Prolog Neural Network
Funkcionální a logické programování – 2. projekt

Bc. Marek Sedláček (xsedla1b)

1 Prolog Neural Network (PLNNet)

PLNNet je projekt na ukázku implementace jednoduché neuronové sítě v jazyce Prolog (SWI-Prolog). Funkce pracují zásadně s maticemi pro ulehčení programování a projekt obsahuje ukázkové a interaktivní funkce.

2 Implementace

Projekt je rozdělený do 3 modulů, a to: `input2`, `plnnet` a `nn`.

- Modul `input2` je lehce modifikovaná verze skriptu se stejným jménem poskytnutá k tvorbě projektu.
- Modul `nn` obsahuje funkce pro práci s maticemi, soubory dat pro trénování a představuje neuronovou síť samotnou. Funkce a operátory pro práci s maticemi jsou z modulu exportovány a je tedy možné je snadno využít pro případné úpravy vstupních nebo výstupních matic.
- Modul `plnnet` je ukázkový kód práce s modulem `nn` včetně interaktivní funkce pro natrénování neuronové sítě a následné predikce dle vstupů uživatele.

2.1 Proces trénování a predikce

Před výpočtem predikce dle uživatelského vstupu je potřeba provést trénování, což je možné pomocí funkce `training/6`, která jako argumenty přijímá aktivační funkci, trénovací vstupy, očekávané výstupy, váhy, počet epoch na trénování a jejím výstupem jsou nově natrénované váhy pro jednotlivé iterace trénování.

Aktivačních funkcí nabízí modul `nn` více:

- `sigmoid/3` – sigmoida,
- `tanh/3` – hyperbolická funkce,
- `softsign/3` – funkce softsign,
- `gaussian/3` – Gaussova funkce.

Je-li jakákoliv funkce volána s hodnotou `true` jako druhým argumentem, vrátí tato funkce svou derivaci, což je potřeba při trénování v procesu backtrackingu.

Funkce `training/6` provádí volání forward funkce následované backward funkcí pro všechny iterace.

Po natrénování vah je možné zavolat predikční funkci `predict/6` opět s aktivační funkcí, vstupem pro predikci a natrénovanými váhami. Výstupem je predikce pro vstupní hodnoty.

3 Spuštění

Pro spuštění programu stačí pouze interpretu `swipl` jako první argument zadat soubor `plnnet.pl`:

```
swipl plnnet.pl
```

Nebo pomocí Makefile s možností `interactive`:

```
make interactive
```

Tímto se spustí interaktivní textové rozhraní, kde stačí zadat počet epoch pro trénování a následně data na predikci (oddělené mezerou).

Další možností je překlad do binárního souboru pomocí Makefile (příkaz `make`). Tato binární verze však volá pouze funkci `main` a není tedy možné zde provádět testování a predikce bez trénování (pro tento přístup je vhodný `interactive` mód).

Trénovací data jsou načtená ze souboru `train`, výstupy pro jednotlivá data ze souboru `outputs` a váhy ze souboru `weights`. Kde formát těchto souborů vždy obsahuje na jednom řádku jedny vstupní data obalená hranatými závorkami, jednotlivé hodnoty jsou odděleny čárkou a řádek je ukončen tečkou. Tedy například:

```
[4.2, 3.14, 0.1].
[1.1, 5.42, 8.4].
```

Tento soubor by reprezentoval matici:

$$\begin{vmatrix} 4{,}2 & 3{,}14 & 0{,}1 \\ 1{,}1 & 5{,}42 & 8{,}4 \end{vmatrix}$$

Pro provedení predikce je tato hodnota vypsána a natrénované váhy jsou uloženy do predikátu `trainedWeights/1`, pro případné další predikce, které je možné provést například následovně:

```
trainedWeights(W), predict(sigmoid, [[1.0, 0.0, 0.0]], W, P).
```

Případně je možné zavolat funkci `main/0`, která spustí opět interaktivní rozhraní.

3.1 Výchozí neuronová síť

Výchozí nastavení vstupních souborů neuronové sítě jsou data, kde se má síť naučit, že očekávaným výstupem predikce je první hodnota ve vstupu booleanových hodnot zapsaných v pohyblivé řádové čárce podobě (pro `[[1.0 0.0 0.0]]` je to `[[1.0]]`).

3.2 Testování sítě

Pro testování sítě na výchozí vstupní data je možné zavolat funkci `runTests/1` s počtem epoch na natrénování a tato funkce poté načte testovací data ze souboru `test` a vypíše pro jednotlivé vstupy očekávanou predikci (první hodnotu) a skutečnou predikci.