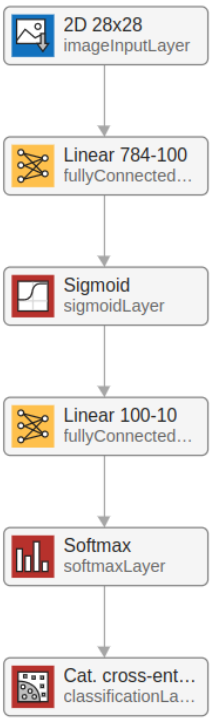
Předmět: Soft Computing
Název projektu: Demonstrace učení BP - základní algoritmus + vybraný optimalizátor
Ak. rok: 2022/2023
Autor: Richard Klem
E-mail: xklemr00@stud.fit.vutbr.cz
Datum: 27. listopadu 2022

1 Zadání a řešené problémy

Zadání definuje úkol jako demonstraci algoritmu backpropagation pro učení neuronových sítí s vybraným optimalizátorem. K základnímu algoritmu backpropagation jsem implementoval optimalizátor Adam jako standartní součást zadání a optimalizátor AmsGrad jako rozšíření. Dále jsem vytvořil grafickou dekstopovou aplikaci, která se zaměřuje na zobrazování průběhu učení neuronové sítě z pohledu ztrátové funkce. Stejně tak i informativní výpisy v příkazové řádce popisují tuto statistiku učení. Jako demonstrační problém slouží problém klasifikace číslic z datasetu MNIST do 10 tříd. Inspirací, jak problematiku uchopit mi byl veřejný repositář pana Sylvaina Guggera[1].

1.1 Řešené problémy/dílčí úkoly

Implementoval jsem neuronovou síť, jejích architekturu můžete vidět na obrázku1. Jedná se o dvě plně propojené lineární vrstvy, kde první využívá aktivační funkci sigmoidy, zatímco na výstup z druhé vrstvy je aplikovaná funkce softmax. Výstup ze softmax funkce je vstupem pro výpočet objektivní funkce více třídní křížové entropie.



Obrázek 1: Architektura implementované sítě

Níže jsou popsány rovnice podle kterých se programově počítají jednotlivé funkce/vrstvy a to dopředným směrem a zpětným směrem. Vzorce jsou uvedeny pro jeden vzorek dat, ale implementačně se pak pracuje v

[1] https://github.com/sgugger/Deep-Learning

prostoru o jednu dimenzi vyšším, protože jsem implementoval regularizační metodu mini-batch.

Sigmoidální funkce je definovaná jako: $y_j = \frac{e^x}{1 + e^x}$

Derivace pak: $\frac{\partial y}{\partial x} = y(1 - y)$

Softmax funkce je definovaná jako: $y_j = \frac{e^{x_k}}{\sum_{k=1}^{n} e^{x_k}}$

Její derivace pak: $\frac{\partial y_j}{\partial x_j} = \begin{cases} y_j(1 - y_j) & pro\ j = i \\ -y_j y_i & pro\ j \neq i \end{cases}$

Což lze ekvivalentně zapsat jako: $y_j \left(g_j - \sum_{k=1}^{n} y_k g_k \right)$

Více třídní křížová entropie: $loss = \begin{cases} -log(x) & pro\ y = 1 \\ 0 & jinak \end{cases}$

Její derivace: $\frac{\partial y}{\partial x} = \frac{-y}{x}$

Lineární vrstva jednoduše maticově zapsána: $Y = X \times W + B$, kde X jsou vstupy, W váhy a B předpětí.

Rozloženě můžeme psát: $y_i = \sum_{k-1}^{n_{in}} x_k w_{k,i} + b_i$

Derivace podle předpětí pak vychází jako: $\nabla_b = \frac{\partial loss}{\partial b_i} = \frac{\partial loss}{\partial y_i} \times \frac{\partial y_i}{\partial b_i} = \frac{\partial loss}{\partial y_i}$

A derivace podle vah: $\nabla_w = \frac{\partial loss}{\partial w_i} = \frac{\partial loss}{\partial y_i} \times \frac{\partial y_i}{\partial w_i} = x \frac{\partial loss}{\partial y_i}$

A zejména vypočítáme derivace vzhledem ke vstupu: x: $\frac{\partial loss}{\partial x} = \sum_{i=1}^{n_{out}} \frac{\partial loss}{\partial y_i} \times \frac{\partial y_i}{\partial x_k} = \sum_{i=1}^{n_{out}} \frac{\partial loss}{\partial y_i} w_i$

Což můžeme zapsat maticově jako: $\nabla_{new} = \nabla_{old} \times W^t$

2 Desktopová aplikace

Aplikace je napsaná v jazyce Python s využitím knihoven:

- `pickle` pro efektivní správu dat,
- `numpy` pro výpočty neuronové sítě,
- `PyQt5` pro vytvoření uživatelského rozhraní v podobě desktopové aplikace a

- `matplotlib` pro tvorbu grafů loss funkce.

Aplikace se nepřekládá, jedná se o sadu pomocných skriptů a hlavní skript `mainwindow.py`, který spouští celou aplikaci. Je připraven pomocný shell skript, který zajistí veškeré potřebné úkony ke spuštění aplikace z čistého prostředí. Více v podkapitole2.1.

Pokud spouštíte aplikaci vzdáleně pomocí nástroje `ssh`, nezapomeňte na přepínač `-X`, aby se okno aplikace mohlo otevřít. Součástí výstupu je i formátovaný výpis do příkazové řádky, který lze použít pro případné další semi-strojové zpracování.

```
None    - Train Loss    =   2.4396657622184748
Amsgrad - Train Loss    =   0.7553334199577334
None    - Train Loss    =   2.2235683718226773
Amsgrad - Train Loss    =   0.39920740352974193
None    - Train Loss    =   2.128961182887082
Amsgrad - Train Loss    =   0.3222333506119762
None    - Evaluation Loss = 2.0830793856488827
Amsgrad - Evaluation Loss = 0.31157846043645643
```

V prvotním nastavení se zobrazuje základní algoritmu backpropagation a AmsGrad optimizátor. Lze si zvolit libovolnou dvouprvkovou permutaci z množiny {`None, Adam, AmsGrad`}. Dále lze přepínat mezi zobrazením do jednoho grafu anebo odděleně. K dispozici je i funkce výpočtu a zobrazení loss hodnoty pro aktuální natrénované modely z evaluačního/testovacího běhu. K další funkci programu patří změna alfa parametru. Doporučuji se držet mezi hodnotami 0.01 a 0.00001 pro stabilitu systému.

Po zadání požadovaných parametrů (nebo ponechání prvotních hodnot) lze trénování krokovat po jedné epoše/iteraci/kroku anebo spustit dávkové trénování o požadovaném počtu kroků.

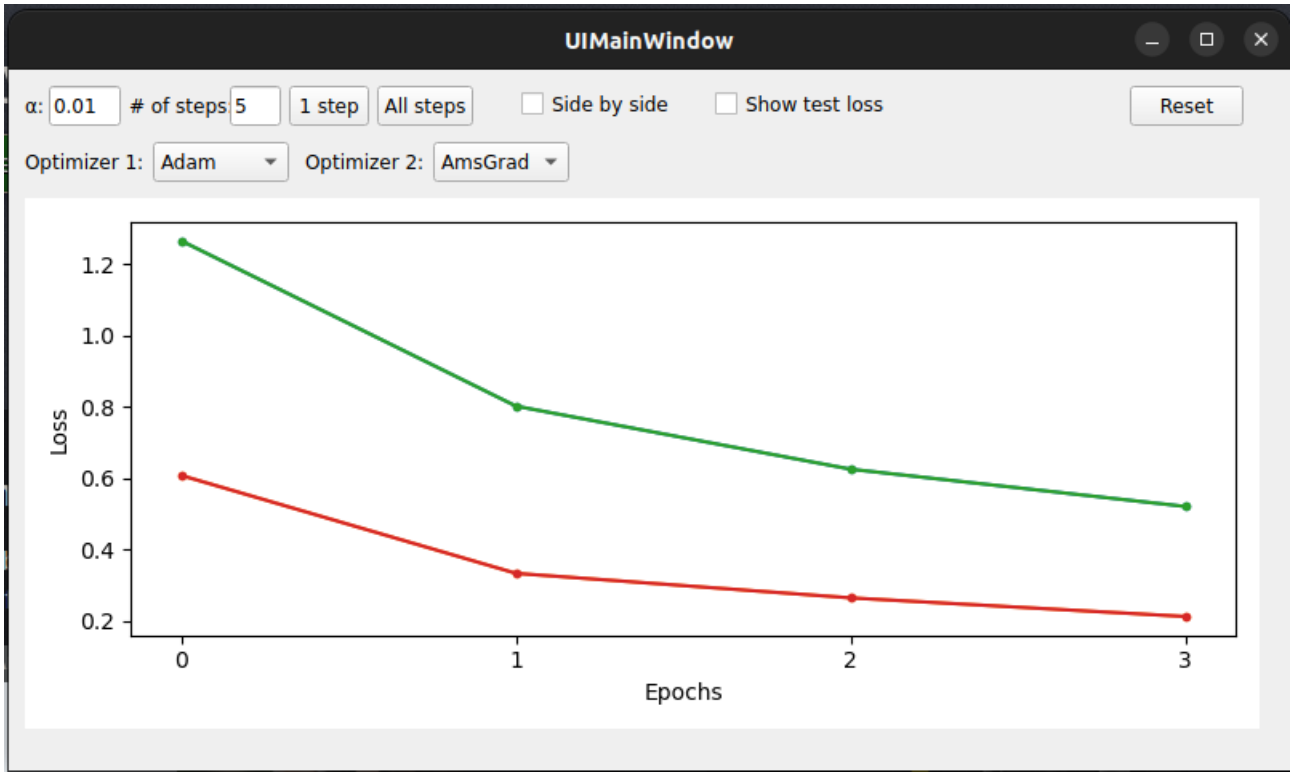
Vykreslování grafů se provádí po každém kroku v případě krokování anebo na konci dávkového běhu najednou. Z tohoto důvodu si prosím přečtěte upozornění v podsekci 2.2.

2.1 Spuštění

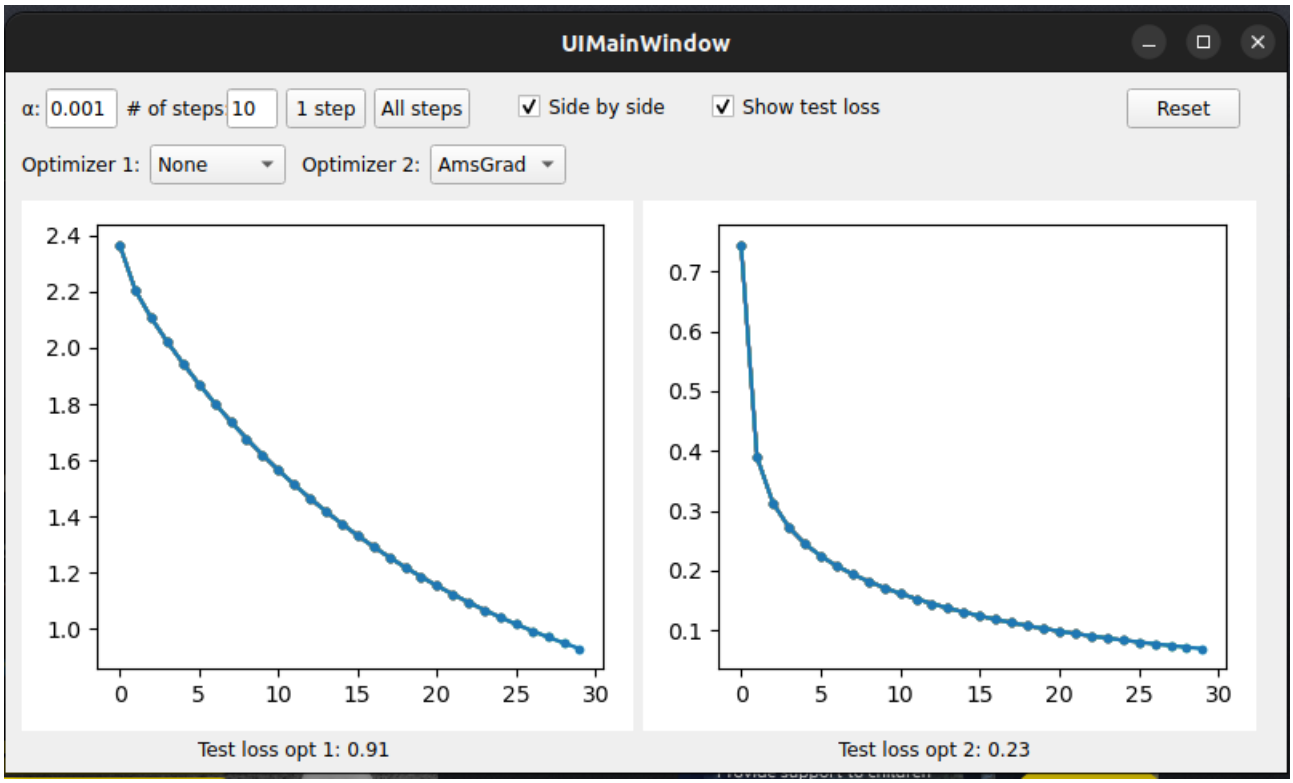
Spusťte skript `run.sh` v hlavním afresáři. Skript rozbalí archiv s redukovaným datasetem MNIST, vytvoří virtuální prostředí, nainstaluje potřebné balíčky a spustí aplikaci s počátečním nastavením.

Pokud by se spuštění skriptu nepodařilo, zkuste jednotlivé řádky ze skriptu po jednom spustit přímo manuálně v příkazové řádce.

Níže jsou uvedeny dva snímky (2 a 3) aplikace sloužící k demonstračním účelům, jaké jsou výstupy aplikace. Také ukazují možné nastavení parametrů aplikace a zejména pak trénovaných modelů.



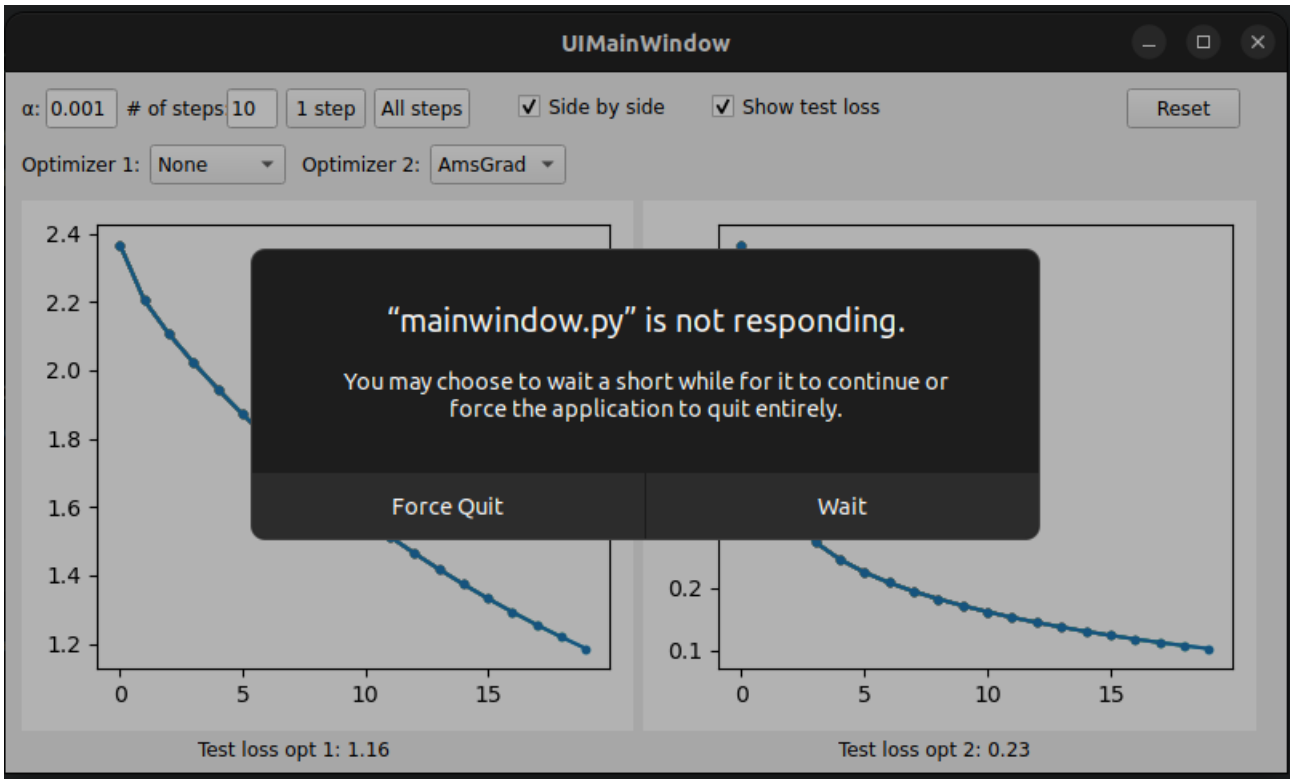
Obrázek 2: Zobrazení obou optimalizátorů v jednom grafu.



Obrázek 3: Zobrazení každého optimalizátoru zvlášť a hodnoty loss funkce pro evaluační běh.

2.2 Omezení programu

Může se stát (na merlinovi pravděpodobné), že pro více epochální běhy - tedy při využití tlačítka *All steps* dojde k zamrznutí okna, protože se výpočet a vykreslení provádí v jednom vlákně. Není třeba nijaké akce, stačí chvíli počkat a modální okno zavřít stiskem tlačítka *Wait*. V případě, že by se okno objevilo znovu, postup opakujte. Exemplární příklad lze vidět na obrázku 4.



Obrázek 4: Modální okno informující o dlouho běžícím programu.