Praktický úvod do umělé inteligence

V. Brabcová; Střední průmyslová škola Ostrov; *brabcova.vladenka@gmail.com*
V. Mikula; Arcibiskupské gymnázium v Praze; *vojtech.mikula@email.cz*
E. Valentová; Gymnázium Jiřího Wolkera Prostějov; *eliska@valentovi.com*

Abstrakt

Cílem této práce bylo seznámení se s umělou inteligencí, pochopení její základní struktury a následné zpracování a programování vlastní neuronové sítě schopné rozpoznávat ručně psané číslice. Sledovali jsme trénovací a testovací přesnost, na základě které jsme upravovali parametry neuronové sítě pro dosažení co nejlepších výsledků, a to až 90% přesnost při klasifikaci ručně psaných číslic.

1 Úvod

Umělá inteligence (AI, artificial intelligence) je obor informatiky, který řeší komplexní úlohy za pomoci matematiky a logiky počítačových systémů schopných učení a rozhodování se. Neuronová síť je systém inspirovaný biologickými strukturami schopný řešit problémy. Umělá inteligence se začala vyvíjet přibližně před 80 lety[1], a v posledních letech jsme svědky jejího významného rozmachu, stává se tak běžnou součástí života podstatné části populace. Jen během minulých měsíců došlo k mnoha pokrokům a bylo zveřejněno hned několik nových technologií a modelů.

2 Základy umělé inteligence

Architektur strojového učení existuje více, tato práce se zaměřuje na tvorbu neuronové sítě typu perceptron s jednou skrytou vrstvou.

2.1 Struktura neuronové sítě

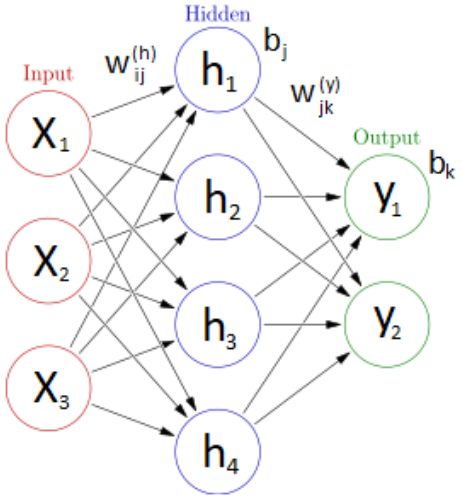
Neuronová síť je hierarchicky strukturovaná. Skládá se z neuronů, které tvoří vrstvy. Tyto vrstvy se dělí na vstupní, skryté a výstupní, přičemž skrytých může být více. Vstupní vrstva (input layer) slouží k přijímání vstupních dat, které jsou předávány dál do neuronů ve skrytých vrstvách. Skryté vrstvy (hidden layers) se nachází mezi vstupní a výstupní vrstvou a transformují vstupní data pomocí vah (důležitosti) a funkcí na výstupní data. Výstupní vrstva (output layer) na základě zpracovaných dat predikuje výsledky. Spojení neuronů v síti se nazývají synapse a mají přidělené váhy, které ovlivňují přenos signálu mezi dalšími neurony. Hodnota vah se mění pomocí algoritmu zpětného šíření chyby (backpropagation) v průběhu učení, čehož se využívá k trénování sítě. Schéma sítě je na obrázku 1.

2.2 Princip – forward pass

Neuronová síť pracuje s číselnými daty. Mezi jednotlivými vrstvami dochází k převádění a modifikaci těchto dat pomocí funkcí.
Jestliže jdou data ze vstupního neuronu x_i do skrytého neuronu h_j s posunem b_j po dráze s váhou w_{ij}, změní se podle vztahu

$$h_j = f\left(\sum_i w_{ij}x_i + b_j\right) ,$$

kde f je nelineární funkce. Pokud bychom použili lineární, dostali bychom na výstupu opět lineární funkci, protože kombinace lineárních funkcí je opět lineární, a nebylo by možné modelovat komplexní závislosti. V našem případě používáme funkci sigmoid, která je definovaná jako



Obrázek 1: Schéma neuronové sítě

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

Obdobně pak mezi skrytým neuronem h_j a výstupním neuronem y_k s posunem b_k po dráze s váhou w_{jk} platí

$$y_k = a\left(\sum_j w_{jk}y_i + b_k\right) ,$$

kde a je takzvaná aktivační funkce, která slouží ke konverzi výstupních dat do požadovaného formátu. Například pokud chceme na výstupu reálná čísla, používá se identita. Pro třídění do kategorií se používá funkce softmax, kterou jsme použili i zde, což je vektorová funkce, jejíž k-tá složka je definovaná jako

$$\text{softmax}(\vec{x})_k = \frac{e^{x_k}}{\sum_j e^{x_j}} .$$

Forward pass je celkový proces výpočtu výstupních dat y ze vstupních dat x. Při správném nastavení vah a posunutí a dostatečném množství skrytých neuronů se dokáže síť podle věty o univerzální aproximaci libovolně přesně přiblížit k jakékoliv funkci, což je zajištěno nelinearitou na skryté vrstvě.

2.3 Trénování – backward pass

K vyhodnocování, jak se výsledek sítě liší od reálného, se používá tzv. „loss“ funkce. V případě naší sítě byla použita střední kvadratická chyba definovaná jako

$$L(y_k, \hat{y}_k) = \frac{1}{N}\sum_{k=1}^{N}(y_k - \hat{y}_k)^2 .$$

Cílem tedy je co nejvíce minimalizovat funkci L, respektive najít takové parametry sítě, aby bylo L co nejmenší. Toho docílíme změnou všech parametrů Θ , tedy vah a posunutí, podle vztahu

$$\vec{\Theta}' = \vec{\Theta} - \alpha \nabla_{\Theta} L \,,$$

kde ∇_{Θ} je vektor parciálních derivací vůči parametrům Θ. Jelikož L závisí na Θ složitým vztahem $L = L(y_k, \hat{y}_k) = L(y_k(x;\Theta), \hat{y}_k) = L(y_k(h_j(x_i; w_{ij}^{(h)}, b_j^{(h)}); w_{jk}^{(y)}, b_k^{(y)}), \hat{y}_k)$, je pro výpočet nutné použít mnohokrát řetízkové pravidlo.

3 Struktura kódu

Naprogramovali jsme neuronovou síť s jednou skrytou vrstvou. V programu jsme použili knihovny numpy, matplotlib.pyplot a sklearn.datasets. Ovšem krom funkce „numpy.einsum“, která vyhodnocuje Einsteinovu sumační konvenci, jsme nepoužili žádné pokročilejší funkce, a tak je naše neuronová síť naprogramovaná od základu – pouze pomocí operací na maticích.

Hlavní částí kódu je třída NeuralNetwork, která má v atributech uloženy své parametry. Mezi její metody patří logicky „forward_pass“ a „backward_pass“, ale přidali jsme i metodu „accuracy“, která umí změřit percentuelní úspěšnost neuronové sítě na daném testovacím datasetu nebo „export“, díky níž můžeme momentální nastavení neuronové sítě uložit pomocí knihovny pickle. Druhou podstatnou složkou je hlavní soubor, ve kterém lze upravovat trénovací hyperparametry jako třeba počet epoch či velikost datasetu.

Ukázka kódu je na obrázku 2. Celý kód lze nalézt na Githubu.[1]

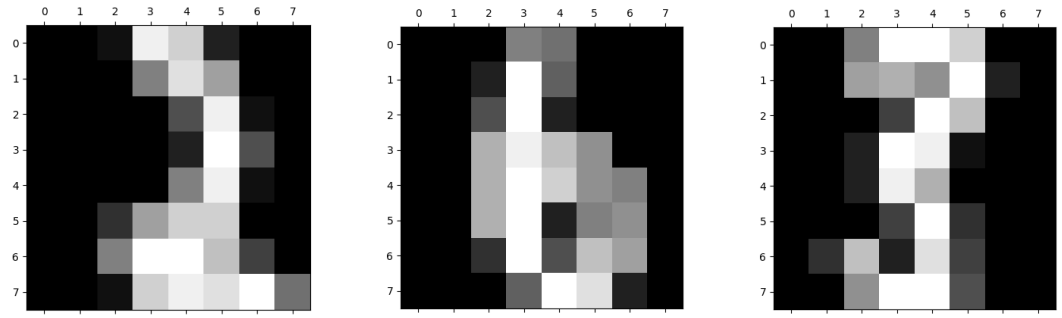
```
def backward_pass(self, x, y_hat) -> None:
    """ Updatuje svoje parametry (weights a biases) na základě trénovacích dat"""
    h = self.compute_h(x)
    y = self.compute_y(h)
    dL_by = (2/K_LAYER)*(y - y_hat)*y*(1-y)
    dL_wy = np.einsum("k,j -> jk", dL_by, h)
    dL_bh = np.einsum("m,jm,j,j -> j", dL_by, self.y_weights, h, 1-h)
    dL_wh = np.einsum("j,i -> ij", dL_bh, x)

    self.y_biases -= ALPHA*dL_by
    self.h_biases -= ALPHA*dL_bh
    self.y_weights -= ALPHA*dL_wy
    self.h_weights -= ALPHA*dL_wh
```

Obrázek 2: Ukázka kódu.

4 Výsledky

Vyzkoušeli jsme několik modelů s různou volbou hyperparametrů (velikost sítě, velikost trénovacího datasetu, počet trénovacích epoch, počet modelů použitých pro predikci). Výsledkem práce je funkční model schopný predikovat s úspěšností až 90 %. Na obrázku 3 jsou ukázky vstupů a identifikovaných výstupů modelu.

[1] https://github.com/ikny/digits-classification



Obrázek 3: Ukázky vstupních dat. Na obrázcích jsou číslice 2, 6 a 3. První dvě model identifikoval správně, poslední identifikoval jako číslici 8.

5 Diskuze

Narazili jsme na několik problémů, se kterými se vývoj umělé inteligence běžně potýká.

Zaprvé, model se může při trénování zaseknout v lokálním minimu funkce L, a tudíž zůstane málo natrénovaný. Tomu se dá předejít vícero způsoby, například opakovaným trénováním nových modelů s různě inicializovanými parametry. Při vyzkoušení tohoto postupu jsme zjistili, že i mezi modely s totožnými hyperparametry mohou vzniknout razantní rozdíly v úspěšnosti.

Zadruhé, pokud je pro model jednodušší zapamatovat si trénovací data než souvislosti mezi nimi (například proto, že trénovací dataset je moc malý), dochází k "overfitování", kdy má model velkou přesnost na trénovacích datech, ale malou na testovacích (při trénování neznámých) datech. Abychom tomu předešli, experimentovali jsme s velikostí datasetu a měřili jsme úspěšnost v celém průběhu trénování.

6 Shrnutí

V rámci projektu jsme porozuměli základům fungování neuronových sítí a jejich trénování. Setkali jsme se s pro nás novými matematickými koncepty, díky nimž jsme dokázali od základu naprogramovat svou vlastní neuronovou síť na klasifikaci a poté ji použili na rozpoznání ručně psaných číslic. V rámci projektu jsme se setkali s problémy, které jsou při trénování neuronové sítě běžné, a pomocí experimentace s nastavením hyperparametrů jsme je řešili.

Poděkování

Děkujeme Martinu Vaňkovi za pomoc a vedení projektu. Dále děkujeme MFF UK za poskytnutí prostor pro realizaci projektu.

Reference

[1] Historie neuronových počítačů a sítí. Cit. 20. 6. 2023. Online. Dostupné z: https://www.fi.muni.cz/usr/jkucera/pv109/2000/xneudert.html

[2] Materiály k předmětu Úvod do strojového učení v Pythonu. Cit. 20. 6. 2023. Online. Dostupné z: https://ufal.mff.cuni.cz/courses/npfl129/2223-winter

[3] Schéma neuronové sítě. Cit. 20. 6. 2023. Online. Dostupné z: https://en.wikipedia.org/wiki/Artificial_neural_network