

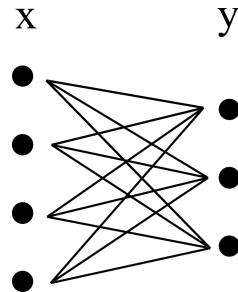
Úvod do Umelej Inteligencie Cvičenie 8 - Perceptrón

November 13, 2025

8. PERCEPTRÓN

Budeme programovať neurónovú sieť – jednovrstvový spojity perceptrón – ktorá sa naučí klasifikovať vstupné dátá do viacerých tried (kategórií).

Model:



Medzi každým vstupom x_j a výstupom y_i je spojenie s váhou w_{ij} – na začiatku sú tieto váhy náhodné, pričom počas učenia sa iteratívne vylepšujú. K n vstupným neurónom pridávame ešte jeden špeciálny x_{n+1} , tzv. *bias*, ktorý je vždy rovný 1.

Počítanie výstupu:

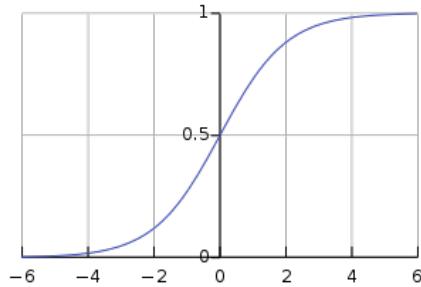
Výstup z perceptrónu sa počíta ako váhovaný súčet všetkých vstupov, ktorý sa preženie cez tzv. aktivačnú funkciu:

$$y_i = f\left(\sum_{j=1}^{n+1} w_{ij} \cdot x_j\right), \quad x_{n+1} = 1$$

Pokiaľ si vezmem vektor \mathbf{x} ako (stĺpcový) vektor všetkých vstupov a biasu $\mathbf{x} = (x_1, \dots, x_n, x_{n+1})^T$, d'alej vektor \mathbf{y} ako vektor všetkých výstupov $\mathbf{y} = (y_1, \dots, y_m)^T$ a maticu \mathbf{W} o veľkosti $(m \times (n+1))$, ktorá obsahuje všetky váhy w_{ij} , tak potom vieme vypočítať všetky výstupy naraz v jednom kroku, pomocou jednoduchého vzorca:

$$\mathbf{y} = f(\mathbf{W} \cdot \mathbf{x}),$$

V tomto prípade berieme, že funkcia f sa aplikuje na každý prvok výsledného vektora samostatne. Ako aktivačnú funkciu f budeme používať logistickú sigmoidu, ktorá má navyše jednoduchú deriváciu:



$$y = f(x) = \frac{1}{1 + e^{-x}} \quad f'(x) = y \cdot (1 - y)$$

Počítanie chyby:

Na trénovanie máme pripravené trénovacie množiny, ktorá obsahuje n -rozmerné vektory \mathbf{x} , a k nim prislúchajúce požadované výstupy – m -rozmerné vektory \mathbf{d} . Výpočet chyby je analogický minulému cvičeniu – opäť ide o regresnú chybu, akurát potrebujeme sumovať cez všetky súradnice našich bodov, t. j. pre jeden bod:

$$e = \sum_{i=1}^m (d_i - y_i)^2 = \|\mathbf{d} - \mathbf{y}\|^2$$

Celková chyba E je jednoducho súčtom (alebo priemerom) chýb e na všetkých príkladoch z trénovacej množiny.

Úprava váh:

Na minimalizáciu chyby používame *gradient descent* metódu, pri ktorej upravujeme váhy w_{ij} v opačnom smere ako je derivácia chyby e , z čoho nám vypadne vzorec:

$$w_{ij} := w_{ij} + \alpha \cdot (d_i - y_i) \cdot f'(\mathbf{w}_i^T \mathbf{x}) \cdot x_j$$

kde α je konštanta, tzv. rýchlosť učenia. Keď doplníme deriváciu sigmoidy do tohto vzorca (namiesto f'), dostaneme:

$$w_{ij} := w_{ij} + \alpha \cdot (d_i - y_i) \cdot y_i \cdot (1 - y_i) \cdot x_j$$

A opäť, aby sme nemuseli počítať všetky váhy w_{ij} osobitne, aj toto sa dá spraviť na jeden krok pomocou maticového zápisu:

$$\mathbf{W} := \mathbf{W} + \alpha \cdot (\boldsymbol{\delta} \times \mathbf{x}), \quad \delta_i = (d_i - y_i) \cdot y_i \cdot (1 - y_i)$$

Všimnite si, že na násobenie vektorov $\boldsymbol{\delta} \times \mathbf{x}$ sa používa tzv. outer product, t. j. výsledkom je matica.

Trénovanie:

Samotné trénovanie prebieha v iteráciach – epochách: v jednej epoce sa prejde všetkými príkladmi z trénovacej množiny, a pre každý z nich sa vypočíta výstup siete, jej chyba, a upravia sa váhy. Počas každej epochy sa príklady vyberajú v náhodnom poradí, aby sme sa vyhli nežiadúcim efektom pri učení. Končíme po určitom počte epoch, alebo keď celková chyba epochy E klesla dostatočne nízko.

Algorithm 1 Neural network training pseudocode:

```
1: procedure TRAINING
2:   Weight matrix  $W$  initialization
3:   while stopping criterion is not met do
4:      $E \leftarrow 0$ 
5:     for each input  $x$  and its target  $d$  in shuffle(inputs) do
6:        $y \leftarrow$  compute output for  $x$ 
7:        $e \leftarrow$  compute error of  $y$ 
8:        $E \leftarrow E + e$ 
9:     adjust weight matrix  $W$  using  $x$ ,  $y$  and  $d$ 
```

Dáta:

Budeme rozpoznávať čísllice, ktoré sú "nakreslené" na 4×7 pixeloch, napr.: (v dátach je čierna reprezentovaná jednotkou a biela nulou):



Takýchto 10 čísllic je nakopírovaných N -krát (parameter v programe, default je 10x), a následne sú mierne zašumené – každý "pixel" sa s malou pravdepodobnosťou prehodí. Takto získame bohatší dataset, na ktorom budeme trénovať rozpoznávanie (zašumených) čísllic.

Vstupný vektor x bude mať teda $4 \times 7 = 28$ zložiek (plus bias). Pri rozpoznávaní budeme číslice klasifikovať do jednej z desiatich tried 0 až 9, pričom budeme používať tzv. one-hot kódovanie výstupu: výstupný vektor y (alebo d) má 10 zložiek – všade sú nuly, iba tam, do ktorej kategórie x patrí, je jednotka. Napríklad číslica nula bude mať výstup $d=(1,0,0,0,0,0,0,0,0,0)$, sedmička zas $d=(0,0,0,0,0,0,1,0,0)$, atď.

Ked' budete siet' učiť, tak ako výstup siete nedostanete hodnoty presne nula a jedna, ale niečo medzi. "Vít'azný" neurón je potom ten s najvyššou hodnotou.

Napríklad výstup $y=(0.4,0.2,0.1,0.1,0.3,0.2,0.9,0.1,0.2)$ teda budeme chápať tak, že siet' rozpoznaла sedmičku. Máte pripravené dva vstupné súbory: v súbore *numbers.in* rozpoznávate číslice tak, ako je popísané vyššie. V súbore *odd_even.in* sú tiež číslice, no rozlišujete ich iba do dvoch tried – párne/nepárne (taktiež pomocou one-hot kódovania). Rozdelenie na párne a nepárne je ľahšia úloha – siet' by sa ju mala naučiť za niekoľko desiatok epoch. Rozpoznávanie čísel je t'ažšie – učenie môže trvať aj niekoľko stoviek epoch.

Úloha:

Do pripravenej kostry programu *perceptron.py* doprogramujte nasledovné pomocné funkcie, plus hlavný trénovací cyklus:

- *initialize_weights()* – inicializuje maticu váh $self.W$ (treba zadat' iba veľkosť matice)
- *sigmoid(x)* – vypočíta hodnotu sigmoidy, pričom x je číslo (skalár). Pokaľ funkciu spravíte využitím *np.exp*, tak x môže byť aj vektor a funkcia vráti opäť vektor.
- *compute_output(x)* – vypočíta výstup siete pre vstupný vektor x . **Výstupom je výsledný vektor y.**
- *compute_error(d, y)* – pomocou požadovaného výstupu d a vášho výstupu y (oba sú vektory) vypočíta (regresnú) chybu e . Výstupom je teda jedno číslo.
- *train(num_epoch)* – využitím predchádzajúcich funkcií natrénuje neurónovú siet'. Siet' sa bude učiť *num_epoch* epoch.

Numpy:

Ako ste si všimli, v tejto úlohe sa veľa hráme s vektormi a maticami. Preto budeme používať python-ovskú knižnicu *numpy*, vďaka ktorej je počítanie s vektormi a maticami veľmi jednoduché a pohodlné. Vďaka *numpy* teda netreba riešiť počítanie y , e , δ , či W po zložkách vo for-cykloch, všetko sa dá vypočítať na jeden riadok pomocou vektorov. Rýchlokurz *numpy* je aj v kóde, venujte pozornosť napr. rôznym možnostiam násobenia dvoch vektorov a matíc.
Poznámka pre špekulantov: použitie knižníc ako TensorFlow, Theano, Keras, Lasagne, atď. je zakázané.