Neurónové siete

Nikolas Knapik
10/12/2024

Contents

MNIST klasifikátor

1. Úvod

Táto úloha sa zameriava na klasifikáciu ručne písaných číslic z dátového súboru MNIST pomocou doprednej neurónovej siete **MLP** (viacvrstvový perceptrón). Model bol otestovaný a trénový toromi rôznymi optimalizačnými algoritmami: SGD, SGD s momentom a Adam. Výsledná presnosť modelu dosiahla približne 97% na testovacej množine.

2. Architektúra modelu

Model pozostáva z nasledujúcich vrstiev:

- **Vstupná vrstva:** 784 neurónov (každý obrázok MNIST má rozmer 28x28, zploštený na 1D vektor.
- **Skrytá vrstva 1:** 256 neurónov s aktivačnou funkciou ReLU.
- **Skrytá vrstva 2:** 128 neurónov s aktivačnou funkciou ReLU.
- **Výstupná vrstva:** 10 neurónov. Jeden pre každú číslicu od 0 do 9, kde výstupné hodnoty sú vo forme logits. Logits sú surové výstupy poslednej vrstvy neurónovej siete. Tie sa neskôr preložia do pravdepodobností a vypočíta sa strata pomocou funkcie $CrossEntropyLoss(\quad)$, ktorá interne aplikuje softmax.

Parametre použité na algoritmus:

Parameter	Hodnota
Batch size	64
Počet epoch	10
Learning rate	0.01 (SGD, SGD s momentom), 0.001 (Adam)
Momentum	0.9 (SGD s momentom)

Normalizácia dát:

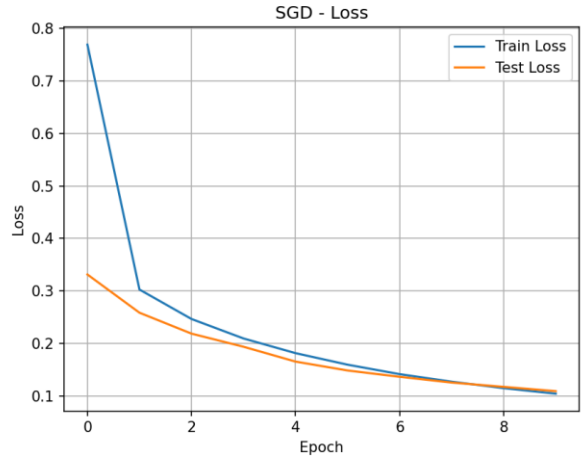
Dáta z datasetu MNIST sú normalizované na stred 0.1307 a štandardnú odchýlku 0.3081. Tieto hodnoty by mali zabezpečiť väčšiu stabilitu tréningu. Tieto hodnoty boli vypočítané na základe množstva obrázkov (60 000) a veľkosti jednotlivých obrázkov (28x28)

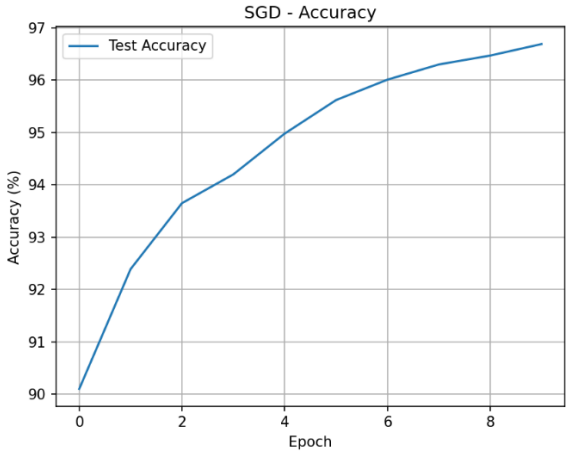
3. Výsledky tréningu

Tieto dáta boli zaznamenané na poslednej epoche.

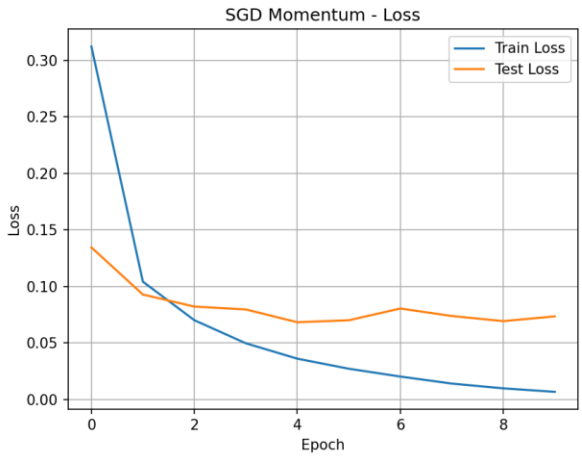
Algoritmus	Trénovacia strata	Testovacia strata	Testovacia presnosť
SGD	0.1046	0.1094	96.69 %
SGD s momentom	0.0067	0.0738	98.03 %
Adam	0.0194	0.0934	97.73 %

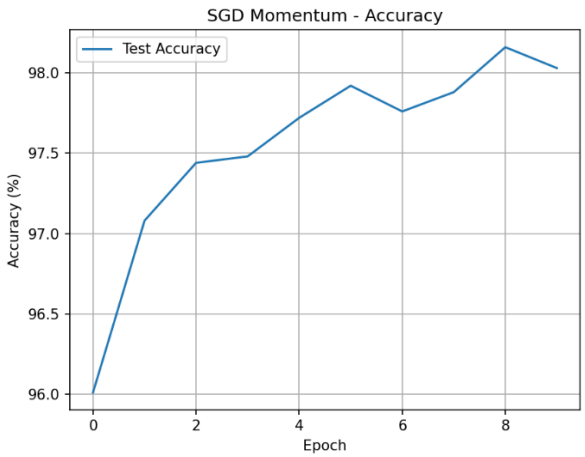
Graf priebehu SGD modela:



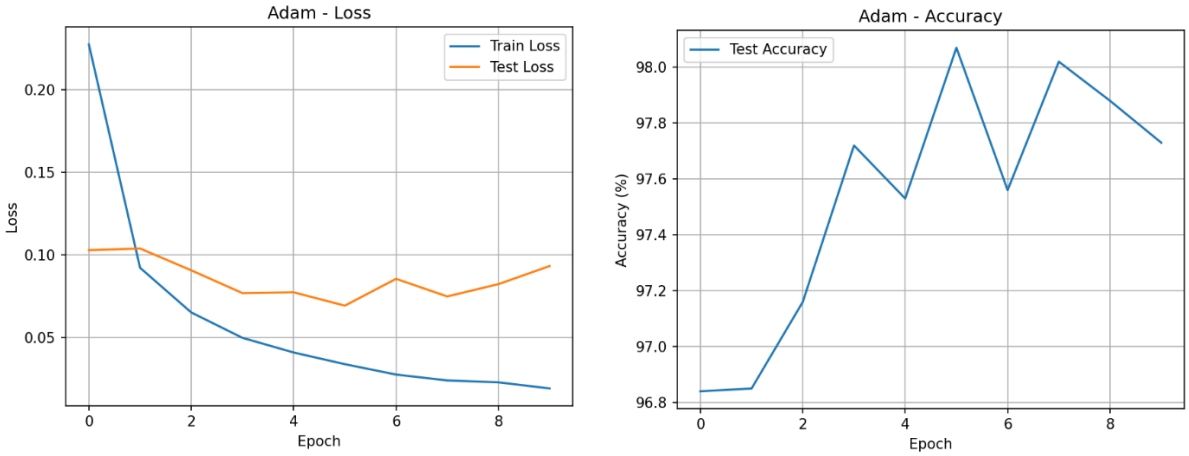


Graf priebehu SGD s momentom modela:



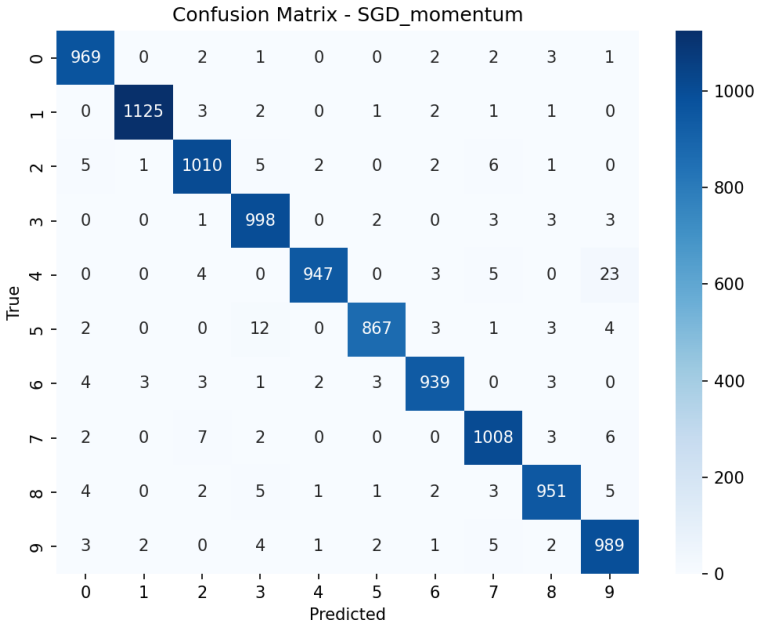


Graf priebehu Adam modela:



4. Confusion Matrix

Najlepší model, trénovaný s optimalizačným algoritmom SGD s momentom. Ktorý dosiahol najvyššiu presnosť 98.03 %.



5. Porovnanie modelov

SGD (Stochastic Gradient Descent):

Algoritmus aktualizuje váhy modelu na základe gradientu chyby vypočítaného pre jeden batch dát. Je jednoduchý a rýchly algoritmus vhodný na problémy, kde je chybová plocha jednoduchá a dobre štruktúrovaná. Avšak aktualizácia gradientov môže oscilovať, čo môže spôsobiť spomalenie tréningu. Taktiež je veľmi citlivý na hodnotu learning rate kedže pri príliš vysokej hodnote môže model diverzovať. Ak je príliš nízka, tréning bude pomalý. V mojom kóde mal výslednú presnosť približne 96.7 %.

SGD s momentom:

Vylepšenie SGD modela s pridaním mementu, ktorý zohľadňuje minulé gradienty. To pomáha urýchliž konvergencia a znižuje oscilácie. V porovnaní s klasickým SGD má rýchlejšiu kovergenciu čo je užitočné pri menších chybových plochách. Požaduje dodatočný parameter na ladenie. V mojích výsledkoch produkoval presnosť približne 97.4 %

Adam:

Kombinuje vlastnosti SGD s momentom a adaptívneho učenia, pričom aktualizácie váh sú upravované na základe druhého momentu gradientov. Adam model prispôsobuje learning rate pre každú váhu, čo zlepšuje tréning. Vhodný aj na zložitejšie problémy. Použitie Adam modela môže viesť k pretrénovaniu, ak sa nepoužije správna regularizácia. Uchováva predchádzajúce gradienty kvôli čomu vyžaduje viac pamäte.

6. Záver

Táto dokumentácia zhŕňa implementáciu viacvrstvového perceptrónu s dvoma skrytými vrstvami. Najlepší optimalizačný algoritmus sa preukázal byť SGD s momentom kedže dosiahol až 98 percentnú testovaciu úspešnosť.

Backpropagation algoritmus

1. Úvod

V tejto úlohe som implementoval algoritmus backpropagation pre dvojvrstvovú doprednú neurónovú sieť (MLP) pomocou knižnice NuumPy. Cieľom bolo naučiť sieť riešiť problémy XOR, AND, a OR pomomcou minimalizácie chyby pomocou chybovej funkcie (MSE).

2. Architektúra siete

- **Vstupná vrstva:** 2 neuróny (pre dva binárne vstupy).
- **Skrytá vrstva:** 4 neuróny s aktivačnou funkciou tanh.
- **Výstupná vrstva:** 1 neurón s aktivačnou funkciou sigmoid.
- **Chybová funkcia:** MSE (Mean Squarred Error).
- **Optimalizačný algoritmus:** SGD bez momenta.
- **Rýchlosť učenia:** $lr = 1.0$.
- **Počet epoch**: 500.

3. Implementácia algoritmu

Dopredný smer (Forward Pass):

V skrytej vrstve sa počíta výstup (z1) a aktiacia (a1) pomocou vzorcov kde X je počiatočná množina. W je váha danej vrstvy a b je bias:

$$z1 = X \cdot W1 + b1$$

$$a1 = tanh(z1)$$

Vo výstupnej vrstve sa taktiež počíta výstup (z2) a aktivácia (a2) pomocou vzorcov:

$$z2 = a1 \cdot W2 + b2$$

$$a2 = sigmoid(z2)$$

Spätný smer (Bacward Pass):

Spätným chodom sa počíta gradient čož je derivácia chybovej funkcie (loss function) vzhľadom na váhy modelu. Gradietn určuje, ktorým smerom a o koľko je potrebné upraviť váhy, aby sa minimalizovala chyba.

Gradient na výstupe (dz2):

$$dz2 = \nabla MSE \cdot sigmoid'(a2)$$

Gradient pre váhy a bias vo výstupnej vrstve (dW2, db2):

$$dW2 = a1T \cdot dz2, \qquad db2 = \sum(dz2)$$

Gradient skrytej vrstvy (dz1):

$$dz1 = dz2 \cdot W2T \cdot tanh'(a1)$$

Gradient pre váhy a bias v skrytej vrstve (dW1, db1):

$$dW1 = XT \cdot dz1, db1 = \sum(dz1)$$

Použité aktivačné funkcie:

V programe som testoval s hlavnými problémamy funkciu tanh na skryté vrstvy a funkciu sigmoid pre výstupnú vrstvu. Následne som otestoval program s funkciou ReLU, ktorá mi dala pre jednu vrstvu nepresnejšie výsledky ako tanh.

Aktualizácia parametrov:

Parametre váh a biasov sa aktualizujú pomocou gradientov. Pre každú vrstvu sa vypočíta gradient a na základe týchto gradientov sa upravuje každá váha bias.

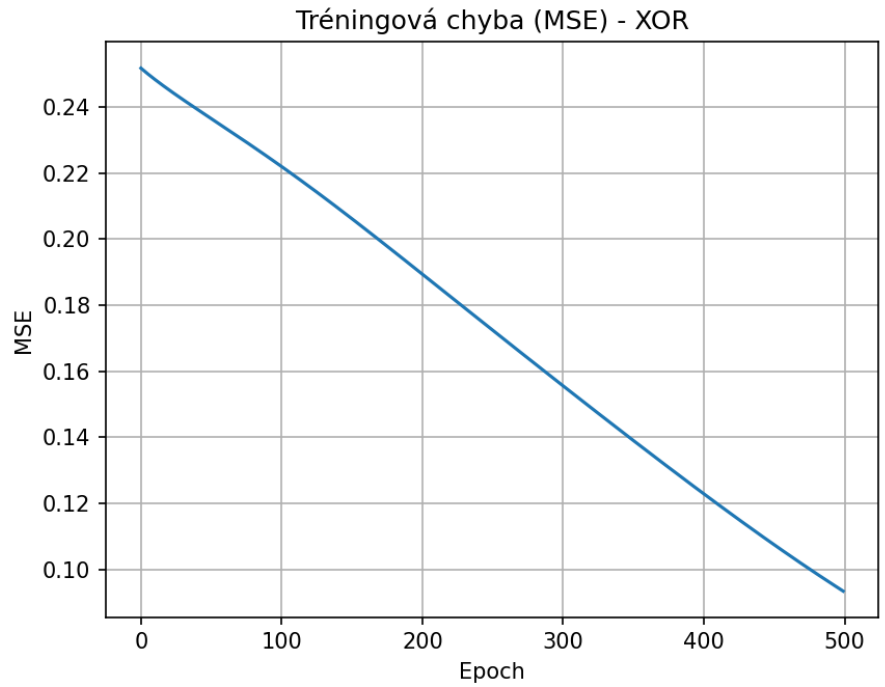
4. Výsledky

Parametre tréningu budú určené pre všekty problémy rovnaké:

- Rýchlosť učenia (lr): 0.1
- Počet epoch: 500

Problém XOR:

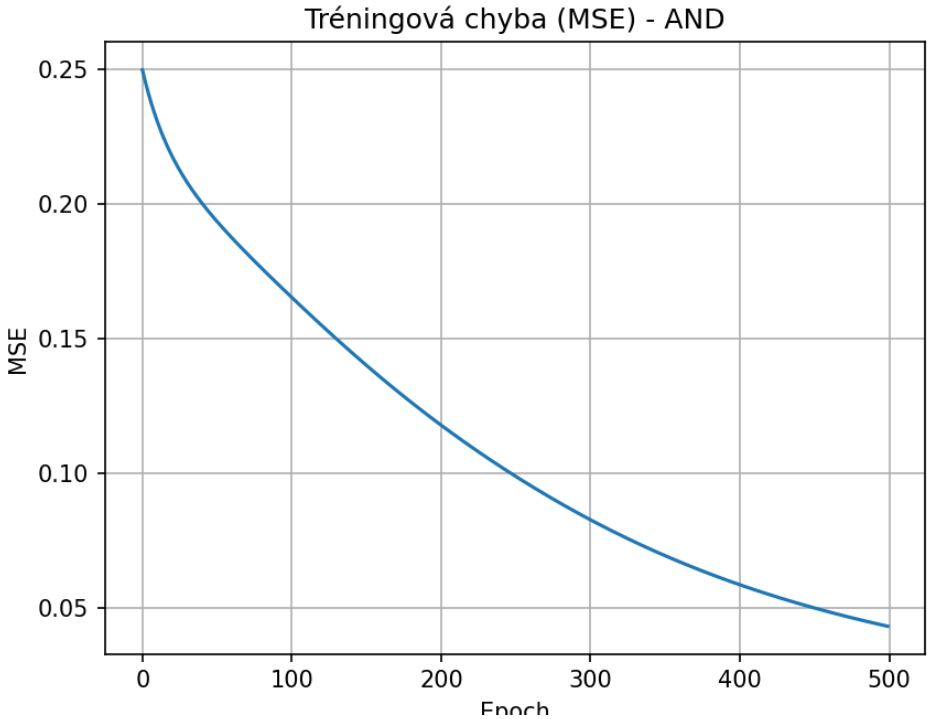
Vstupy (*X*)	Skutočné (Y)	Predikácie	Zaokrúhlenie
[0,0]	0	~0.195	0
[0,1]	1	~0.725	1
[1,0]	1	~0.677	1
[1,1]	0	~0.393	0



Počas tréningu sa hodnota chyby (MSE) postupne znižovala. Predikcia seite na problém XOR je správna. Spočiatku mi problém XOR nechcel dať správny výsledok. Predikcia zrejme uviazla v minime a na konci bola každa predikcia približne 0.50 čiže sa nevedel rozhodnúť či to je jedna alebo nula. Toto som opravil správnym inicializovaním váh pomocou **He initailization**, ktorá mi nakoniec chybu opravila.

Problém AND:

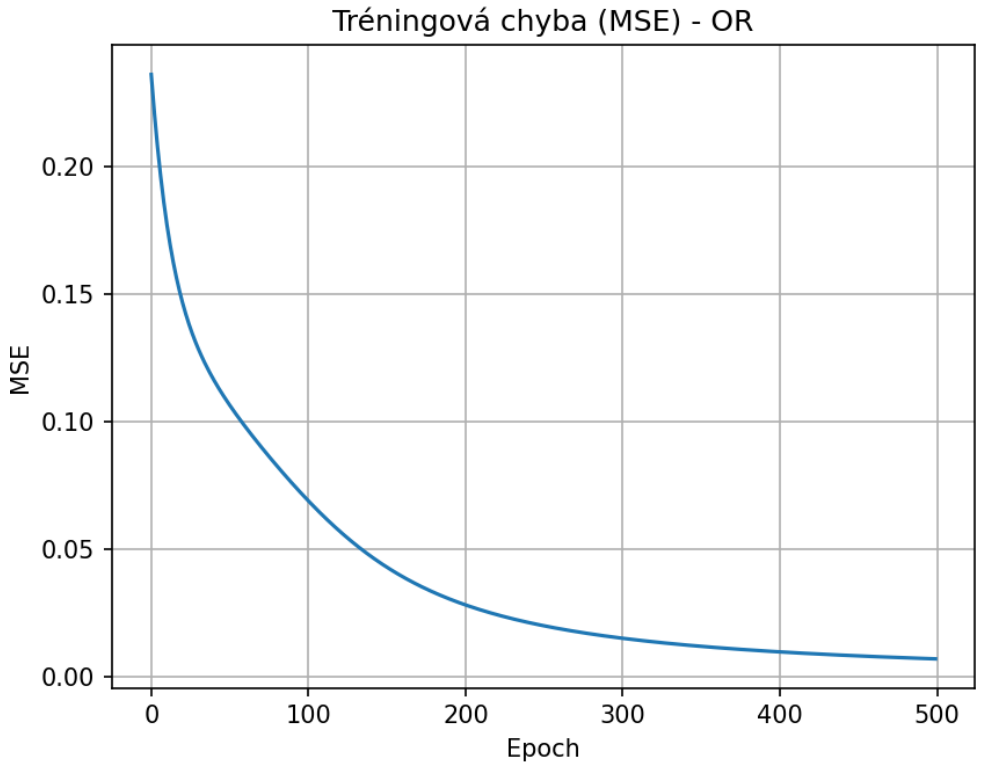
Vstupy (*X*)	Skutočné (Y)	Predikácie	Zaokrúhlenie
[0,0]	0	~0.046	0
[0,1]	0	~0.189	0
[1,0]	0	~0.237	0
[1,1]	1	~0.722	1



Trénovanie problému AND funguje správne. Predikcie program vypočíta správne a na grafe vidno že chyba postupe klesala pri 500 epochách.

Problém OR:

Vstupy (*X*)	Skutočné (Y)	Predikácie	Zaokrúhlenie
[0,0]	0	~0.121	0
[0,1]	1	~0.907	1
[1,0]	1	~0.930	1
[1,1]	1	~0.978	1



Program pre problem OR taktiež funguje správne. Na výsledných hodnotách je vidno že program si bol takmer istý pri predikciách kedže pre nulu sú veľmi malé a pre 1 sú veľmi vysoké. Na grafe taktiež vidno, že chyba sa postupne počas trénovania znižovala a klesala.

5. Referencie:

- Mastering MNIST Classification with PyTorch: A Step-by-Step Tutorial | by Bragadeesh Sundararajan | Medium
- Implementing Backpropagation in Python: Building a Neural Network from Scratch — Andres Berejnoi | by Andrés Berejnoi | Medium

6. Záver

Táto dokumentácia opísala implementáciu back propagation algoritmus. Program som otestoval pre jednu aj pre dve skryté vrstvy. Pre dve vrstvy mi prigram dokázal dať lepšie aproximácie ako pre jednu vrstvu.