VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA INFORMAČNÍCH TECHNOLOGIÍ

Praktická úloha řešená sítí RCE

Projekt do předmětu SFC, 2019/2020

Tomáš Blažek (xblaze31)

23.11.2019

1 Úvod

Cílem bylo vytvořit aplikaci, která implementuje funkci neuronové sítě RCE (Restricted Coulomb Energy) a následně vhodně zvolit praktickou úlohu, která by byla neuronovou sítí RCE řešena. Tím je myšlena úloha, u které by byla možná jednoduchá validace výsledků odezvy neuronové sítě a případně nenáročné zadávání vstupu. Primárním cílem je ukázat především funkci a vlastnosti neuronové sítě této architektury.

2 Architektura neuronové sítě RCE

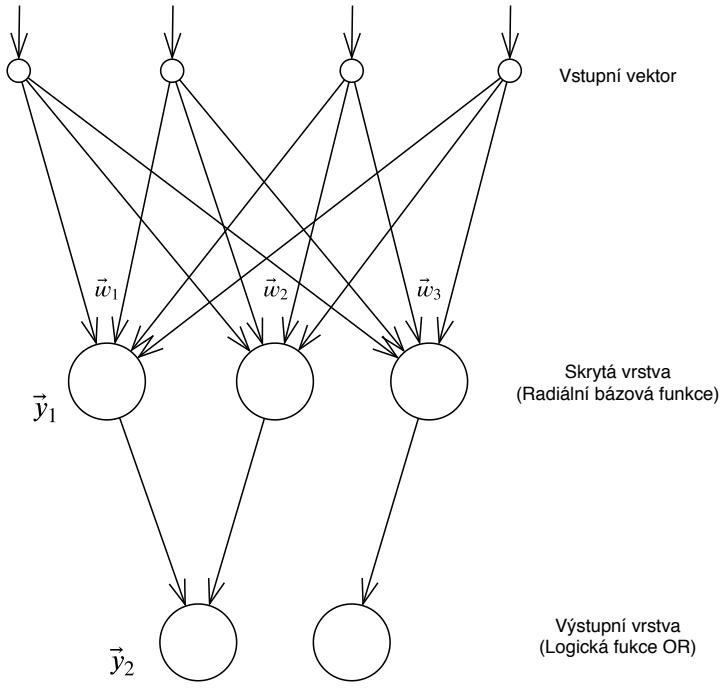
Jedná se o dvouvrstvou neuronovou síť s proměnnou topologií. První vrstva, která se nazývá skrytá, obsahuje radiální bázovou funkci a skokovou aktivační funkci. Druhá vrstva, která je vrstvou výstupní, má odezvu logické funkce OR. Výpočet vnitřího potenciálu neuronů skryté vrstvy u_k je podle vzorce 1, kterým se jinak řečeno spočítá vzdálenost mezi vektorem vah $\vec{w_k}$ (střed hyperkoule) a vstupním vektorem $\vec{i}$.[3]

$$u_k = \sqrt{\sum_{i=1}^{n}(i_i - w_{ki})^2} \tag{1}$$

Aktivační funkce skryté vrstvy je dána vzorcem 2. Jedná se o skokovou funkci a nabývá buď hodnotu 1 anebo hodnotu 0. To podle toho, zda-li je vzorek uvnitř hyperkoule daného neuronu k (tj. 1), anebo se nachází mimo ni (tj. 0). Velikost hyperkoule je definována poloměrem r_k, pro každý neuron.

$$y_k = \begin{cases} 1 & : u_k \leq r_k \\ 0 & : u_k > r_k \end{cases} \tag{2}$$

Neuronové síťě s radiální bázovou funkcí, jako jsou právě neuronové síťě RCE, jsou vhodné například ke klasifikaci a funkční aproximaci v n-dimensionálním prostoru.



Obrázek 1: Architektura RCE neuronové sítě

3 Řešená úloha

Jako praktická úloha, která má být řešena neuronovou síťí RCE, byla zvolena úloha balančních vah. Byla zvolena především z důvodu toho, že výsledky poskytnuté neuronovou sítí je možné snadno ověřit.

Obrázek 2: Ilustrační obrázek balančních vah[2]

Úloha spočívá v tom, že máme k dispozici balanční váhy se dvěma miskam, jak je uvedeno na obrázku 2. Do obou misek je možné umístit závaží o určité hmotnosti a vzdálenosti od středu váhy. Množina dat, na níž je síť trénovaná[1], byla získána z Machine Learning Repository[2] a obsahuje vektory ve tvaru:

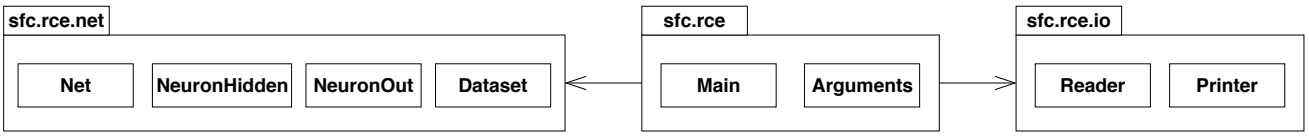
```
Class-Name, Left-Weight, Left-Distance, Right-Weight, Right-Distance
```

Jednotlivé atributy mohou nabývat různých hodnot:

- **Class-Name**: Označuje náklon balanční váhy a může nabývat tří hodnot. (Left, Balanced, Right)
- **Left-Weight**: Váha, která je umístěna na levou část váhy. Trénovací množina obsahuje následující hodnoty. (1, 2, 3, 4, 5)
- **Left-Distance**: Vzdálenost od středu váhy, na které je položeno závaží na levou misku. Trénovací množina obsahuje následující hodnoty. (1, 2, 3, 4, 5)
- **Right-Weight**: Váha, která je umístěna na pravou část váhy. Trénovací množina obsahuje následující hodnoty. (1, 2, 3, 4, 5)
- **Right-Distance**: Vzdálenost od středu váhy, na které je položeno závaží na pravou misku. Trénovací množina obsahuje následující hodnoty. (1, 2, 3, 4, 5)

4 Implementace

Aplikace je rozložena do několika balíčků (packages) rozčleněných podle funkční logiky. Do hlavního balíčku spadají třídy `Main` a `Arguments`, které se starají o zpracování vstupních argumentů a hlavního chodu aplikace. Dále jsou to balíčky `io` a `net`.



Obrázek 3: Diagram balíčků aplikace

[1] Kokrétní dataset je možné získat na adrese http://mlr.cs.umass.edu/ml/datasets/Balance+Scale

4.1 Balíček net

Tento balíček obsahuje čtyři třídy, které implementují algoritmus RCE neuronové sítě. Základní třídou tohoto balíčku je třída `Net`, která obsahuje fukce pro inicializaci sítě, trénování sítě, klasifikaci apod. Dále obsahuje třídy `NeuronHidden` a `NeuronOut`, které slouží k ukládání jednotlivých vzahů mezi neurony v síti. Jako poslední v tomto balíčku je třída `Dataset`, jejímž úkolem je práce s množinou vektorů, s níž neuronová síť pracuje.

4.2 Balíček io

Obsahuje třídy `Reader` a `Printer`, které implementují vstupní a výstupní operace. Třída `Reader` načítá vstupní data v podobně řetězce a převádí je do vnitřní implementace v podobně datasetu (třída `Dataset`). Dále se tato třída stará o komunikaci s uživatelem. Třída `Printer` slouží pro výpis dat uživateli ve formátované podobě.

4.3 Implementační technologie

Technologie použité pro implementaci byly zvoleny následující. Programovací jazyk Java (verze 1.8) a nástroj Apache Ant sloužicí pro kompilaci zdrojových kódú, sestavení výchozí operace a vygenerování programové dokumentace.

5 Manuál

Aplikaci je nutno před použitím přeložit k vytvoření spustitelného archivu jar. Následující podkapitoly popisují postupy pro překlad, spuštění a ovládání programu.

5.1 Překlad

V kořenovém adresaři je soubor `build.xml`, který obsahuje informace potřebné k překladu pomocí nástroje Apache Ant. Stačí tedy v tomto adresáři použít příkaz:

```
ant compile
```

Ten provede překlad zdrojových kódů. Výsledky překladu jsou následně uloženy do složky adresáře `build`. Spustitelná aplikace je vygenerována v kořenovém adresáři jako jar soubor s názvem `sfc-rce.jar`. Jako další akci provede tento příkaz vygenerování programové dokumentace do adresáře `doc/program`.[1]

5.2 Použití

Aplikace se spustí pomocí příkazu příkazu:

```
java -jar sfc-rce.jar
```

Následně se bude aplikace ptát na vstupní vzorek v podobně jednoduchých dotazů. Nicméně v tomto momentě je v aplikaci nenatrénovaná neuronová síť. Tím pádem budou všechny odpovědi sítě ve tvaru „Unkwown“. Pro spuštění tréninku sítě je nutné použít přepínač `-t <soubor_s_trénovací_množinou>`.

Popis přepínačů

Při spuštění aplikace je možno využít přepínačů a nakonfigurovat si tak neuronovou síť před použitím podle potřeby.

- `-t <soubor_s_trénovací_množinou>`
  Tento přepínač umožňuje zvolit soubor s trénovací množinou, kterou má být síť natrénována. Formát obsahu toho souboru by měl mít strukturu takovou, že na každé řádku se vyskytuje jeden vektor (vzorek) trénovací množiny a vektor je ve formátu sekvence čísel oddělených čárkou. První hodnota v tomto sloupci nemusí být číslo, jelikož označuje cílovou třídu daného vzorku.

- `-v <soubor_s_validační_množinou>`
  Podobně jako předchozí přepínač nastavuje nastavuje soubor. Tentokrát je to ale soubor s validační množinou. Při zadání tohoto přepínače neuronová síť klasifikuje všechny vzorky v souboru, vypíše na výstup výsledky této klasifikace a aplikace se ukončí.

- `-R <maximální_velikost_hyperkoule>`
  Aplikace umožňuje nakonfigurovat maximální velikost hyperkoulí při vytváření/učení neuronové sítě.

- `-r <zmenšovací_poměr_hyperkoule>`
  Aplikace umožňuje nakonfigurovat poměr jakým se budou hyperkoule zmenšovat při trénování neuronové sítě. Jeho hodnota by se měla pohybovat v intervalu $(0, 1)$.

5.3 Závěr

Neuronová síť dosahovala úspěšnosti klasifikace kolem 80 %. Síti byly předloženy dvě množiny, jedna určená na natrénování sítě a druhá pro validaci odezvy. Ty byly vytvořeny z již zmíněné množiny dat z kapitoly 3. Vzorky byly náhodně zamíchány a prvních 525 vzorků bylo zvoleno jako tréninková množina a zbylých 100 jako validační množina. Úspěšnost klasifikace závisela z velké části na zvolených parametrech sítě, a to na maximální velikosti hyperkoule a zmenšovacím poměru.

Reference

[1] APACHE. *The Apache Ant Project*. 2019. Dostupné na: https://www.fit.vutbr.cz/study/courses/SFC/private/19sfc_3.pdf.

[2] DUA, D. a GRAFF, C. *UCI Machine Learning Repository*. 2017. Dostupné na: http://archive.ics.uci.edu/ml.

[3] ZBOŘIL, F. *Neuronové sítě s RBF neurony* [online]. říjen 2019 [cit. 2019-11-05]. Dostupné na: https://www.fit.vutbr.cz/study/courses/SFC/private/19sfc_3.pdf.