SFC - Projekt 2022/2023

Praktická úloha řešená sítí BP - predikce

Autor: Bc. Jan Lorenc (xloren15)
Datum: 12. listopadu 2022

Obsah

1 Zadání

Cílem projektu je implementovat neuronovou síť učící se pomocí algoritmu backpropagation a zvolit si praktickou prediktivní úlohu, kterou bude síť řešit. Vzhledem k volnému zadání jsem se rozhodl projekt pojmout tak, že naimplementuji obecnou knihovnu pro vytváření neuronových sítí. Praktické úlohy byly zvoleny celkem tři. Pro jednoduché otestování funkčnosti provádí jedna úloha lineární regresi nad zašumělými daty a dále predikuje vývoj funkce. V dalším příkladu je aproximována zašumělá sinusoida. Hlavním úkolem pak byla zvolena predikce ceny domů v Bostonu dle existujícího datasetu[1].

2 Knihovna pro neuronové sítě

V adrésáři *src/* odevzdaného řešení se nachází 4 další podsložky. Tyto obsahují třídy pro vytváření neuronových sítí. Kód je úhledný a poctivě komentovaný, v následujících sekcí tedy popíší jen jejich obecnou implementaci.

ActivationFunctions

Tento modul obsahuje implementace tří aktivačních funkcí, které jsou dále použity v prediktivních úlohách. Těmito funkcemi jsou lineární funkce, ReLU a sigmoida. Každá z těchto funkcí je reprezentována třídou se stejným rozhraním implementujícím metody `fn` a `grad`. Toto umožňuje jejich snadnou záměnu ve vrstvách sítě bez nutnosti změny kódu. Metoda `fn` vypočítá hodnotu dané funkce a je používána při dopředném průchodu. Metoda `grad` počítá derivaci potřebnou pro zpětný průchod.

LossFunctions

Uvnitř tohoto modulu lze najít chybové funkce Mean Squared Error (MSE) a Cross Entropy. Funkce jsou opět reprezentovány třídami se stejným rozhraním jako aktivační funkce. Pro predikci byla ve všech úlohách použita pouze MSE, implementace Cross Entropy tedy nebyla testována. Vzhledem k její popularitě jsem se ji však rozhodl implementovat pro úplnost knihovny.

Optimizers

Pro zvýšení efektivity algoritmu a pro efektivní porovnání s referenčním řešením cílové úlohy jsem se rozhodl implementovat i optimizátor Adam. Implementace se drží vzorce pro Adam, takže zde není nic zvláštního, nicméně za zmíňku stojí jeho potřeba pro znalost aktuální iterace pro umocnění β_1 a β_2 ve fázi korekce. Implementačně je proto potřeba index aktuální iterace přenášet celou zpětnou fázi BP algoritmu.

NeuralNetwork

V tomto modulu se nachází jádro implementace neuronové sítě pro BP. Celkem obsahuje 2 třídy a to `NN` a `Layer`. Samotná `NN` třída prakticky slouží jen jako kontejner pro vrstvy. Obsahuje metodu pro přidání vrstev a její metody `forward` a `backward` pouze provolávají stejnojmené metody daných vrstev ve správném pořadí. V dopředné fázi tak posouvají výsledky daných vrstev dál a ve zpětné propagaci přenáší chybu do předchozí vrstvy.

Samotné výpočty probíhají v `Layer` třídě. Její váhy jsou inicializovány normálním rozložením dle parametrů a biasy na 0. Metoda `forward` přijímá vstupní vektor a zapamatuje si jej pro zpětnou propagaci. Dále vypočítá vnitřní potenciál neuronů lineární bázovou funkcí $\vec{u} = \vec{w}\vec{x} + \vec{b}$ a taktéž si ho uloží pro zpětný průchod. Metoda poté vrací výsledek aktivační funkce pro vnitřní potenciál u. Metoda `backward` přijímá gradient předchozí vrstvy (následující v hierarchii, předchozí z pohledu BP) a vypočítá gradient pro své váhy a biasy za použití Adam optimizátoru. V případě poslední vrstvy přijímá chybu sítě. Na závěr metoda upraví váhy a biasy dané vrstvy na základě vypočítaných gradientů.

[1] https://machinelearningmastery.com/regression-tutorial-keras-deep-learning-library-python/

3 Praktické úlohy

V této kapitole jsou popsány provedené úlohy. Pro otestování funkčnosti sítě je řešena úloha lineární regrese. Jelikož neuronové sítě jsou schopny se naučit libovolnou funkci, další úlohou je aproximace sinusoidy. Na závěr je provedena reálná regresní úloha a porovnána s referenčním řešením.

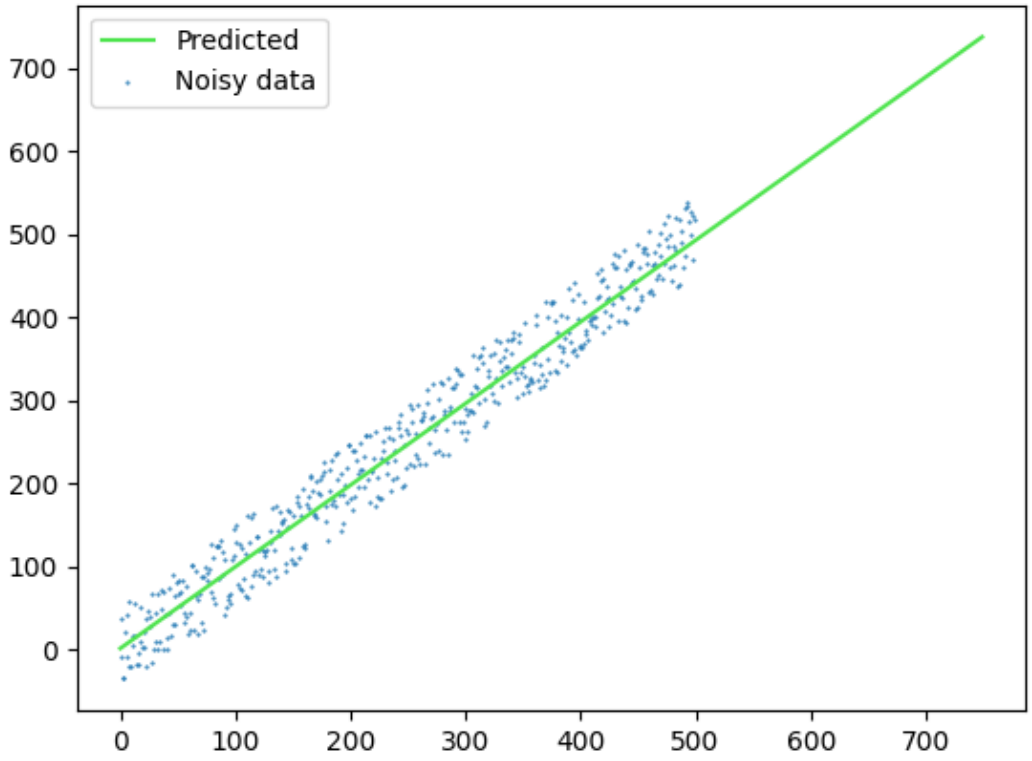
Pro trénování sítě byla vytvořena třída `Trainer`. Ta trénuje přístupem mini-batch. Metoda `train` umožnuje nastavit počet epoch a velikost dávky. Síť je trénována na trénovacích datech a rovnou validována na testovacích. Natrénovaný model je možné uložit a pro každou úlohu je uložen v adresáři *trainedModels/*.

Ve třídě `DataLoader` je prováděna veškerá práce s daty. Pro lineární regresi a aproximaci sinusoidy jsou vygenerovány zašumělé funkce pro trénování a čisté funkce pro validaci. Pro úlohu predikce ceny domů v Bostonu, pro kterou je dataset uložen v *data/dataset.csv*, provádí rozdělení na trénovací/testovací data a na vstupní příznaky a výstup.

Třída `Runner` slouží jako spouštěč úloh. Pro každou úlohu načte data, nadefinuje architekturu sítě, provede výpočet a uloží/zobrazí výsledky.

3.1 Lineární regrese

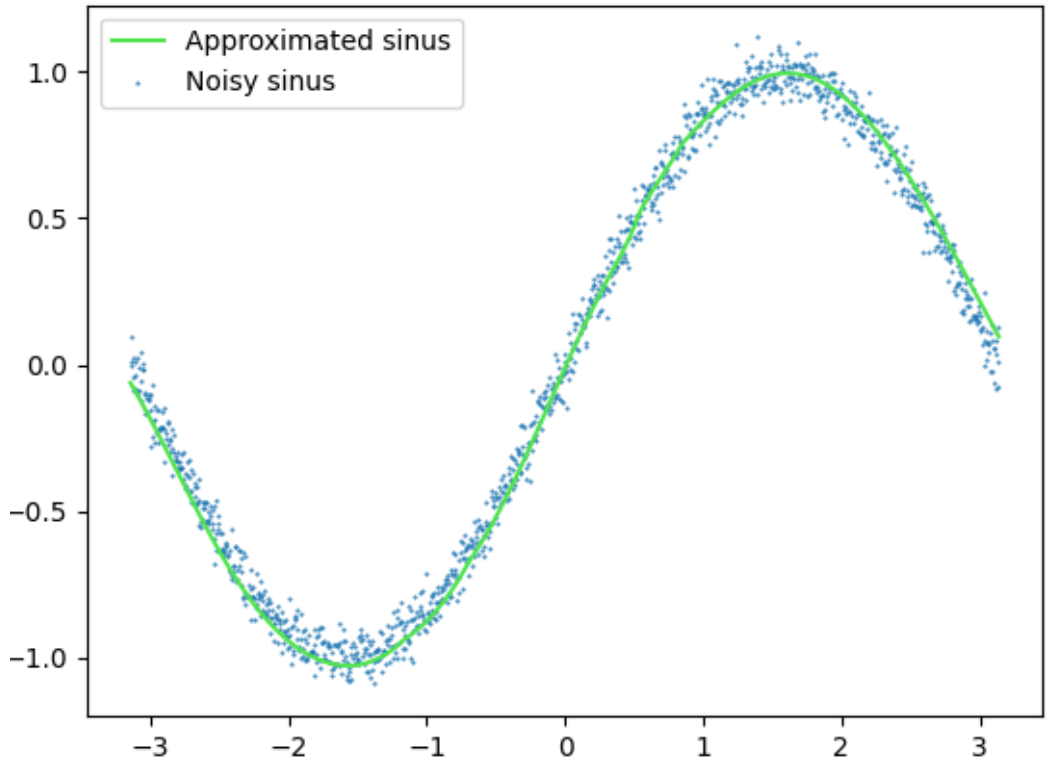
Pro tuto úlohu bylo vygenerováno 500 vzorků po jednoduché přímce $y = x$, ke kterým byl přidán šum z rovnoměrného rozdělení $< -50, 50 >$. Testovací data obsahovala stejné hodnoty osy x k aproximaci a dalších 250 vzorků k predikci. K trénování byla použita síť o dvou vrstvách s MSE chybovou funkcí, kde první vrstva obsahovala 64 neuronů s ReLU aktivací a poslední vrstva 1 neuron bez aktivace.



Obrázek 1: Výsledek trénování po 64 epochách s velikostí batch 32.

3.2 Aproximace sinusoidy

Pro řešení této úlohy byly použity vzorky sinusoidy o aplitudě 1 z intervalu $< -\pi, \pi >$ s krokem 0.005. Tato jedna perioda sinusoidy byla dále Gaussovsky zašumělá. Pro aproximaci se použila třívrstvá síť s MSE chybovou funkcí, kde první vrstva obsahovala 16 neuronů s ReLU aktivací, druhá taktéž 16 neuronů se sigmoid aktivací a výstupní vrstvou je jediný neuron bez aktivace.



Obrázek 2: Výsledek trénování po 256 epochách s velikostí batch 8.

3.3 Predikce ceny domů

Námět k této úloze včetně datasetu jsem převzal z existujícího online řešení[2], se kterým jsem i porovnával výsledky. Dataset je poměrně malý, pouze 506 záznamů, nicméně pro účely projektu, kde jde o implementaci BP algoritmu, zcela postačuje. Každý záznam obsahuje 14 číselných příznaků, z čehož poslední je cena v tísících dolarů. Celkem tedy pracuji se 13 příznaky a predikuji výslednou cenu. Autor uvedeného internetového článku uvádí, že dobrý výsledek pro MSE chybu je kolem 20 (tisíc dolarů). On sám s různými topologiemi sítí dosahuje výsledků v rozmezí 20-30 při implementaci knihovnou Keras.

K otestování vlastního řešení jsem použil stejnou topologii a způsob testování, jako je ve článku. Metoda `houses_k_fold` třídy `Runner` spouští trénování k-krát (10x ve výchozím nastavení) s různým rozdělením dat na trénovací a testovací. Na standardní výstup pak vypíše průměr chyb ze všech běhů. Výsledky se v obecnosti pohybují v rozmezí 15-30 MSE, v závislosti na náhodném rozdělení dat s občasnými výkyvy způsobenými náhodností mini-batch trénováním. Lze z toho však vyhodnotit, že výsledky odpovídají očekávání. Topologie sítě se v této úloze skládá stejně jako ve článku ze 2 vrstev, první s 13 neurony a ReLU aktivací, druhou s 1 neuronem bez aktivace.

Aby existoval i natrénovaný model pro odevzdání, třída `Runner` nabízí ještě metodu `houses`, která spustí pouze 1 trénování a uloží nejen model, ale i trénovací a testovací data pro referenci. Tato data lze nalézt v adresáři *data/*. Pro odevzdané rozdělení datasetu je model natrénovaný s chybou 25.323, což opět odpovídá referenční implementaci.

[2] https://machinelearningmastery.com/regression-tutorial-keras-deep-learning-library-python/

4 Spuštění aplikace

Aplikace je implementována v jazyce Python 3, takže je potřeba mít zprovozněný interpret. Dále již aplikace používá pouze knihovnu numpy pro výpočty a matplotlib k vytvoření grafů lineární regrese a aproximace sinusoidy. Tyto jsou taktéž uvedeny v souboru *src/requirements.txt*. Moduly počítají s kořenovým adresářem, který je odevzdaný, nikoliv složkou *src/*. Aplikaci je možné spustit následovně:

```
python3 src/main.py [-t houses | houses-kfold | linear | sinus]
```

Výběr úlohy se provádí přepínačem `-t` či `--task`. Ve výchozím nastavení se spustí úloha *houses*, tedy jedno spuštění predikce cen domů. Aplikace použije již natrénované modely ze složky *trainedModels*. Pokud je žádoucí spustit nové trénování, je třeba pozměnit volání v souboru *src/main.py*.

Závěr

Byla implementována obecná knihovna pro dopředné neuronové sítě využívající algoritmus zpětné propagace. Za účelem dosažení lepších výsledků a možnost porovnání s jinými řešeními je základní BP algoritmus obohacen o Adam optimizátor. Byť je cílem projektu predikce, knihovna umožňuje vytvářet sítě i pro jiné úlohy. Byly provedeny celkem 3 regresní úlohy s využitím vytvořené knihovny a všechny proběhly úspěšně.