

Zpracování obrazu pomocí neuronových sítí

Blok 1: Úvod do zpracování obrazu a neuronových sítí

Michal Vašinek

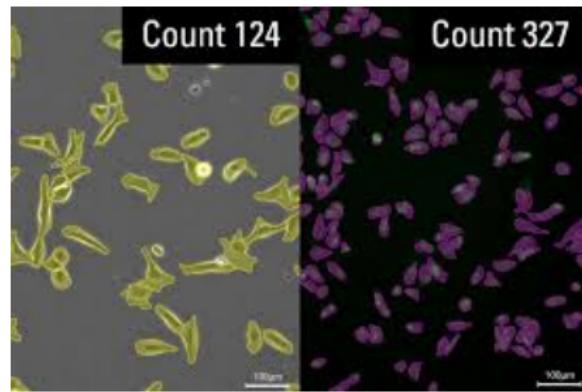
25.4.2025

Motivace a úvod

- Zpracování obrazu je v biologii klíčové: mikroskopická data, analýza buněk, detekce struktur
- Potřeba automatizace: manuální analýza je časově náročná a subjektivní
- Ukázky úloh:
 - **Počítání buněk** (např. během kultivace nebo screening testů)
 - **Detekce struktur** (např. jádra, mitochondrie)
 - **Klasifikace mikroorganismů** (např. podle tvaru nebo fluorescenční značky)

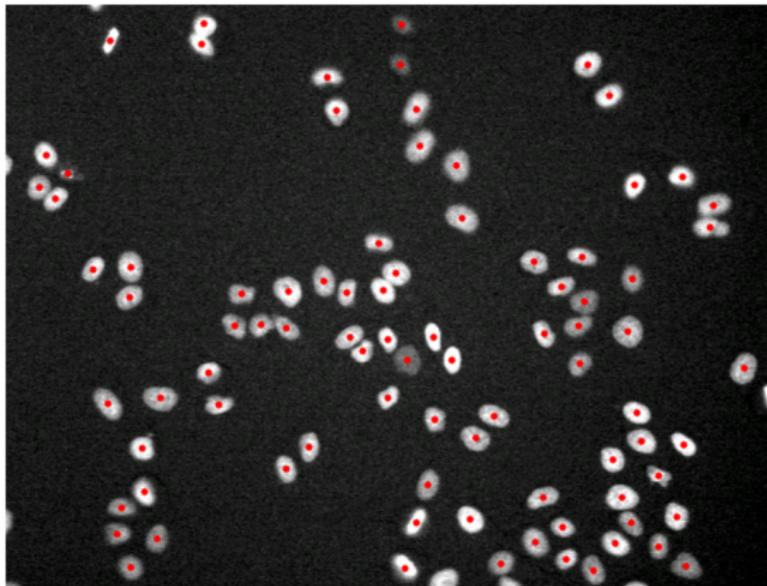
Ukázka: Počítání buněk

- Automatické počítání buněk ve snímcích z mikroskopu
- Výhody oproti manuálnímu značení: rychlosť, menší chybovost



Ukázka: Detekce struktur

- Identifikace organel (např. jádro, mitochondrie)
- Pomocí prahování a segmentace



Ukázka: Klasifikace mikroorganismů

- Rozpoznání druhu mikroorganismu dle obrazu
- Aplikace v diagnostice, výzkumu, ekologii

Class : Rod_bacteria
Pred : Rod_bacteria



Class : Paramecium
Pred : Paramecium



Class : Euglena
Pred : Euglena



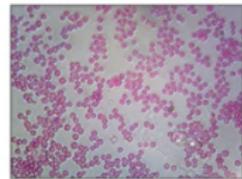
Class : Yeast
Pred : Yeast



Class : Rod_bacteria
Pred : Rod_bacteria

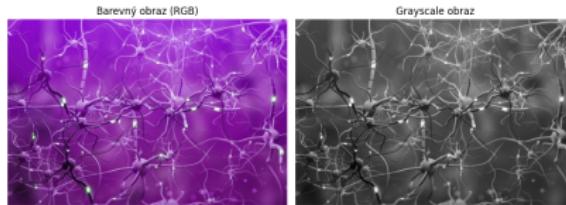


Class : Spherical_bacteria
Pred : Spherical_bacteria



Obraz jako matice

- Digitální obraz je tvořen mřížkou bodů (pixelů), každý pixel nese informaci o jasu nebo barvě.
- **Grayscale (černobílý) obraz:** každý pixel má hodnotu od 0 (černá) do 255 (bílá).
- **RGB obraz:** každý pixel obsahuje tři hodnoty – červenou (R), zelenou (G) a modrou (B), každá v rozsahu 0–255.
- Obraz tedy reprezentujeme jednou hodnotou (grayscale) nebo třemi hodnotami (RGB) matici.



Mikroskopické obrazy a formáty souborů

- Mikroskopy typicky produkují snímky v odstínech šedi (grayscale).
- Důvod: zachycení intenzity světla nebo fluorescence v jedné vlnové délce.
- Používají se hlavně tyto formáty:
 - **TIFF (.tif)** – bezztrátový, podporuje více kanálů a hloubek, vhodný pro vědecké účely.
 - **PNG (.png)** – bezztrátový, vhodný pro prezentace, ne vždy podporuje metadata.
 - **JPG (.jpg)** – ztrátová komprese, menší soubor, ale nevhodný pro analýzu dat.
- TIFF je nejčastěji používaný ve výzkumu – zachovává kvalitu a metadata (např. měřítka).

Histogram, kontrast, jas

- **Histogram** je graf, který ukazuje, kolik pixelů v obrázku má určitou úroveň jasu (od 0 do 255).
- Pomáhá pochopit rozložení světlých a tmavých oblastí v obrazu.
- Pokud je většina pixelů tmavá, histogram bude mít vrchol vlevo, u světlých obrazů vpravo.
- Může odhalit špatně nasvětlený nebo kontrastní snímek.
- Na základě histogramu lze upravit kontrast a jas, aby byly detaily lépe viditelné.
- V biologii často potřebujeme zvýraznit struktury – histogram pomáhá zjistit, zda je potřeba obraz zpracovat.

Prahování – jednoduchá segmentace

- **Prahování** je proces, kdy se obraz převede na černobílý (binární) podle daného prahu.
- Každý pixel s jasem vyšším než práh je nastaven na bílou (1), ostatní na černou (0).
- Cílem je oddělit objekty (např. buňky) od pozadí.
- Používá se, když jsou objekty dobře kontrastní vůči pozadí.
- Existují dva typy:
 - **Statické prahování:** pevně daný práh (např. 128).
 - **Adaptivní prahování:** práh se počítá lokálně pro každou část obrazu.
- Nevhodné pro obrazy s nehomogenním osvětlením.

Adaptivní prahování podrobněji

- U adaptivního prahování se hodnota prahu stanovuje individuálně pro každou oblast (okno) v obrazu.
- Vhodné pro snímky s nerovnoměrným osvětlením nebo stínováním.
- Typy adaptivního prahování:
 - **Průměr okolí (mean)** – práh je dán průměrnou hodnotou v okolním okně.
 - **Gaussovo vážené průměrování (Gaussian)** – větší váha je dána blízkým pixelům.
- Parametry: velikost okna (např. 11×11), konstanta C, která se odečítá od výpočtu.
- Výhoda: umí zvýraznit detaily v různě osvětlených částech snímku.
- Nevýhoda: citlivost na velikost okna a šum v obrazu.

Gaussův filtr – rozmazání obrazu

- Gaussův filtr je metoda pro **potlačení šumu** v obraze pomocí rozmazání.
- Používá konvoluci obrazu s gaussovskou funkcí (zvoncovitý profil).
- Efektivně vyhlazuje přechody a snižuje detail – vhodný jako předzpracování pro segmentaci nebo detekci hran.
- Parametry:
 - Velikost jádra (např. 3x3, 5x5) – určuje rozsah rozmazání.
 - Směrodatná odchylka σ – určuje šířku filtru.
- Výhody: jednoduchost, rychlosť, dobré potlačení šumu.
- Nevýhoda: může rozmazat jemné detaily objektů.

Sobelův filtr – detekce hran

- Sobelův filtr slouží k **detekci hran** v obrazu – oblastí s náhlou změnou jasu.
- Funguje na základě výpočtu derivace obrazu (změny intenzity).
- Využívá dva filtry – pro horizontální a vertikální směr (Sobel X a Sobel Y).
- Kombinací obou vznikne gradientová mapa – detekce hran v různých směrech.
- Časté využití pro zvýraznění kontur buněk nebo struktur.
- Nevýhoda: citlivost na šum – často se kombinuje s rozmazáním (např. Gauss).

Mediánový filtr – odstranění šumu

- Mediánový filtr slouží k **odstranění impulsního šumu** z obrazu.
- Na rozdíl od Gaussova filtrov nezpůsobuje rozmazání hran.
- Každý pixel se nahradí **mediánem** hodnot ze svého okolí (např. 3x3 nebo 5x5).
- Vhodný pro biologické snímky, kde je důležité zachovat tvary buněk a kontury.
- Výhody:
 - zachování hran a struktur,
 - účinný proti „salt and pepper“ šumu.
- Nevýhoda: méně účinný u jiných typů šumu (např. Gaussovského).

Otsuovo prahování

- Otsuova metoda automaticky najde optimální práh pro rozdělení histogramu na dvě třídy (pozadí a objekt).
- Práh se volí tak, aby minimalizoval rozptyl uvnitř tříd (a maximalizoval rozptyl mezi nimi).
- Vhodné, když histogram ukazuje dvě výrazné skupiny jasových hodnot.
- Výhoda: není třeba ručně nastavovat práh.
- Nevýhoda: nefunguje dobře, pokud se objekty a pozadí překrývají nebo nemají dostatečný kontrast.

Morfologické operace

- Slouží k úpravám tvaru binárních objektů po segmentaci (např. po prahování).
- Dvě základní operace:
 - **Eroze:** zmenšuje objekty, odstraňuje malé detaily.
 - **Dilatace:** rozšiřuje objekty, vyplňuje mezery.
- Kombinace:
 - **Otevření (opening):** eroze + dilatace – odstraní šum.
 - **Zavření (closing):** dilatace + eroze – vyplní mezery.
- Užitečné pro čištění obrazu před detekcí objektů nebo počítáním buněk.
- Vhodné použít po binarizaci obrazu.

Ukázka v Pythonu (Jupyter)

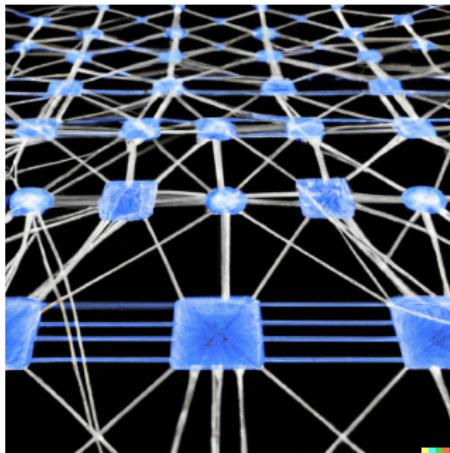
- Knihovny: OpenCV, scikit-image, matplotlib
- Načtení a zobrazení obrázku
- Úprava kontrastu a prahování

```
import cv2, matplotlib.pyplot as plt
```

Co jsou to neuronové sítě?

Neuronová síť

Systém vzájemně propojených neuronů, který může být trénován a použit pro rozpoznávání vzorů, predikci, klasifikaci a další úkoly.

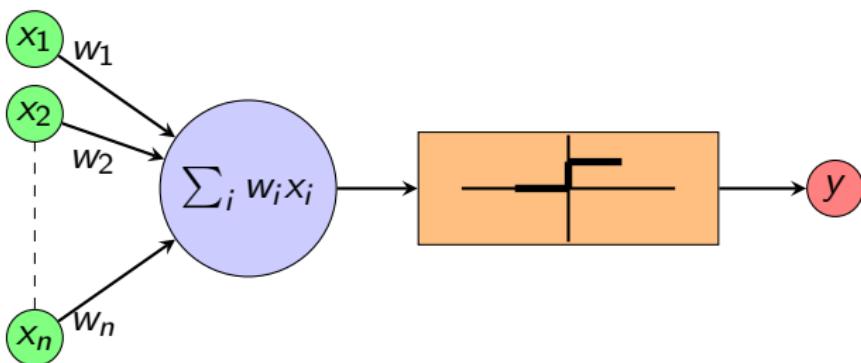


Obrázek: Obrázek vygenerovaný DALL-E 2 od OpenAI. Dotaz: umělá neuronová síť.

Základní pojmy – Neuron

Neuron

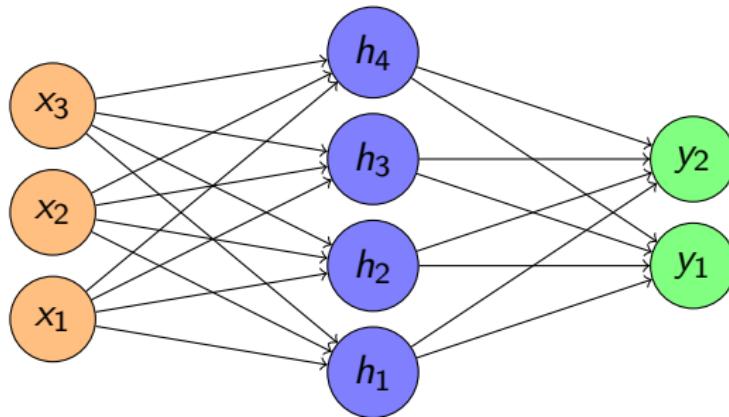
Základní stavební jednotka neuronových sítí, zjednodušený model buňky lidského mozku.



Obrázek: Model umělého neuronu podle McCullougha a Pittse.

Architektury

Neuronové sítě mohou mít různá uspořádání neuronů do funkčních hierarchií, včetně jedno-vrstvých perceptronů a složitých hlubokých neuronových sítí.

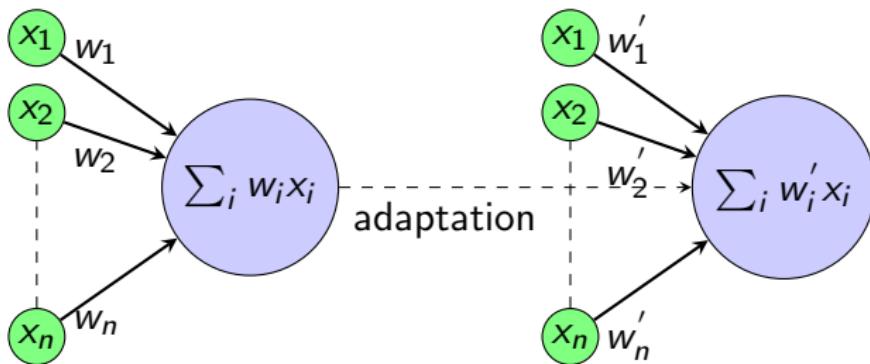


Obrázek: Plně propojené neurony s jednou skrytou vrstvou a dvěma výstupními neurony.

Základní pojmy – Učení

Učení

- Neuronové sítě se učí ze školicích dat pomocí algoritmů, jako je zpětná propagace (Backpropagation).
- Učení neuronové sítě je založeno na přizpůsobování vah za účelem snížení rozdílu mezi očekávaným a predikovaným výstupem.

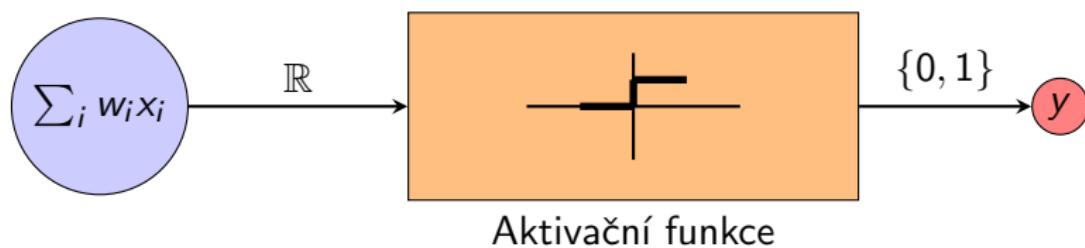


Obrázek: Učení je proces přizpůsobování vah v neuronové síti.

Základní pojmy – Aktivační funkce

Aktivační funkce

Každý neuron používá aktivační funkci k určení výstupu na základě vážených vstupů a prahové hodnoty.



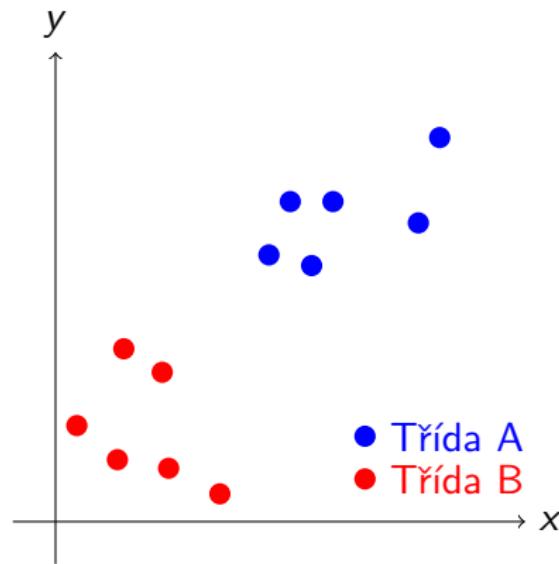
Obrázek: Aktivační funkce převádí prostor vstupních hodnot na prostor výstupních hodnot. V tomto případě převádí množinu reálných čísel \mathbb{R} na množinu $\{0, 1\}$.

Neuronové sítě – Historický vývoj

- **1943:** Model neuronu McCulloch-Pitts
- **1957:** Frank Rosenblatt představil perceptron
- **1980s:** Zpětná propagace (Backpropagation)
- **1990s:** SVM (Support Vector Machines)
- **2000s:** Hluboké neuronové sítě – Geoffrey Hinton
- **2010s:** Hluboké učení (Deep Learning), vítěz soutěže ImageNet
- **2020s:** Neuronové sítě všude – autonomní vozidla, velké jazykové modely (OpenAI – ChatGPT, Google – Gemini), překlady založené na hlubokém učení (DeepL)

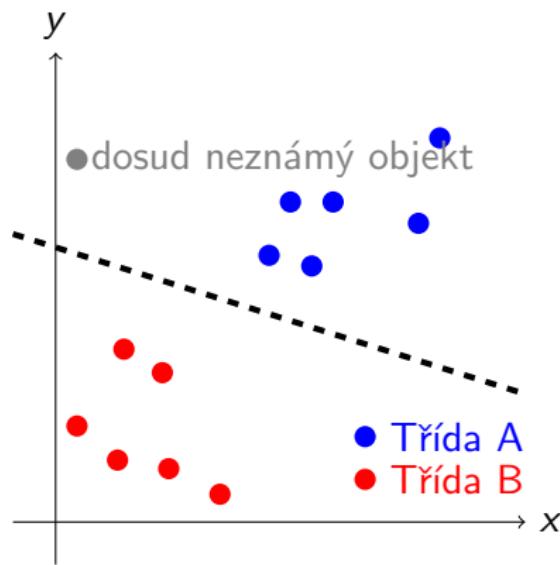
Binární klasifikace

Představme si, že máme záznamy reprezentující dvě třídy objektů a že jsme u každého objektu změřili dvě hodnoty x a y . Cílem je najít přímku, která co nejlépe oddělí tyto dvě třídy.



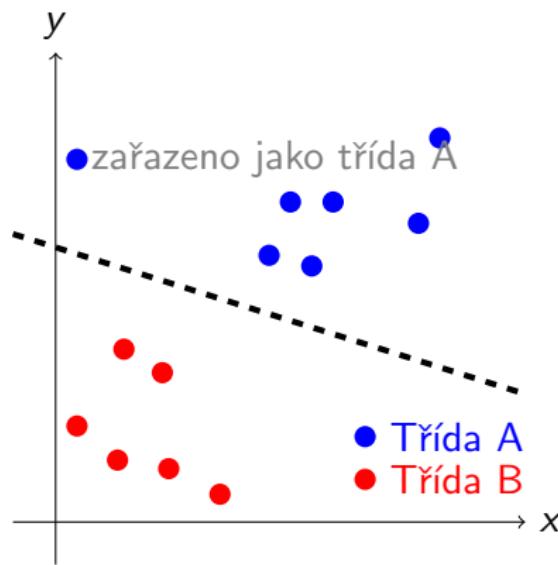
Binární klasifikace

Pokud dokážeme nalézt oddělující přímku, mohli bychom automaticky klasifikovat záznamy podle hodnot x a y .



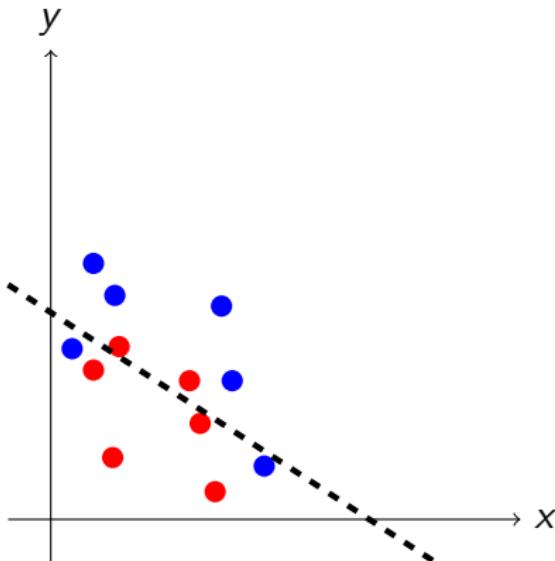
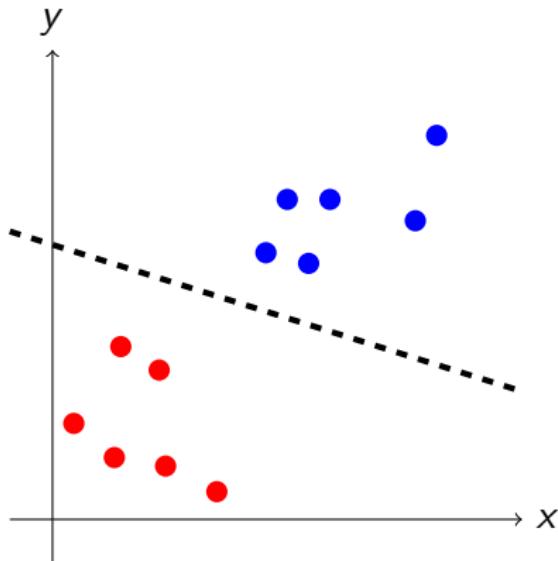
Binární klasifikace

Pokud se objekt nachází nad oddělující přímkou, je zařazen do třídy A, v opačném případě do třídy B.



Lineárne separovateľná množina dat

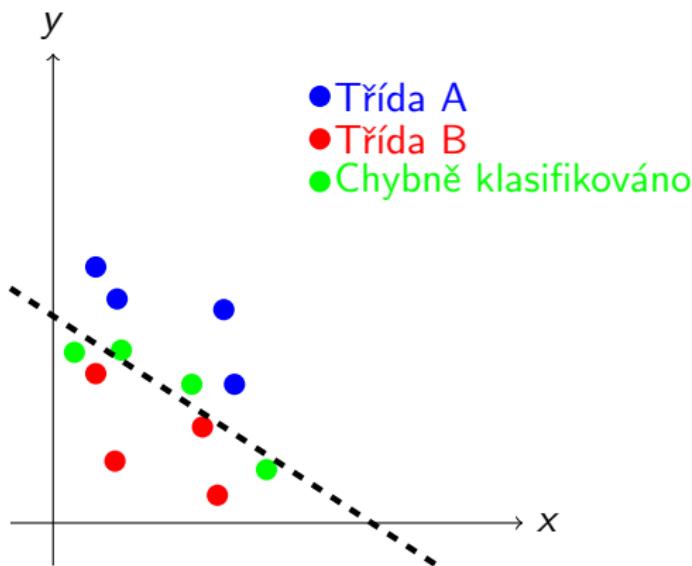
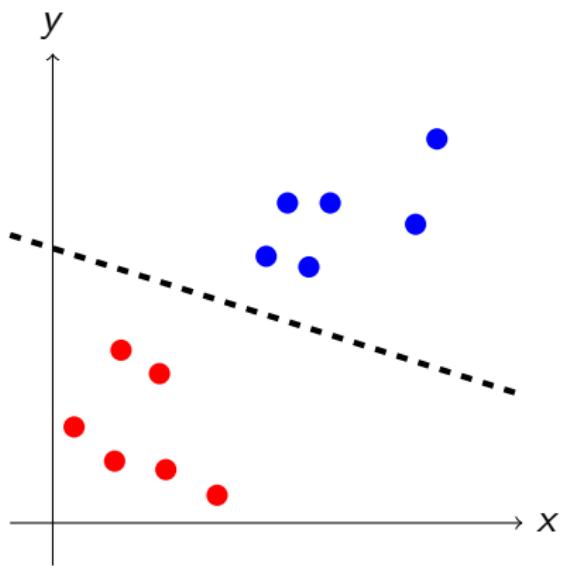
- **Lineárne separovateľná** – jsme schopni najíť oddelujúci priamku.
- **Lineárne neseparovateľná** – nejsme schopni najíť oddelujúci priamku.



Lineárne neseparovateľná množina dat

Chybně klasifikované záznamy

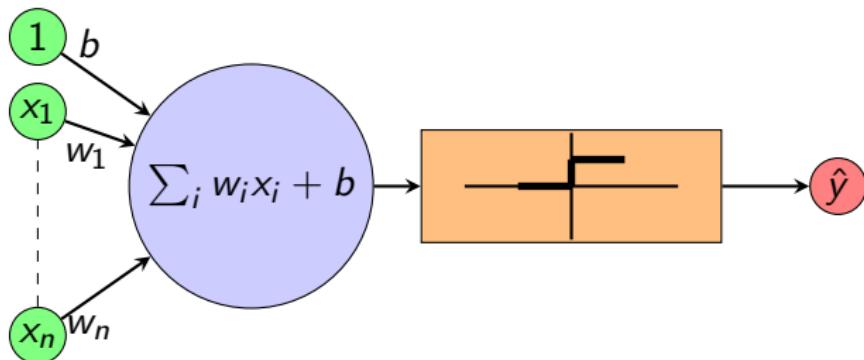
Pokud se pokusíme oddeliť objekty z lineárne neseparovateľnej množiny dat pomocí priamy, vzniknou chybně klasifikované objekty.



Perceptron

Definice

Perceptron je nejjednodušší forma jednotky neuronové sítě, schopná provádět úlohy binární klasifikace.



Obrázek: Model perceptronu.

- **Matematický zápis:** Výstup \hat{y} perceptronu pro vstupní vektor $\mathbf{x} = (x_1, x_2, \dots, x_n)$ a váhy $\mathbf{w} = (w_1, w_2, \dots, w_n)$ se vypočítá podle následujícího vzorce:

$$\hat{y} = \text{Step} \left(\sum_{i=1}^n x_i \cdot w_i + b \right)$$

kde b je parametr posunu (bias) a $\text{Step}(z)$ je skoková funkce definovaná takto:

$$\text{Step}(z) = \begin{cases} 1, & \text{pokud } z > 0 \\ 0, & \text{jinak} \end{cases}$$

- **Rozhodovací hranice:** Perceptron klasifikuje vstupy na základě lineární rozhodovací hranice ve vstupním prostoru.

Perceptron – průchod vpřed: příklad

Příklad – výchozí parametry

Uvažujme perceptron se dvěma vstupními příznaky $x_1 = 1,2$ a $x_2 = -0,7$, váhami w_1 , w_2 a posunem b . Váhy a posun: $w_1 = 0,6$, $w_2 = -0,8$, $b = -0,5$.

- **Výpočet průchodu vpřed:** Vážený součet vstupů a posunu se spočítá následovně:

$$z = x_1 \cdot w_1 + x_2 \cdot w_2 + b$$

- **Aktivace:** Použitím skokové funkce určíme výstup \hat{y} :

$$\hat{y} = \text{Step}(z) = \begin{cases} 1, & \text{pokud } z > 0 \\ 0, & \text{jinak} \end{cases}$$

- **Příklad výpočtu:** Pro $x_1 = 1,2$ a $x_2 = -0,7$ spočítáme

$$\hat{y} = \text{Step}(1,2 \cdot 0,6 + (-0,7) \cdot (-0,8) - 0,5) = \text{Step}(0,78) = 1$$

Cíl

Natrénovat perceptron tak, aby správně klasifikoval vstupní vzory úpravou svých vah (w) a posunu (b).

- **Aktualizační pravidlo:** Váhy a posun se upravují pomocí perceptronového učícího algoritmu. Pro chybně klasifikovaný vstup (x, y) , kde y je cílový výstup (0 nebo 1) a \hat{y} je predikovaný výstup, platí následující pravidlo:

$$w_i \leftarrow w_i + \alpha \cdot (y - \hat{y}) \cdot x_i$$

$$b \leftarrow b + \alpha \cdot (y - \hat{y})$$

kde α je učící rychlosť (learning rate) a x_i je i -tá složka vstupného vektoru x .

- **Dopad:** Toto pravidlo posouvá rozhodovací hranici blíže ke špatně klasifikovanému bodu, čímž zlepšuje přesnost klasifikace perceptronu.

Perceptron – aktualizace vah: příklad

- **Cíl:** Natrénovat perceptron ke správné klasifikaci vstupů. Uvažujme chybně klasifikovaný vstup $\mathbf{x} = (1, 2, -0,7)$ s cílovým výstupem $y = 0$.
- **Počáteční váhy a posun:** $w_1 = 0,6$, $w_2 = -0,8$, $b = -0,5$.
- **Predikce:** Spočítejme vážený součet plus posun:

$$z = 1,2 \times 0,6 + (-0,7) \times (-0,8) + (-0,5) = 0,78$$

- **Chybná klasifikace:** Predikovaný výstup $\hat{y} = \text{Step}(0,78) = 1$ neodpovídá cílovému výstupu $y = 0$.
- **Aktualizace vah:** Aktualizujme váhy pomocí učící rychlosti $\alpha = 0,1$ a perceptronového pravidla učení:

$$w_1 \leftarrow 0,6 + 0,1 \cdot (0 - 1) \cdot 1,2 = 0,48$$

$$w_2 \leftarrow -0,8 + 0,1 \cdot (0 - 1) \cdot (-0,7) = -0,73$$

$$b \leftarrow -0,5 + 0,1 \cdot (0 - 1) = -0,6$$

Perceptron – aktualizace vah: příklad

- **Aktualizace vah:** Aktualizujme váhy pomocí učící rychlosti $\alpha = 0,1$ a perceptronového pravidla učení:

$$w_1 \leftarrow 0,6 + 0,1 \times (0 - 1) \times 1,2 = 0,48$$

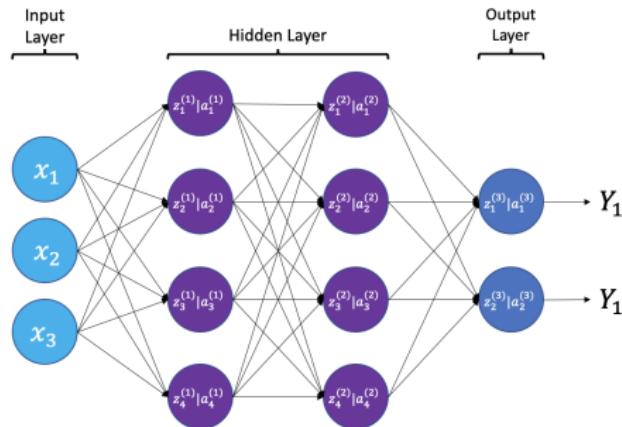
$$w_2 \leftarrow -0,8 + 0,1 \times (0 - 1) \times (-0,7) = -0,73$$

$$b \leftarrow -0,5 + 0,1 \times (0 - 1) = -0,6$$

- **Další predikce:** $z = 1,2 \times 0,48 + (-0,7) \times (-0,73) + (-0,6) = 0,487$
- Výstup váženého součtu před aktualizací vah byl 0,78, po úpravě vah 0,487.
- Stále se jedná o chybně klasifikovaný případ, ale výsledek je blíže očekávané hodnotě.

Hustě propojené neuronové sítě (DNN)

- Hustě propojené neuronové sítě (DNN), se skládají z vrstev neuronů, které jsou vzájemně propojené.
- Každý neuron v jedné vrstvě je propojen se všemi neurony ve vrstvě následující.



Husté neuronové sítě (DNN) – Aplikace, ukázka

- Husté neuronové sítě jsou základem hlubokého učení a dokáží modelovat složité vzory v datech.
- Jsou široce využívány v různých oblastech, včetně rozpoznávání obrazu a řeči, zpracování přirozeného jazyka a hraní her.
- DNN používají aktivační funkce jako ReLU (Rectified Linear Unit) nebo sigmoid, které zavádějí nelinearitu a umožňují tak učení složitých vztahů v datech.

Úloha

Aktivační funkce zavádějí do neuronové sítě nelinearitu, což jí umožňuje učit se složité vzory v datech.

Vizualizace sigmoidální funkce

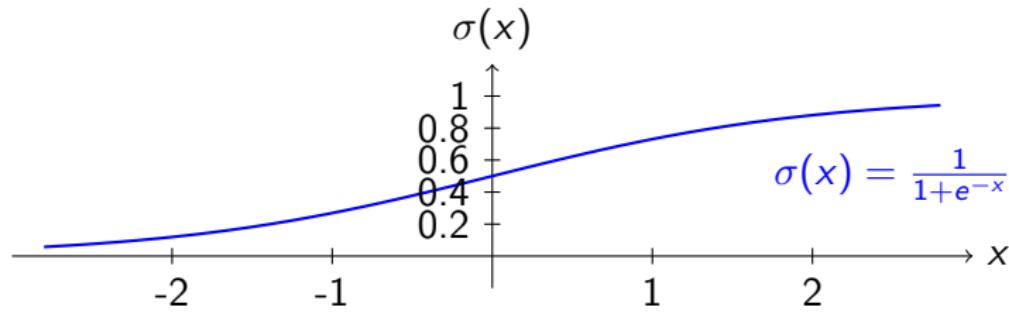
Sigmoid

$$\sigma(x) = \frac{1}{1+e^{-x}}$$

Rozsah: (0, 1)

Výstup: Stlačuje vstupní hodnoty do intervalu mezi 0 a 1.

Používaná ve starších sítích, problém s mizejícím gradientem.



Vizualizace ReLU funkce

ReLU (Rectified Linear Unit)

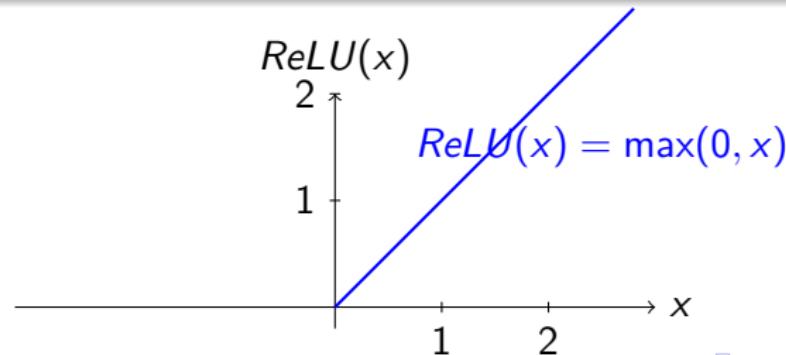
$$\text{ReLU}(x) = \max(0, x)$$

Rozsah: $[0, \infty)$

Výstup: Kladné hodnoty ponechá beze změny, záporné nastaví na 0.

Používá se k řešení problému mizejícího gradientu.

Součást moderních hlubokých neuronových sítí.



Úloha

Ztrátové funkce měří rozdíl mezi predikovanými hodnotami (výstup neuronové sítě) a skutečnými hodnotami (tzv. ground truth) ve školicích datech.

- **Střední kvadratická chyba (MSE):** $MSE = \frac{1}{N} \sum_{i=1}^N (y_i - \hat{y}_i)^2$
Měří průměrný čtvercový rozdíl mezi predikcemi (\hat{y}_i) a skutečnými hodnotami (y_i).
- **Binární křížová entropie:** $-\frac{1}{N} \sum_{i=1}^N [y_i \log(\hat{y}_i) + (1 - y_i) \log(1 - \hat{y}_i)]$
Používá se pro úlohy binární klasifikace, penalizuje odchylky od skutečných třídních štítků.
- **Kategorická křížová entropie:** $-\frac{1}{N} \sum_{i=1}^N \sum_{j=1}^C y_{ij} \log(\hat{y}_{ij})$
Používá se pro vícerádní klasifikaci, kde C je počet tříd.

Zpětná propagace (Backpropagation)

Definice

Zpětná propagace je algoritmus učení s učitelem, který se používá pro trénování umělých neuronových sítí. Jedná se o metodu efektivního výpočtu gradientů ztrátové funkce vzhledem k vahám v síti.

• Kroky:

- ① **Průchod vpřed (Forward Pass):** Spočítej predikovaný výstup neuronové sítě pomocí aktuálních vah.
- ② **Výpočet ztráty (Loss Computation):** Urči ztrátu mezi predikovaným výstupem a skutečnými cílovými hodnotami.
- ③ **Zpětný průchod (Backpropagation):** Vypočítej gradienty ztráty vůči parametrům sítě (vahám a posunům) pomocí pravidla pro derivaci složené funkce (chain rule).
- ④ **Gradientní sestup (Gradient Descent):** Aktualizuj váhy a posuny sítě pomocí vypočtených gradientů za účelem minimalizace ztráty.

Zpětná propagace (Backpropagation)

- **Klíčové pojmy:**
 - **Pravidlo pro derivaci složené funkce (Chain Rule):** Zpětná propagace využívá řetězové pravidlo z diferenciálního počtu k efektivnímu výpočtu gradientů v neuronové síti s více vrstvami.
 - **Rychlosť učení (Learning Rate):** Rychlosť učení je hyperparametr, ktorý určuje velikosť krokov provádzencích pri gradientném sestupe.
 - **Iterace:** Zpětná propagace zahrnuje vícenásobné iterace (epochy) pres celý trénovací dataset za účelem optimalizacie parametru sítě.

Zpětná propagace: Matematický zápis

Cíl:

Minimalizovat ztrátovou funkci L úpravou vah a posunů (biasů) neuronové sítě pomocí gradientního sestupu.

- **Odvození:** Pomocí řetězového pravidla se derivace ztráty vzhledem k vahám ve vrstvě l vypočítá následovně:

$$\frac{\partial L}{\partial W_{ij}^{(l)}} = \frac{\partial L}{\partial z_i^{(l+1)}} \cdot \frac{\partial z_i^{(l+1)}}{\partial W_{ij}^{(l)}}$$

$$\frac{\partial L}{\partial b_i^{(l)}} = \frac{\partial L}{\partial z_i^{(l+1)}}$$

kde $z_i^{(l+1)}$ je vstup do neuronu i ve vrstvě $l + 1$.

Zpětná propagace: Matematický zápis

- **Aktualizace vah:** Po výpočtu gradientů se váhy a posuny aktualizují pomocí gradientního sestupu:

$$W_{ij}^{(l)} = W_{ij}^{(l)} - \alpha \frac{\partial L}{\partial W_{ij}^{(l)}}$$

$$b_i^{(l)} = b_i^{(l)} - \alpha \frac{\partial L}{\partial b_i^{(l)}}$$

kde α je rychlosť učení (learning rate).

Zpětná propagace: Derivace ve ztrátové funkci

Ztrátová funkce: $L = \frac{1}{2}(y - \hat{y})^2$ (Střední kvadratická chyba)

Výstupní vrstva:

- **Predikovaný výstup:** \hat{y}
- **Cílový výstup:** y

Derivace ztráty podle predikovaného výstupu:

$$\frac{\partial L}{\partial \hat{y}} = \hat{y} - y$$

Zpětné šíření chyby:

- **Gradient ve výstupní vrstvě:** $\delta^{(výstup)} = \frac{\partial L}{\partial \hat{y}}$
- **Použití řetězového pravidla pro předchozí vrstvy:** Derivace se počítají zpětně sítí pomocí řetězového pravidla.

Zpětná propagace: Derivace pomocí řetězového pravidla

Neuron ve skryté vrstvě: $z = \sum_i (w_i \cdot x_i) + b$

Aktivační funkce: $a = \sigma(z)$ (např. Sigmoid)

Derivace ztráty podle výstupu aktivační funkce:

$$\frac{\partial L}{\partial a} = \frac{\partial L}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial a}$$

Derivace aktivační funkce:

$$\frac{\partial a}{\partial z} = a \cdot (1 - a)$$

(pro sigmoidální funkci)

Derivace ztráty podle výstupu neuronu:

$$\frac{\partial L}{\partial z} = \frac{\partial L}{\partial a} \cdot \frac{\partial a}{\partial z}$$

Derivace ztráty podle vah a posunu:

$$\frac{\partial L}{\partial w_i} = \frac{\partial L}{\partial z} \cdot x_i$$

$$\frac{\partial L}{\partial b} = \frac{\partial L}{\partial z}$$

Poznámka:

- Derivace se počítají pomocí řetězového pravidla pro šíření chyby zpět sítí.

Nedotrénování (Underfitting)

Definice

Nastává, když je model strojového učení příliš jednoduchý na to, aby zachytíl základní vzory v datech. Dosahuje špatných výsledků nejen na trénovacích datech, ale i na dosud neviděných (testovacích) datech.

- **Příčiny nedotrénování:**
 - Příliš málo vstupních příznaků nebo příliš jednoduchá architektura modelu.
 - Nedostatečné trénování nebo příliš silná regularizace.
- **Řešení:** Zvaž zvýšení složitosti modelu přidáním více vrstev nebo neuronů, přidej relevantní vstupní příznaky, nebo prodluž dobu trénování. Také sběr rozmanitějších a reprezentativnějších trénovacích dat může zlepšit schopnost modelu učit se daný problém.

Přetrénování (Overfitting)

Definice

Nastává, když se model strojového učení naučí trénovací data až příliš dobře, včetně šumu a odlehlých hodnot. Dosahuje výborných výsledků na trénovacích datech, ale špatných na testovacích, protože se příliš zaměřil na specifika trénovacích dat.

- **Příčiny přetrénování:**

- Příliš složitý model s příliš mnoha vstupními příznaky.
- Málo trénovacích dat nebo absence vhodné regularizace.

- **Řešení:** Najít rovnováhu mezi složitostí modelu a velikostí datové sady. Problémy s nedotrénováním a přetrénováním lze řešit pomocí technik, jako je křížová validace, regularizace nebo výběr příznaků.