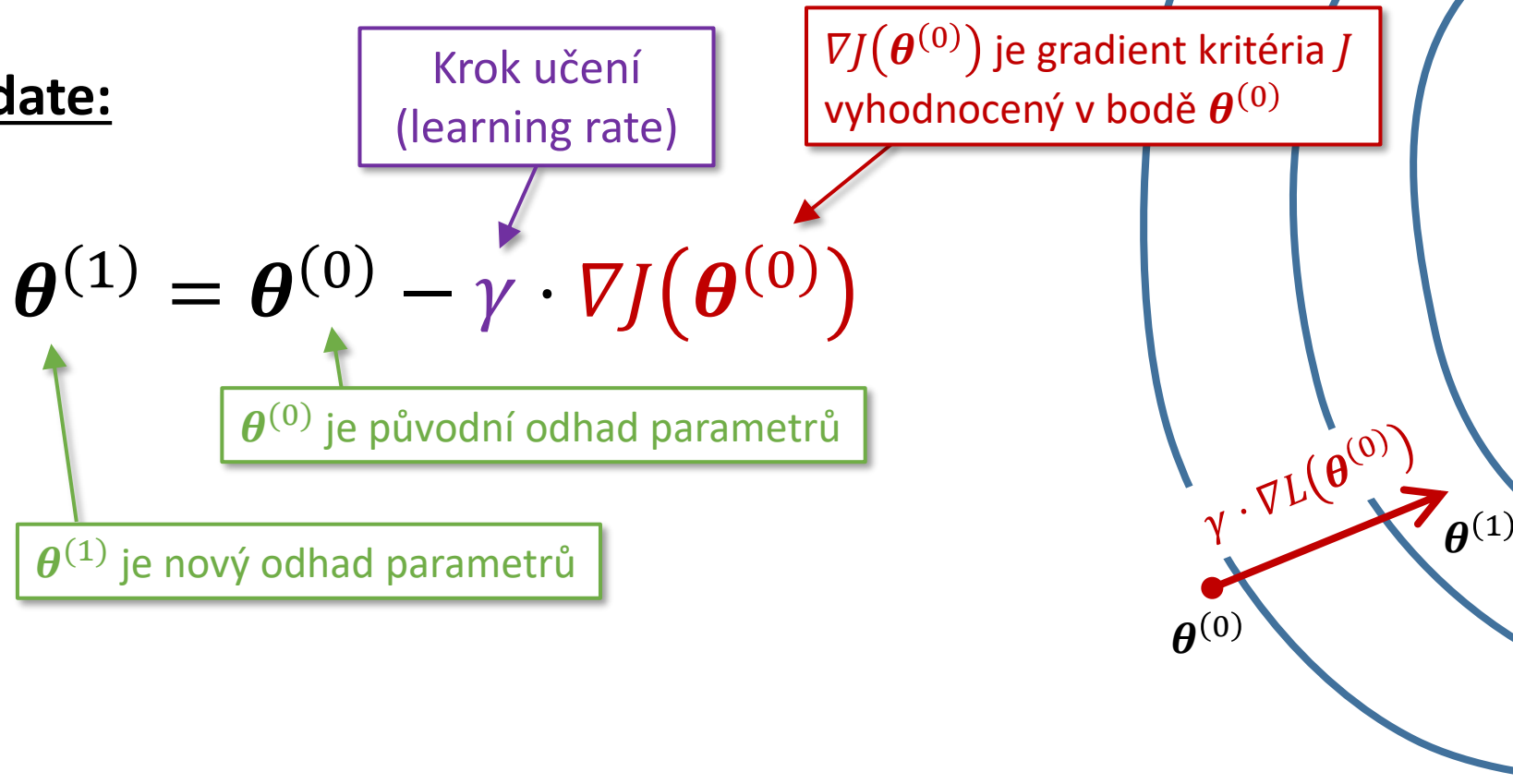
- Na $J(\boldsymbol{\theta}) = L(\boldsymbol{\theta}) + \lambda R(\mathbf{W})$ nahlížíme jako na jakoukoli jinou funkci $y = f(x)$
- Proměnná x jsou hledané parametry $\boldsymbol{\theta} = \{\mathbf{W}; \mathbf{b}\}$
- Minimum funkce $f(x)$ lze nalézt např. metodou největšího spádu
 - Postupně upravujeme odhad optima posunem ve směru, ve kterém funkce nejrychleji klesá = mínus gradientem

Metoda největšího spádu (Gradient Descent)

Inicializace:

- parametry $\theta^{(0)}$ na náhodné hodnoty

Update:



Metoda největšího spádu (Gradient Descent)

Inicializace:

- parametry $\theta^{(0)}$ na náhodné hodnoty

Update:

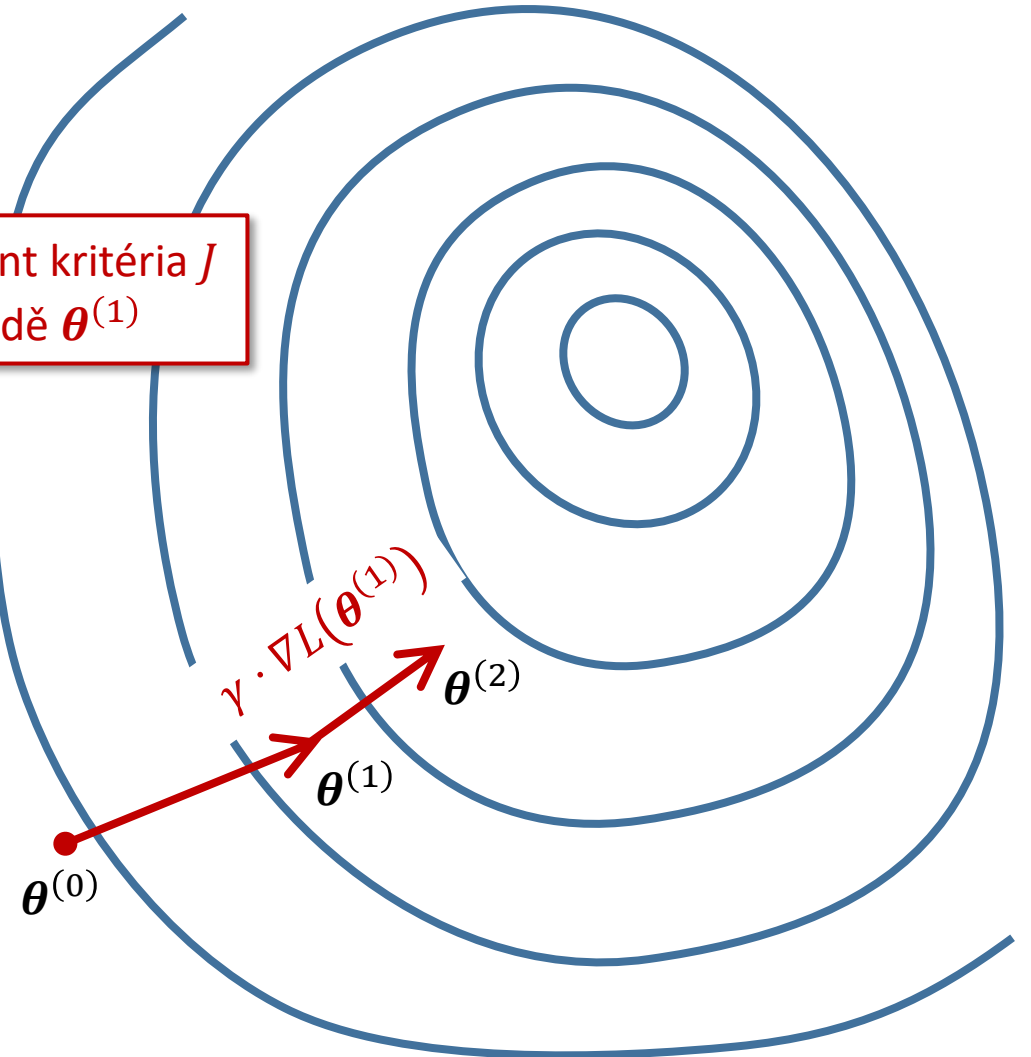
Krok učení
(learning rate)

$\nabla J(\theta^{(1)})$ je gradient kritéria J
vyhodnocený v bodě $\theta^{(1)}$

$$\theta^{(2)} = \theta^{(1)} - \gamma \cdot \nabla J(\theta^{(1)})$$

$\theta^{(1)}$ je původní odhad parametrů

$\theta^{(2)}$ je nový odhad parametrů

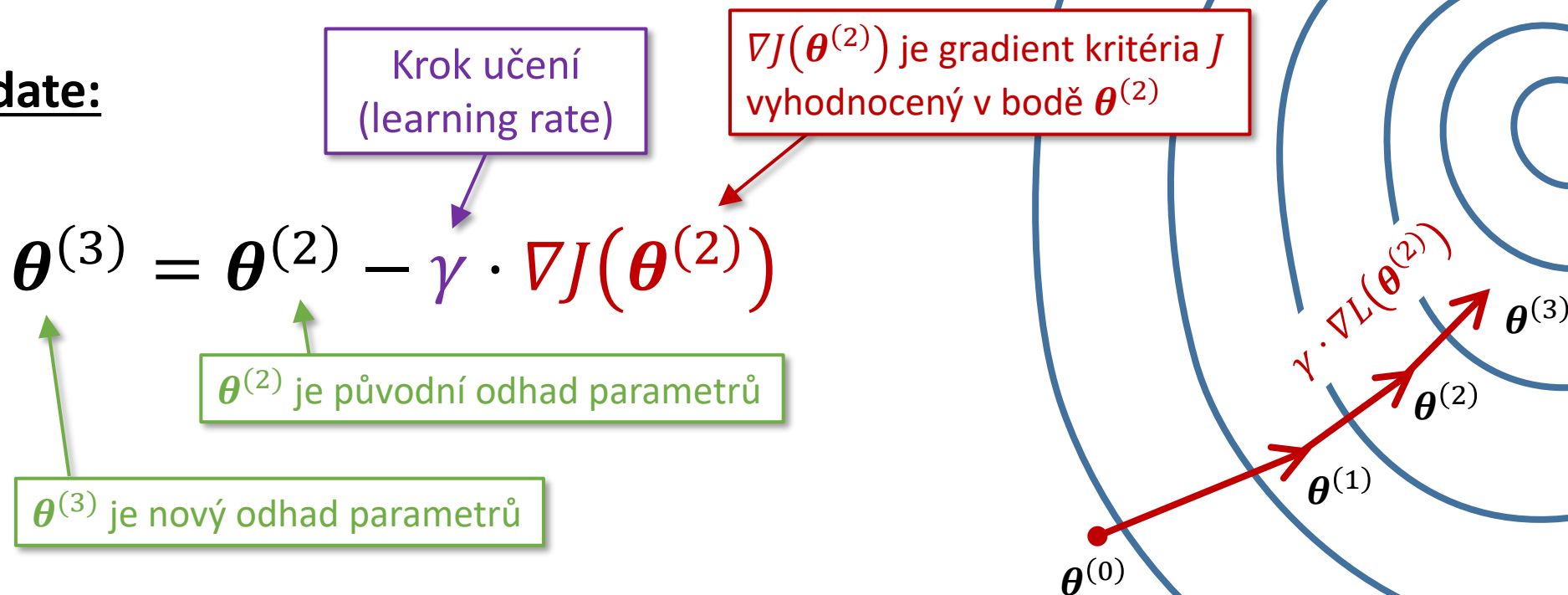


Metoda největšího spádu (Gradient Descent)

Inicializace:

- parametry $\theta^{(0)}$ na náhodné hodnoty

Update:



Metoda největšího spádu (Gradient Descent)

Inicializace:

- parametry $\theta^{(0)}$ na náhodné hodnoty

Update:

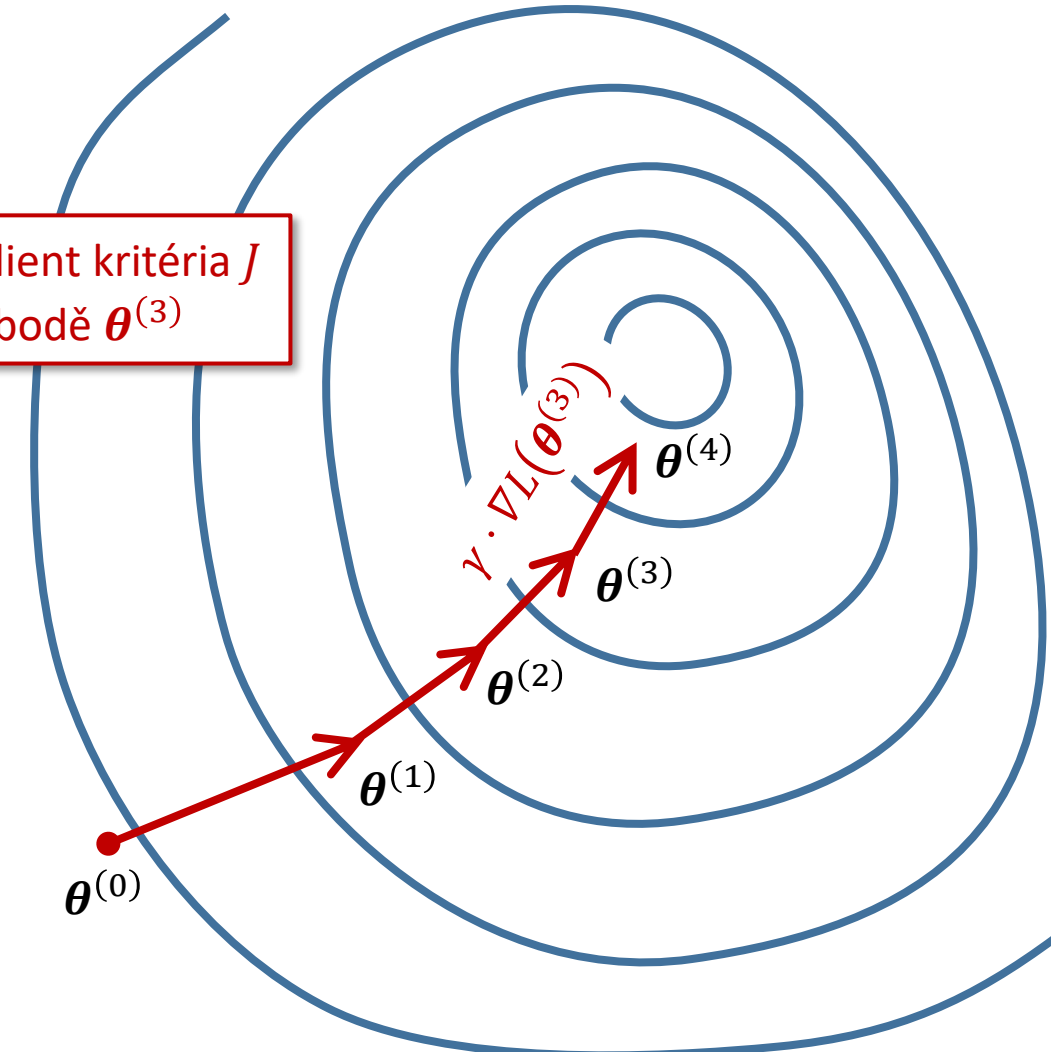
$$\theta^{(4)} = \theta^{(3)} - \gamma \cdot \nabla J(\theta^{(3)})$$

$\theta^{(4)}$ je nový odhad parametrů

$\theta^{(3)}$ je původní odhad parametrů

Krok učení (learning rate)

$\nabla J(\theta^{(3)})$ je gradient kritéria J vyhodnocený v bodě $\theta^{(3)}$



Metoda největšího spádu (Gradient Descent)

Inicializace:

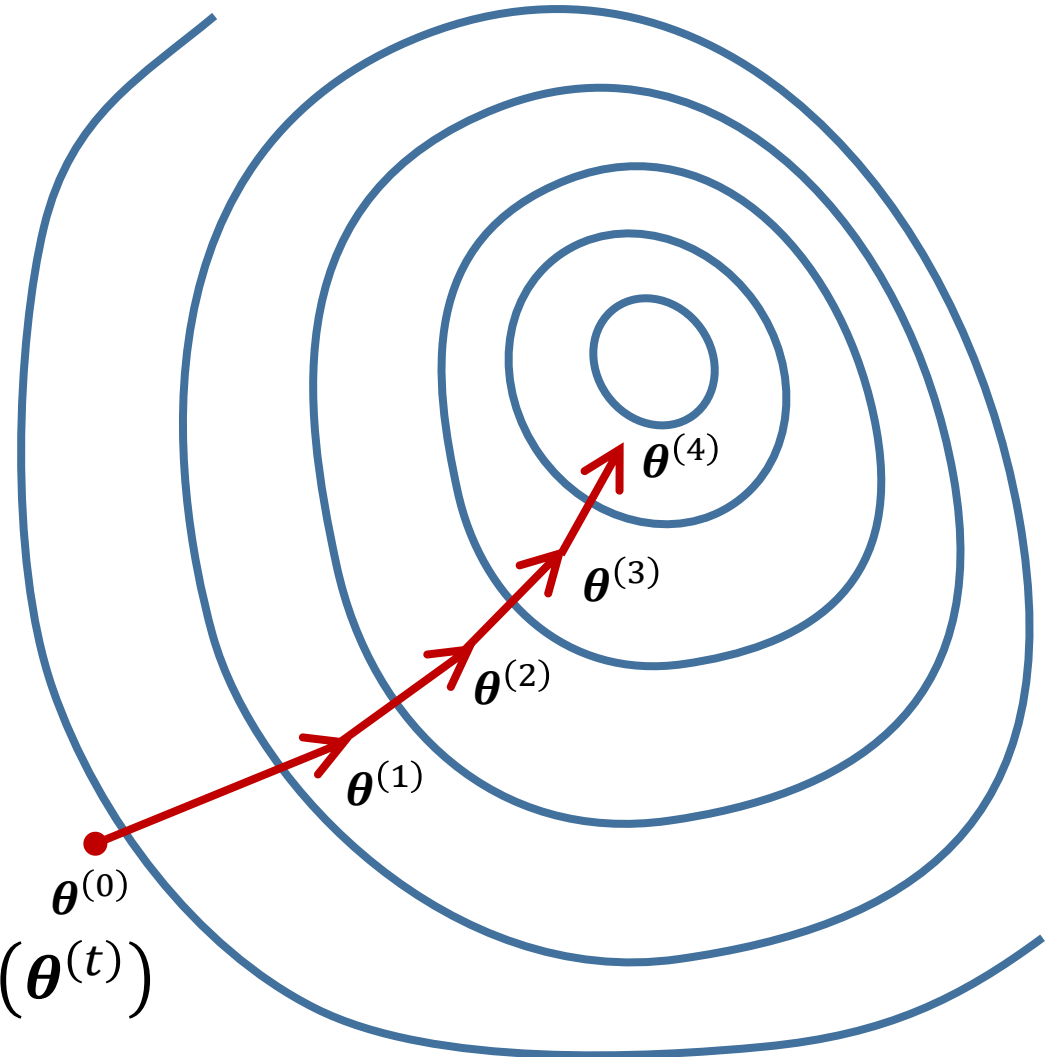
- parametry $\theta^{(0)}$ na náhodné hodnoty

Opakujeme:

$$\theta^{(t+1)} = \theta^{(t)} - \gamma \cdot \nabla J(\theta^{(t)})$$

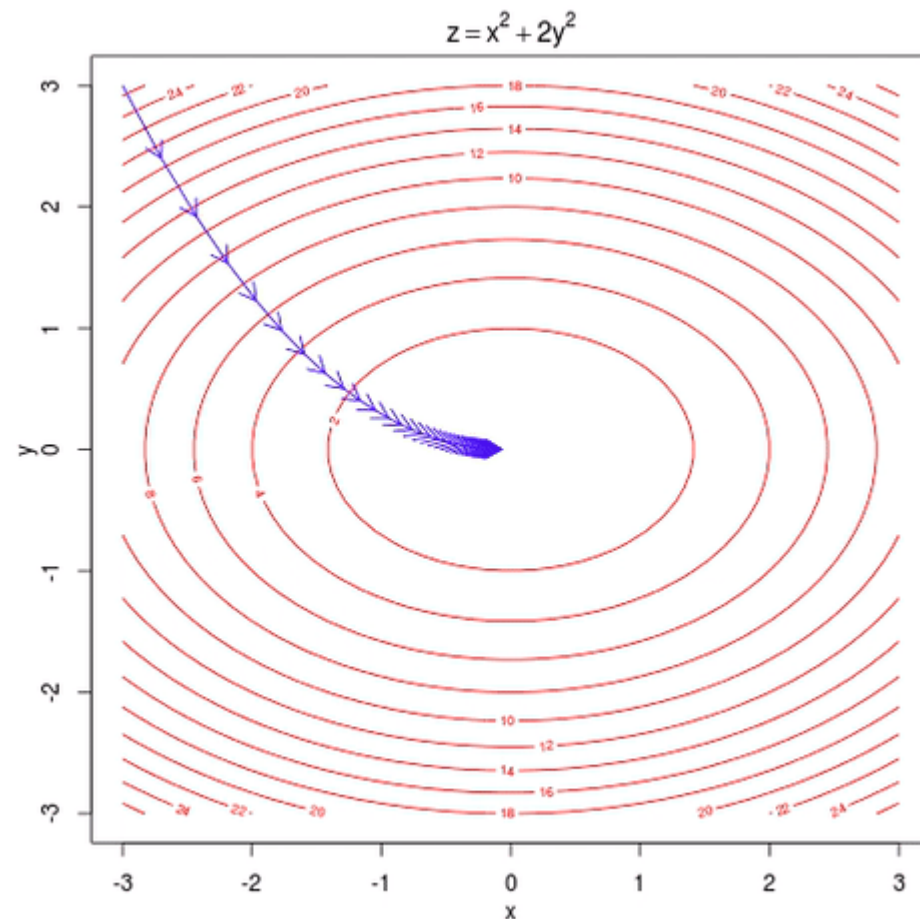
Skončíme:

- po vykonaném počtu kroků
- kritérium J již delší dobu neklesá, $J(\theta^{(t+1)}) \geq J(\theta^{(t)})$



Velikost kroku γ

- Hyperparametr
- V kontextu sítí obvykle tzv. learning rate
- **Výrazný vliv na výsledný model**
- Typické hodnoty $\gamma \approx 10^{-3}$



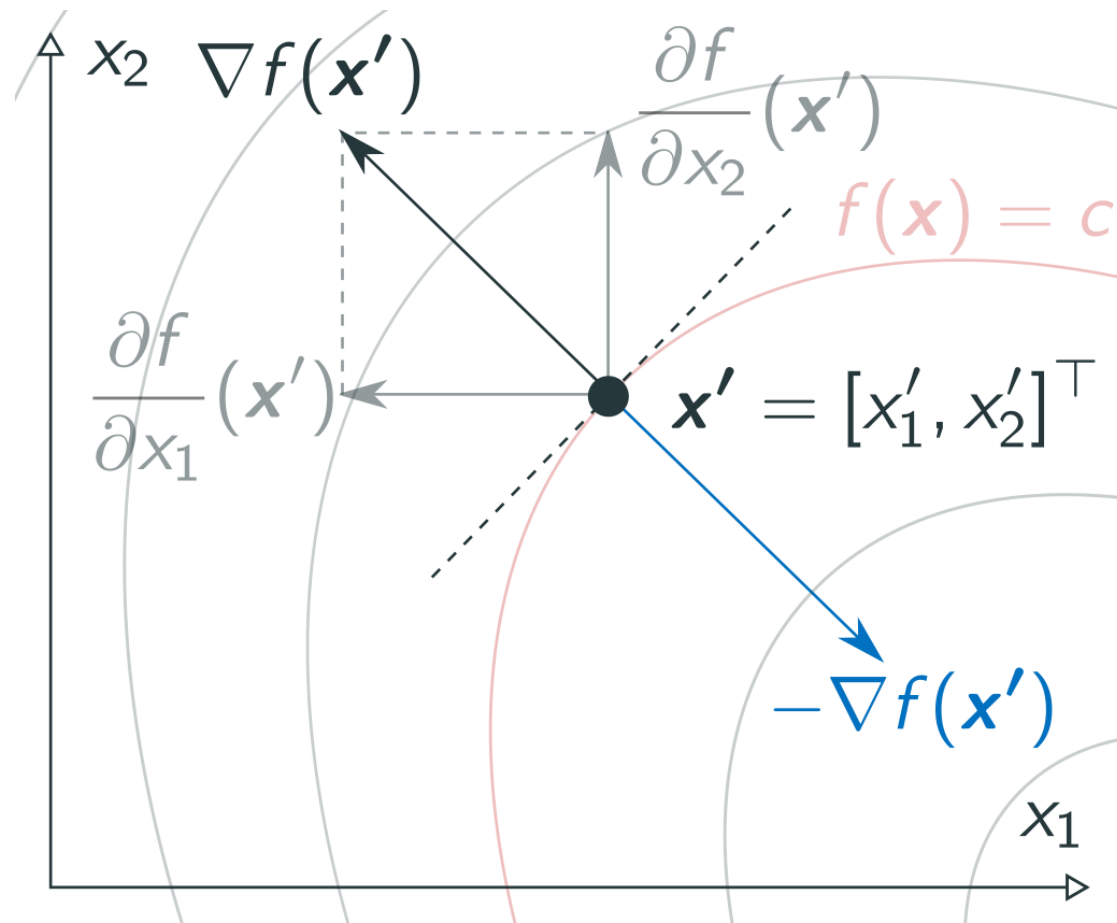
animace: <http://vis.supstat.com/2013/03/gradient-descent-algorithm-with-r> (nefunkční)

Gradient

- Gradient je vektor **parciálních derivací**

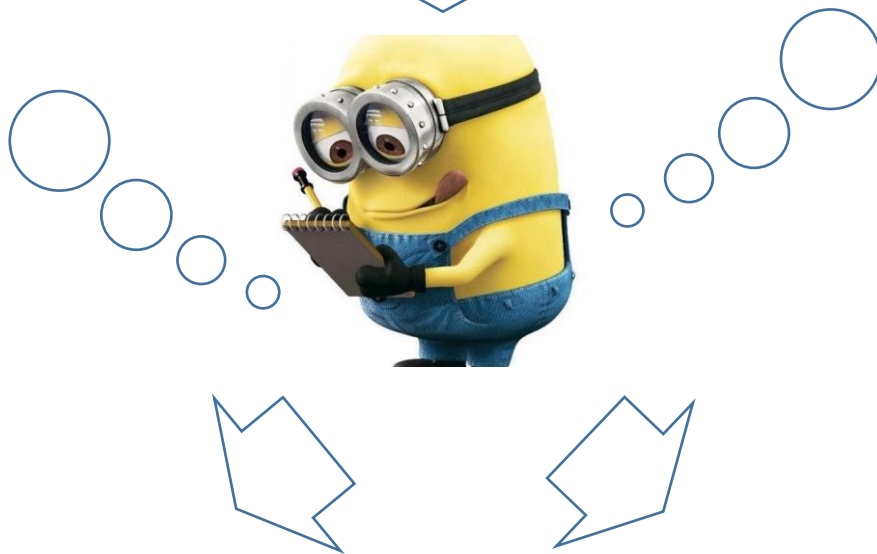
$$\nabla f(x') = \left[\frac{\partial f}{\partial x_1}(x'_1), \dots, \frac{\partial f}{\partial x_D}(x'_D) \right]^\top$$

- Pro minimalizaci kritéria musíme derivovat



Gradient křížové entropie

$$J(\boldsymbol{\theta}) = \lambda \|\mathbf{W}\|^2 + \sum_{n=1}^N \sum_{c=1}^C p_{nc} \log \frac{\exp(\mathbf{w}_c \mathbf{x}_n + b_c)}{\sum_{i=1}^C \exp(\mathbf{w}_i \mathbf{x}_n + b_i)} \quad \leftarrow q_{nc}$$



Gradient na c -tý řádek matice \mathbf{W}

$$\frac{\partial J}{\partial \mathbf{w}_c} = 2\lambda \mathbf{w}_c + \sum_{n=1}^N (q_{nc} - p_{nc}) \mathbf{x}_n$$

$$\frac{\partial J}{\partial b_c} = \sum_{n=1}^N (q_{nc} - p_{nc})$$

Gradient na c -tý prvek vektoru \mathbf{b}

Softmax cross entropy: přehled

- Celkově tedy kritérium je

$$J(\boldsymbol{\theta}) = \lambda \|\mathbf{W}\|^2 + \sum_{n=1}^N \sum_{c=1}^C p_{nc} \log q_{nc}$$

kde

$$\mathbf{q}_n = [q_{n1}, \dots, q_{nC}]^T = \text{Softmax}(\mathbf{W}\mathbf{x}_n + \mathbf{b})$$

- Parametry jsou matice a vektor

$$\mathbf{W} = \begin{bmatrix} \mathbf{w}_1^T \\ \vdots \\ \mathbf{w}_C^T \end{bmatrix} = \begin{bmatrix} w_{11} & \dots & w_{1D} \\ \vdots & \ddots & \vdots \\ w_{C1} & \dots & w_{CD} \end{bmatrix} \quad \mathbf{b} = \begin{bmatrix} b_1 \\ \vdots \\ b_C \end{bmatrix}$$

- Gradienty:

$$\frac{\partial J}{\partial \mathbf{w}_c} = 2\lambda \mathbf{w}_c + \sum_{n=1}^N (q_{nc} - p_{nc}) \mathbf{x}_n \quad \frac{\partial J}{\partial b_c} = \sum_{n=1}^N (q_{nc} - p_{nc})$$

rozměr?

rozměr?

Gradient descent (GD) pro multiclass logistickou regresi

Inicializujeme:

- $\theta = \{W, b\}$ na náhodné hodnoty

Opakujeme:

1. Pro každý vzorek x_n v trénovací sadě x_1, \dots, x_N
 - a. predikujeme pravděpodobnosti q_n
 - b. vyhodnotíme dílčí kritérium J_n a přičteme k celkové hodnotě $J(\theta)$
 - c. vyhodnotíme dílčí gradient ∇J_n a přičteme k celkovému ∇J
2. updatujeme parametry θ akumulovaným gradientem ∇J s krokem γ

Zastavíme:

- po fixním počtu iterací
- parametry θ se ustálí
- hodnota kritéria $J(\theta)$ již delší dobu neklesá

Gradient descent pro binární logistickou regresi: poznámky

- Uvedený postup je velmi neefektivní
- Update vždy až po kompletním nasčítání gradientů přes celou trénovací sadu
- Např. ImageNet však cca 14 milionů obrázků
- Vstupní vektory (obrázky) sice předpokládáme nezávislé, jsou si ale podobné v tom smyslu, že pocházejí ze stejného rozdělení pravděpodobnosti
- Možná na update stačí malý vzorek (**minibatch**), není nutné vidět všechny obrázky
- Takto vznikne tzv. **Minibatch Gradient Descent**
- Pokud pouze jeden vzorek → **Stochastic Gradient Descent (SGD)**
 - https://en.wikipedia.org/wiki/Stochastic_gradient_descent

Varianty GD a názvosloví dle velikosti batche

velikost batche B	různé názvy pro totéž
$B = N$	Gradient descent
	Batch gradient descent
	Steepest descent
$1 < B \ll N$	Minibatch gradient descent
$B = 1$	Stochastic gradient descent (SGD)
	Online gradient descent
	Incremental gradient descent

- Aby to nebylo jednoduché, obvykle minibatch = batch
- Minibatch gradient descent najdeme v knihovnách pod jménem Stochastic gradient descent (SGD) s volitelným `batch_size` parametrem (B)
- „Pořádek je pro blbce, inteligent zvládá chaos.“

Stochastic Gradient Descent (SGD) pro multiclass logistickou regresi

Inicializujeme:

- $\theta = \{W, b\}$ na náhodné hodnoty

Opakujeme:

1. Pro každý vzorek x_n v navzorkované dávce x_1, \dots, x_B
 - a. predikujeme pravděpodobnosti q_n
 - b. vyhodnotíme dílčí kritérium J_n a přičteme k celkové hodnotě $J(\theta)$
 - c. vyhodnotíme dílčí gradient ∇J_n a přičteme k celkovému ∇J
2. updatujeme parametry θ akumulovaným gradientem ∇J s krokem γ

Zastavíme:

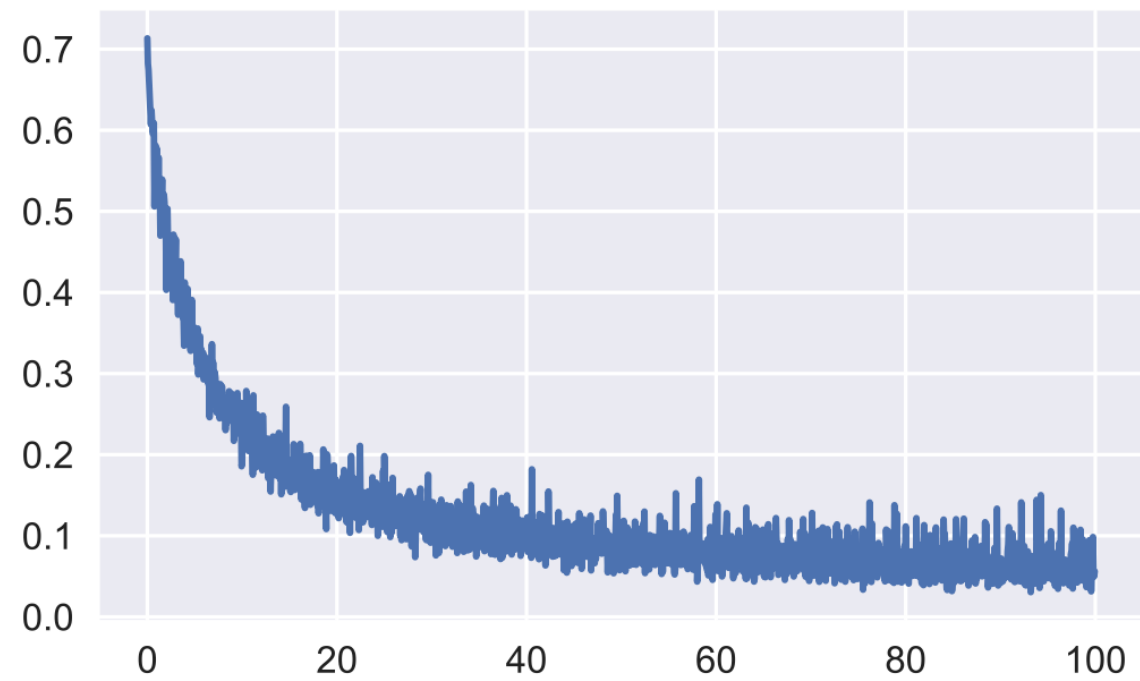
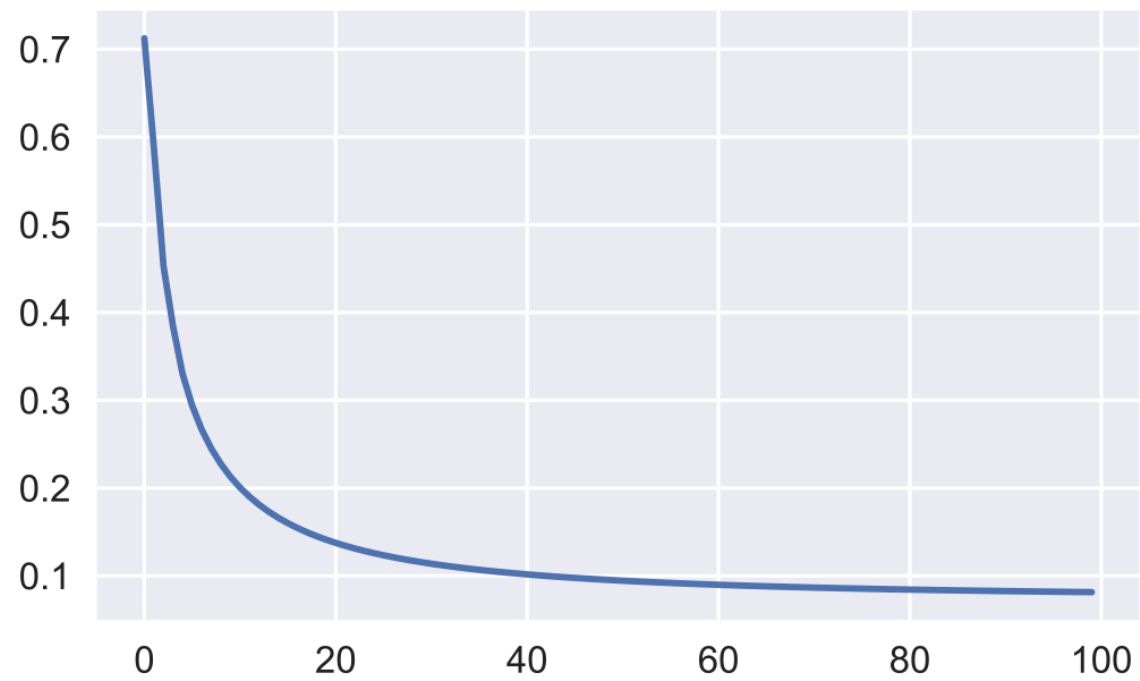
- po fixním počtu iterací
- parametry θ se ustálí
- hodnota kritéria $J(\theta)$ již delší dobu neklesá

GD vs SGD

Gradient descent	Stochastic gradient descent
skutečný gradient	pouze aproximuje gradient
stabilnější konvergence	rychlejší konvergence
konverguje do minima	osciluje kolem minima
náchylnější k upadnutí do lokálního minima	robustnější vůči lokálním minimum

Především vzhledem k výpočetním nárokům plného GD se pro učení neuronových sítí se prakticky výhradně používá Stochastic/Minibatch GD

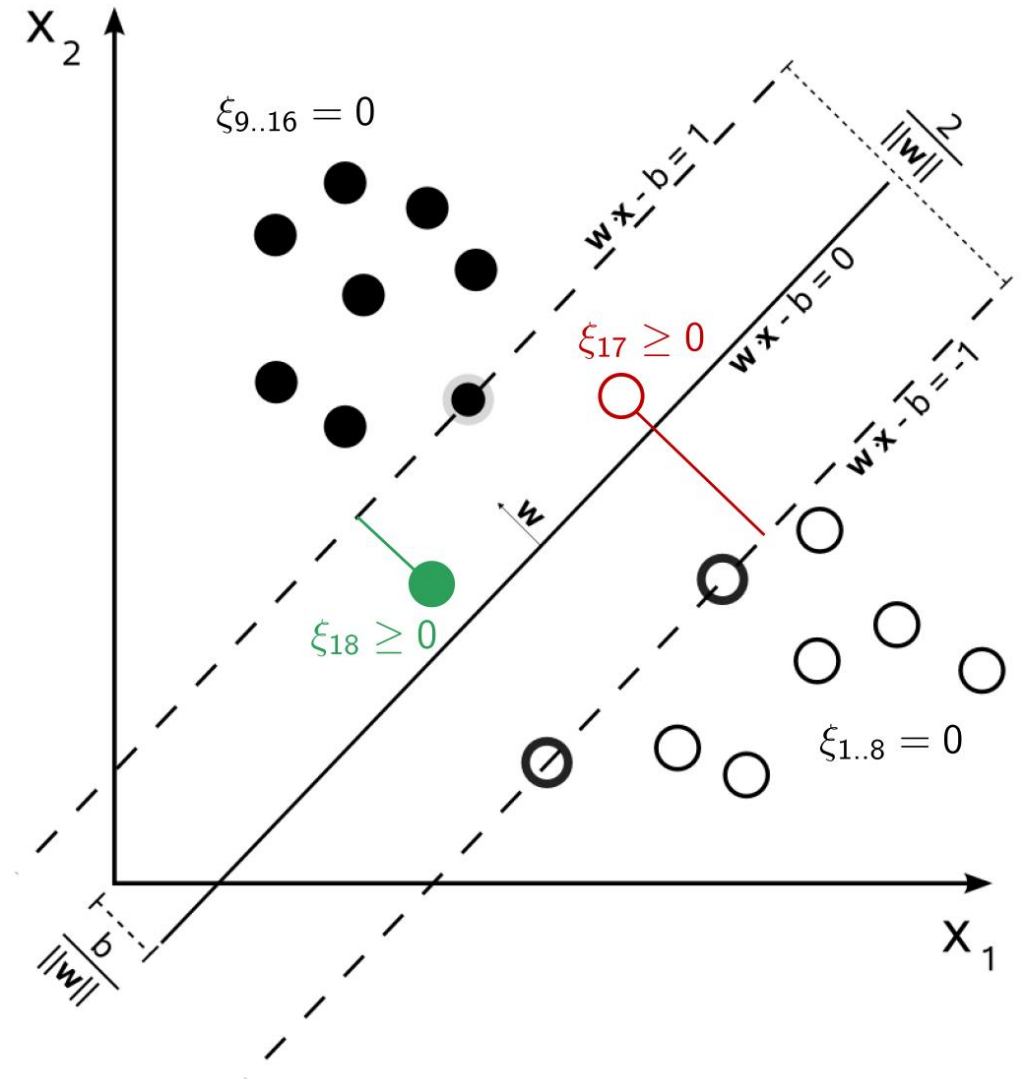
GD vs SGD



Support Vector Machine

Support Vector Machine (SVM)

- Není pravděpodobnostní model
- Max-margin klasifikátor: pokud jsou třídy lineárně separovatelné, vždy najde optimální nadplochu, která nejen odděluje, ale je zároveň co nejdále od obou tříd
- Pokud třídy nejsou lineárně separovatelné, zavádí se tzv. slack variables $\xi_n \geq 0$
- Pro některé body tedy podmínka “dokonalé klasifikace” nemusí být splněna



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

Hinge loss

- Soft margin SVM loss se slack variables

$$\begin{array}{ll}\text{minimize}_{\theta, \xi_n} & \|\mathbf{w}\|^2 + \lambda' \sum_{n=1}^N \xi_n \\ \text{subject to} & y_n s_n \geq 1 - \xi_n \\ & \xi_n \geq 0 \\ & n = 1, \dots, N\end{array}$$

- Podmínky lze sloučit do jednoho výrazu

$$\xi_n = L_n = \max(0, 1 - y_n s_n)$$

a dosadit kritéria \rightarrow vznikne **hinge loss**

- Člen $\|\mathbf{w}\|^2$ funguje jako regularizace

Weston-Watkins multiclass hinge loss

- Upravuje hinge loss na

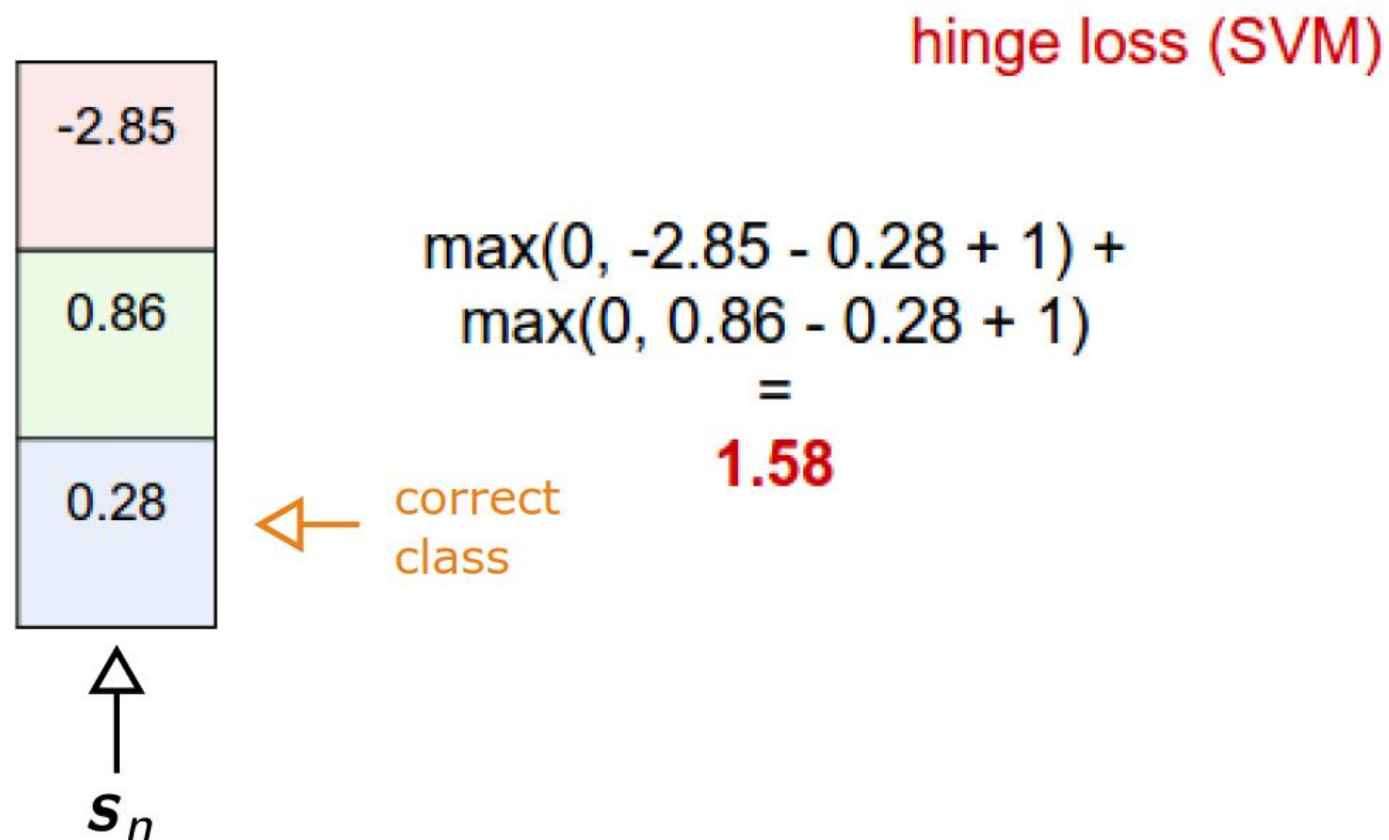
$$L_n = \sum_{c \neq y_n} \max(0, 1 + \textcolor{red}{s_{nc}} - s_{ny_n})$$

- Při více třídách je skóre vektor
- Aby $L_n \rightarrow 0$, musí:
 - skóre požadované třídy s_{ny_n} být co nejvyšší
 - skóre všech $c = 1, \dots, C, c \neq y_n$ ostatních tříd s_{nc} co nejnižší

binární varianta

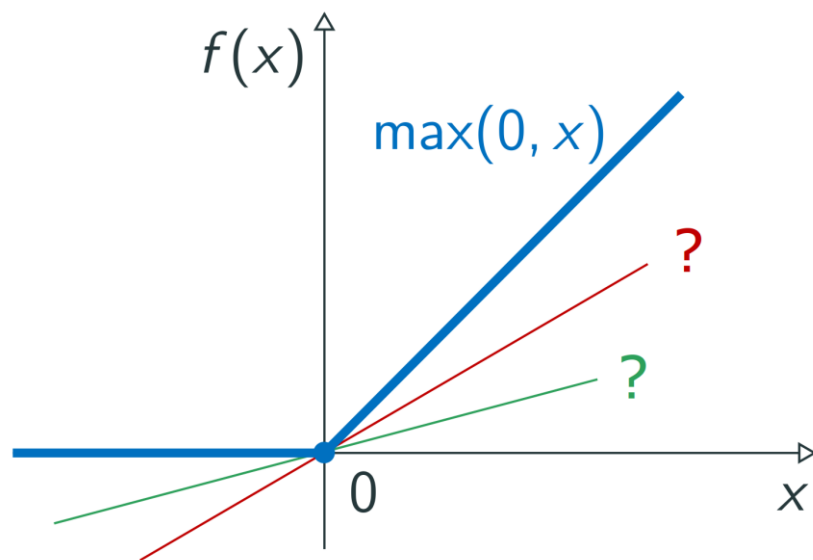
$$L_n = \max(0, 1 - y_n s_n)$$

Weston-Watkins multiclass hinge loss příklad



Subgradient

- Funkce $\max(0, x)$ není diferencovatelná
- Problém “bod zlomu” v nule:



- Řeší tzv. subgradient \rightarrow prostě vybereme jednu z možných variant
- Např. v nule bude gradient nula
- (Sub)gradient tedy může být:

$$\frac{\partial}{\partial x} \max(0, x) = \begin{cases} 0 & \text{když } x \leq 0 \\ 1 & \text{když } x > 0 \end{cases}$$

(Sub)gradient hinge kritéria

$$L_n = \sum_{i \neq y_n} \max(0, \underbrace{1 + s_{ni} - s_{ny_n}}_{\zeta_{ni}})$$

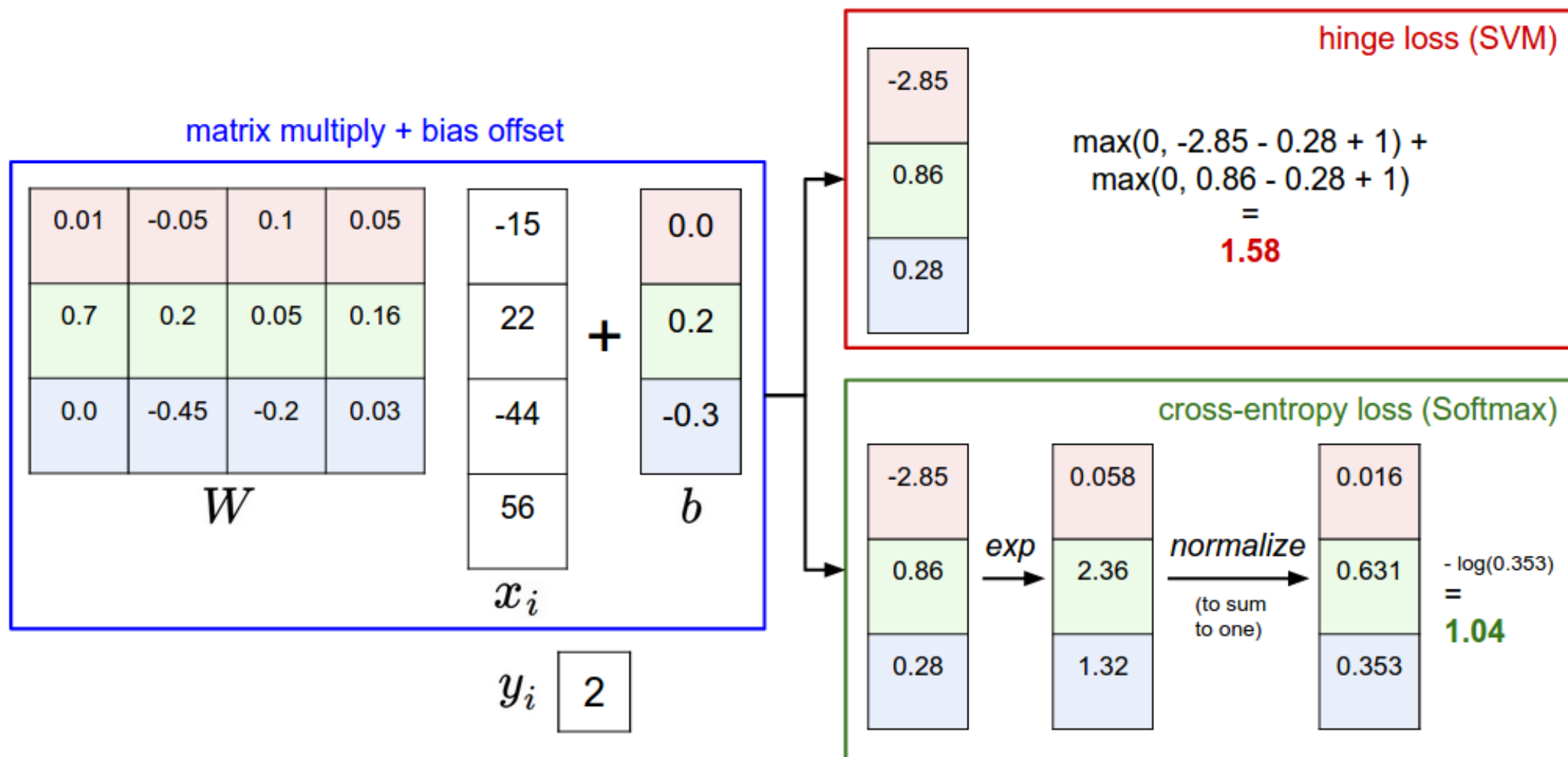
- Všimněme si, že obě skóre s_{nc} závisí pouze na c -tém řádku \mathbf{W} a \mathbf{b}
- Gradient na c -tý řádek \mathbf{W} :

$$\frac{\partial L_n}{\partial \mathbf{w}_c} = \begin{cases} - \sum_{i \neq y_n} \mathbb{1}(\zeta_{ni} > 0) \mathbf{x}_n & \text{pokud } c = y_n \\ \mathbb{1}(\zeta_{nc} > 0) \mathbf{x}_n & \text{pokud } c \neq y_n \end{cases}$$

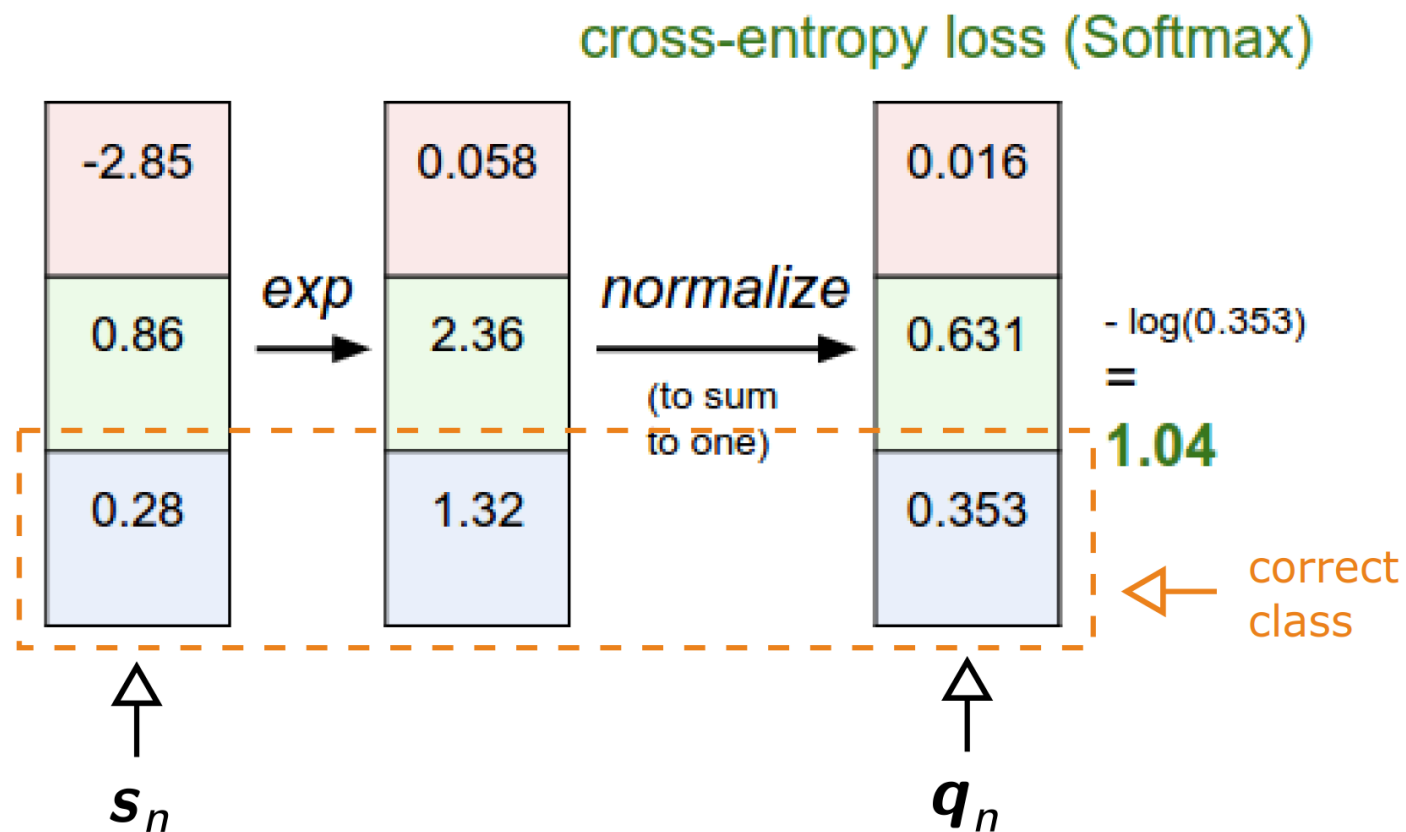
kde $\mathbb{1}(\text{podmínka}) = 1$, pokud je podmínka splněna, jinak 0

- Pro biasy podobně, pouze bez \mathbf{x}_n

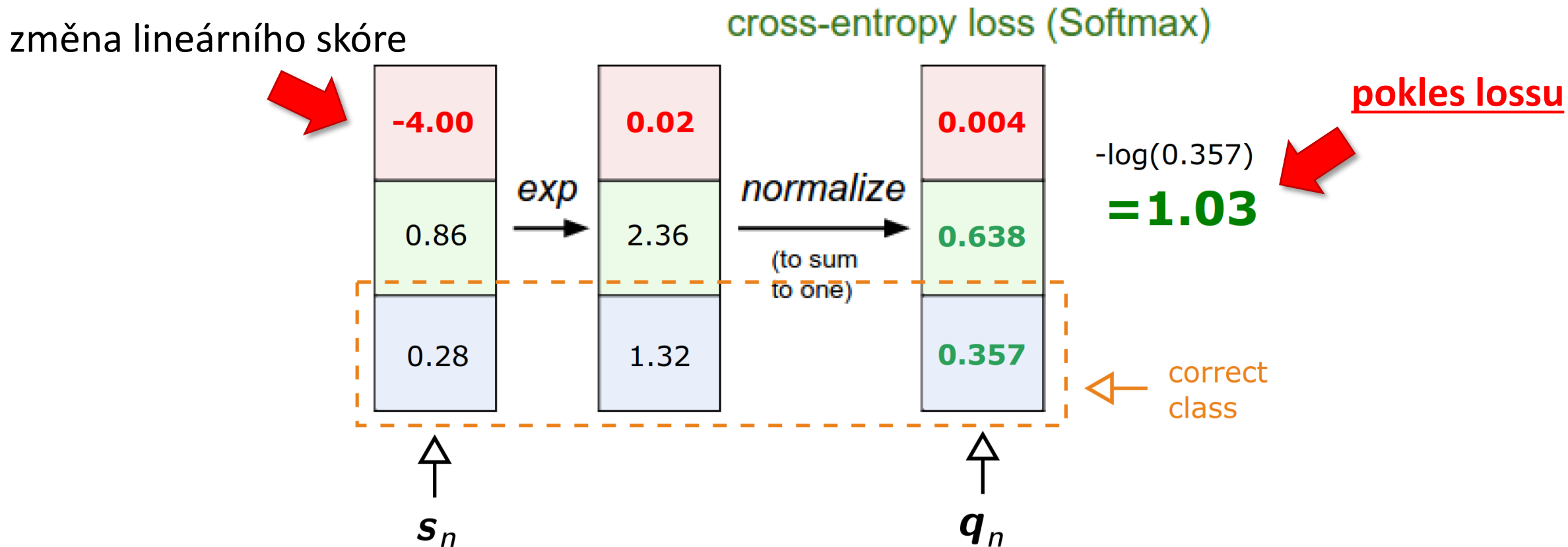
Cross entropy a hinge loss



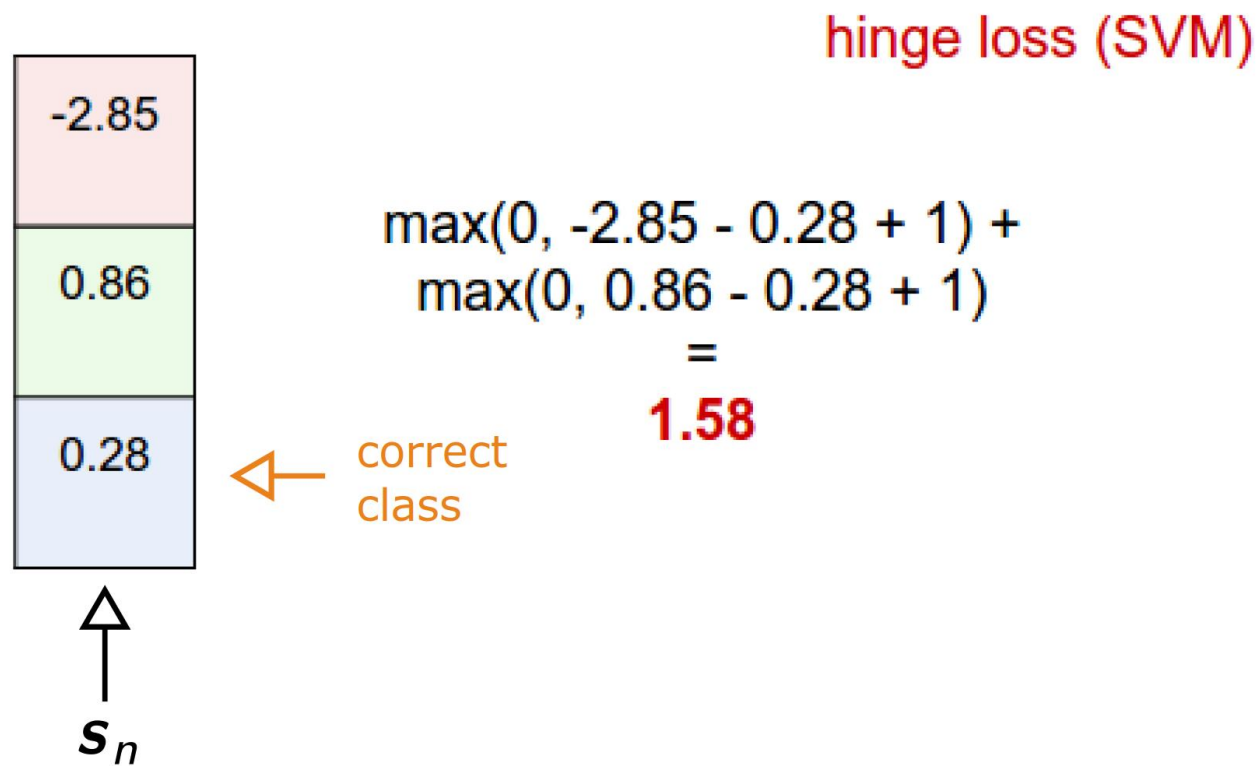
Citlivost cross entropy na změnu skóre



Citlivost cross entropy na změnu skóre

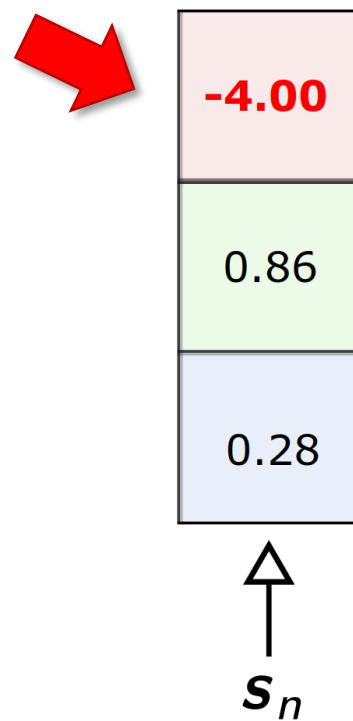


Citlivost hinge lossu na změnu skóre



Citlivost hinge lossu na změnu skóre

změna lineárního skóre



correct
class

hinge loss (SVM)

$$\begin{aligned} &\max(0, -4.00 - 0.28 + 1) + \\ &\max(0, 0.86 - 0.28 + 1) \\ &= \\ &1.58 \end{aligned}$$

loss nezměněn

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 klasifikace obě kritéria přibližně stejně dobrá
- Díky robustnosti vůči outlierům na menších datasetech výkonnější spíše SVM
- Overhead způsobený funkcí softmax je především u hlubokých sítí zanedbatelný

Shrnutí

- Diskriminativní klasifikace je vlastně jen minimalizace funkce
- Funkce kvantifikuje, jak moc špatný náš klasifikátor je → tzv. loss či kritérium
- Proměnná, vůči které minimalizujeme, jsou tedy parametry klasifikátoru
- Funkce může být libovolně složitá, avšak mělo by být snadné spočítat její gradient
- Díky tomu můžeme minimalizovat pomocí metody největšího spádu (GD)
- Gradient není nutné počítat úplně, efektivnější je aproximace a častější update (SGD)
- Logistická regrese a SVM jsou obojí lineární klasifikátory
- Liší se pouze kritériem

Shrnutí: trénování pomocí SGD

Inicializujeme:

- $\theta = \{W, b\}$ na náhodné hodnoty

Opakujeme:

1. navzorkování dávky (batch)
 2. forward + cache
 3. backward (gradient)
 4. update s krokem γ
- } různé pro LR a SVM

Zastavíme:

- po fixním počtu iterací
- parametry θ se ustálí
- hodnota kritéria již delší dobu neklesá

Shrnutí: logistická regrese (softmax klasifikátor) a SVM

Softmax LR	Křížová entropie $L_n = -\log q_{y_n}$ $y_n \in \{1, \dots, C\}$	$\frac{\partial L_n}{\partial \mathbf{w}_c} = (q_{nc} - p_{nc})\mathbf{x}_n,$ $\frac{\partial L_n}{\partial b_c} = q_{nc} - p_{nc}$ $\mathbf{q}_n = [q_{n1}, \dots, q_{nC}] = \text{Softmax}(\mathbf{s}_n)$
Multiclass SVM	Weston-Watkins hinge loss $L_n = \sum_{i \neq y_n} \max(0, 1 + s_{ni} - s_{ny_n})$ $y_n \in \{1, \dots, C\}$	$\frac{\partial L_n}{\partial \mathbf{w}_c} = \begin{cases} -\sum_{i \neq y_n} \mathbb{1}(1 + s_{ni} - s_{ny_n} > 0)\mathbf{x}_n & \text{když } c = y_n \\ \mathbb{1}(1 + s_{nc} - s_{ny_n} > 0)\mathbf{x}_n & \text{když } c \neq y_n \end{cases}$ $\frac{\partial L_n}{\partial b_c} = \begin{cases} -\sum_{i \neq y_n} \mathbb{1}(1 + s_{nc} - s_{ny_n} > 0) & \text{když } c = y_n \\ \mathbb{1}(1 + s_{nc} - s_{ny_n} > 0) & \text{když } c \neq y_n \end{cases}$