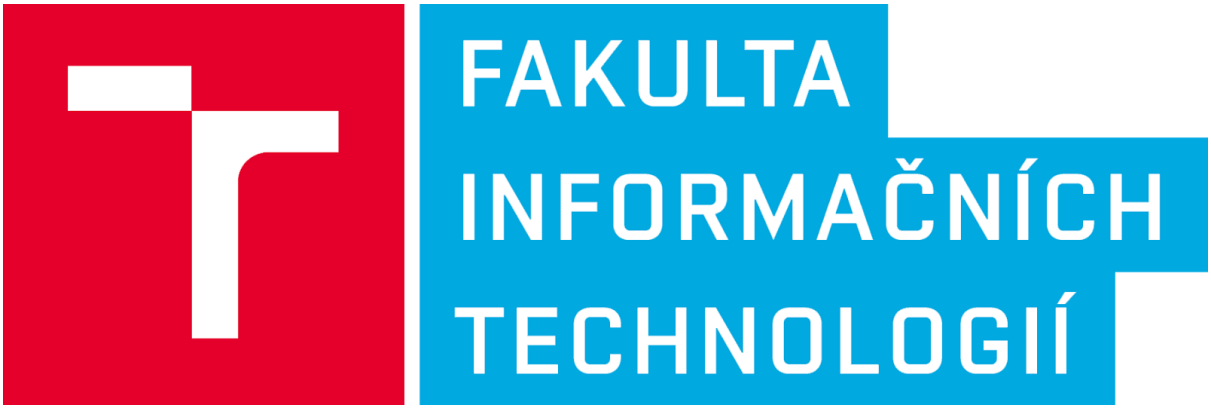


Omezený Boltzmannův Stroj

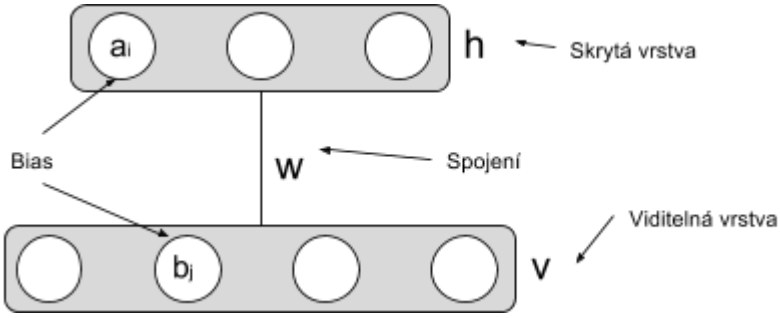
30.11.2017

Tomáš Coufal

xcoufa09

Omezený Boltzmannův stroj

Jedním z typů neuronových sítí dovolující tzv. učení bez učitele je Omezený Boltzmannův stroj. Jedná se o dvouvrstvou síť neuronů, kdy jedna vrstva je skrytá, druhá viditelná. Každý neuron je binární a má spojení se všemi neurony druhé vrstvy, naopak není propojen s neurony ve stejné vrstvě (proto se nazývá omezeným Boltzmannovým strojem).



Navíc si každý neuron, kromě své binární hodnoty h_i (resp. v_j po viditelnou vrstvu), pamatuje hodnotu biasu a_i (resp. b_j). Také je třeba zmínit, že spojení mezi neurony je symetrické, tedy platí, že $W_{ij} = W_{ji}$.

Učení neuronové sítě

Tato síť využívá pro výpočet energie určité konfigurace aktivních neuronů předpis:

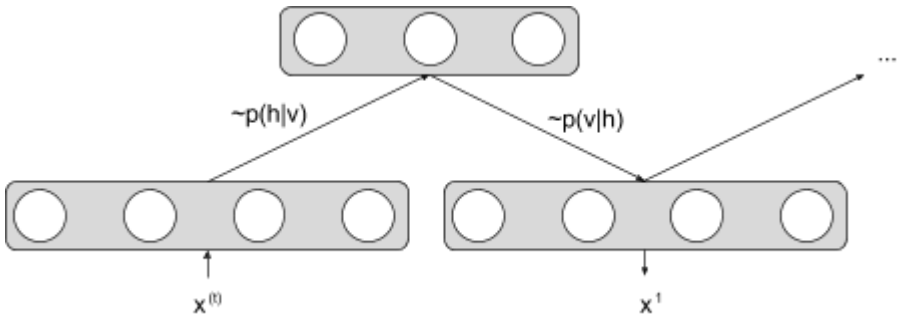
$$E(v,h) = -\sum_i \sum_j W_{ij} h_i v_j - \sum_j b_j v_j - \sum_i a_i h_i$$

Tedy zohledňuje váhy, kde jsou oba konce spojení aktivní a zároveň použije ty odchylky (biasy), kde je daný neuron aktivní.
Následně z této energie odvodí pravděpodobnost aktivace daného neuronu:

$$p(h|v) = \prod_i \frac{exp(h_i \sum_j W_{ij} v_j + a_i h_i)}{1 + (a_i + \sum_j W_{ij} v_j)} = \prod_i p(h_i|v) \qquad p(v|h) = \prod_j \frac{exp(v_j \sum_i W_{ij} h_i + b_j v_j)}{1 + (b_j + \sum_i W_{ij} h_i)} = \prod_j p(v_j|h)$$

$$p(h_i = 1|v) = \frac{1}{1 + exp(-a_i - \sum_j W_{ij} v_j)} = sigm(a_i + \sum_j W_{ij} v_j) \quad p(v_j = 1|h) = \frac{1}{1 + exp(-b_j - \sum_i W_{ij} h_i)} = sigm(b_j + \sum_i W_{ij} h_i)$$

Síť je trénována pomocí algoritmu Contrastive Divergence[1] (CD) modelované jako Markovův řetězec, snažící se maximalizovat (logaritmickou) pravděpodobnost trénovacího vzorku. Tedy:



[1] Hinton G., Training Products of Experts by Minimizing Contrastive Divergence. Neural Computation 2002. http://www.cs.toronto.edu/~hinton/absps/nccd.pdf

V praxi se pro omezený Boltzmannův stroj používá CD-1, tedy stačí pokud se provedou pouze následující kroky:

1. Testovací vzorek se použije jako konfigurace aktivních neuronů viditelné vrstvy
2. Pozitivní fáze CD:
    a. Spočte se pravděpodobnost aktivací neuronů skryté vrstvy
    b. Aktivuje se skrytá vrstva podle právě spočítaných pravděpodobností
3. Negativní fáze CD:
    a. Spočte se pravděpodobnost aktivací neuronů ve viditelné vrstvě
    b. Aktivují se neurony viditelné vrstvy podle právě získaných pravděpodobností
    c. Spočte se pravděpodobnost aktivací neuronů skryté vrstvy
    d. Aktivuje se skrytá vrstva podle právě spočítaných pravděpodobností
4. Upraví se váhy spojení na základě pravděpodobností pro skrytou vrstvu v pozitivní a negativní fázi a aktivace viditelné vrstvy v negativní fázi
5. Upraví se bias všech neuronů:
    a. Skrytá vrstva počítá z aktivace z pozitivní fáze a pravděpodobnosti v negativní
    b. Viditelná z testovacího vzorku a vlastní aktivace v negativní fázi

Tento postup se opakuje pro každý trénovací vstup, po dobu určitého množství trénovacích epoch. Každý vzorek ovlivňuje síť s předem daným učícím faktorem (learning rate).

Úprava vah a biasu se tedy provede podle následujících pravidel:

$$W_{ij} = rate \cdot \left[p(h_i|x) \cdot x_j^{(t)} - p(h_i|v) \cdot v_j\right] / samples$$
$$a_i = rate \cdot \left[h_i - p(h_i|v)\right] / samples$$
$$b_j = rate \cdot \left[x_j^{(t)} - v\right] / samples$$

Odpověď na vstupní vzorek

Ve chvíli, kdy je síť natrénována, je schopná odpovědi na reálný vstup. Ten zpracuje podobně jako při učení, jen neupravuje své váhy a bias. Tedy provede se následující:

1. Použije vstup jako konfiguraci aktivovaných neuronů viditelné vrstvy
2. Spočte pravděpodobnost aktivace neuronů skryté vrstvy na základě biasu skrytých neuronů a vah spojení, kterými jsou aktivovány
3. Spočte pravděpodobnost odpověď sítě z biasu viditelných neuronů a vah aktivovaných spojení ze skryté vrstvy

Implementace RBM

Program byl implementován v jazyce C++ a využívá třídy `RBM` a rozhraní `utils`. Interakce z uživatelem přes CLI je obsažena v `main.cpp`.

Třída `RBM`

Model omezeného Boltzmannova stroje je reprezentován touto třídou. Při vytvoření objektu `RBM`, je síť vždy nastavena na tyto implicitní hodnoty:

- Hodnota a_i pro každý neuron skryté vrstvy je nastavena na $a_i = 0$
- Hodnota b_j pro každý neuron viditelné vrstvy je nastavena na $b_j = 0$
- Každá váha W_{ij} je nastavena na výchozí náhodnou hodnotu z intervalu $<-\frac{1}{J}, \frac{1}{J}>$, kde J je počet neuronů viditelné vrstvy

`RBM::train`

Tato metoda zajistí natrénování sítě podle vektoru testovacích vstupů.

`RBM::probability_of_h_given_v` a `RBM::probability_of_v_given_h`

Metody, které zjistí pravděpodobnosti aktivace jedná vrstvy neuronů na základě konfigurace té druhé. Dále provede aktivaci neuronů vrstvy.

`RBM::propagate_from_visible` a `RBM::propagate_from_hidden`

Pomocné funkce zjišťující pravděpodobnost aktivace pro určitý, konkrétní, neuron.

`RBM::run`

Zjišťuje odpověď sítě na uživatelský vstup.

Rozhraní `utils`

Poskytuje pomocné funkce pro načtení a zpracování vstupních dat do formátu akceptovaného třídou `RBM`.

Kompilace a spuštění

Program lze sestavit vyvoláním příkazu `make` následovně:

```
$ make
g++ -std=c++11 -Wall -Werror -pedantic -c rbm.cpp -o rbm.o
g++ -std=c++11 -Wall -Werror -pedantic rbm.o main.cpp -o rbm
```

Samotné spuštění je poté provedeno voláním pravidla `make run` či spuštěním sestaveného programu:

```
$ make run
… nebo také
$ ./rbm
```

Program následně provede uživatele počátečním nastavením sítě, kdy je možné změnit výchozí chování, či toto nastavení ponechat:

```
Initialization phase
--------------------------------------------------------------------------------
Specify learning rate (default = 0.1):
Training epochs amount (default = 10000): 1000
Neurons in visible layer (default = 6): 8
Neurons in hidden layer (default = 3): 4
```

Následně je zobrazeno nastavení sítě před začátkem učení:

```
Initialized Restricted Boltzmann Machine
--------------------------------------------------------------------------------
Bias of visible nodes (8):      [0.000000, 0.000000, 0.000000, 0.000000, 0.000000,
0.000000, 0.000000, 0.000000]
Bias of hidden nodes (4):       [0.000000, 0.000000, 0.000000, 0.000000]
Weights:
          |      #0      #1      #2      #3      #4      #5      #6      #7
-------------------------------------------------------------------------------
Hidden #0 |  0.0713  0.0104  0.0087  0.1109 -0.0325 -0.0251  0.0154  0.0349
Hidden #1 |  0.0411 -0.0498 -0.0074  0.0260  0.0147 -0.0669  0.0949 -0.0319
Hidden #2 | -0.0964  0.0134 -0.0542  0.0857 -0.0044 -0.1108 -0.1153  0.0836
Hidden #3 | -0.0132 -0.0898 -0.0102 -0.0492 -0.0737  0.0414 -0.1000  0.1225
```

Poté je uživatel vyzván k nahrání testovacích dat. Ty jsou uložena ve formátu CSV, kdy každý řádek reprezentuje jeden testovací vzorek. Řešení načítání dat je natolik robustní, že pokud testovací vzorek obsahuje širší vstup než kolik má zkoušená síť neuronů ve vstupní vrstvě, jsou tyto přesahující

hodnoty ignorovány. Stejně tak, pokud je vzorek užší, je doplněn neaktivními elementy. Odevzdané řešení obsahuje 2 sety testovacích dat: `sample.csv` a `sample2.csv`:

```
Load training data
------------------------------------------------------------------------------
File with training data (default = samples.csv):
[1, 1, 1, 0, 0, 0, 0, 0]
[1, 0, 1, 0, 0, 0, 0, 0]
[1, 1, 1, 0, 0, 0, 0, 0]
[0, 0, 1, 1, 1, 0, 0, 0]
[0, 0, 1, 0, 1, 0, 0, 0]
[0, 0, 1, 1, 1, 0, 0, 0]
```

Následně síť konfrontuje uživatele se stavem sítě po natrénování a vyzve jej k zadání vstupních dat.

```
Trained network (1000 epochs)
------------------------------------------------------------------------------
Bias of visible nodes (8):      [0.066667, -0.783333, 1.783333, -0.766667,
-0.183333, -2.266667, -2.266667, -2.250000]
Bias of hidden nodes (4):       [0.332437, 0.331328, 0.080717, -0.441498]
Weights:
          |      #0      #1      #2      #3      #4      #5      #6      #7
-------------------------------------------------------------------------------
Hidden #0 |  0.9096  0.2188  1.5981 -0.7873 -0.6505 -1.3842 -1.3829 -1.3558
Hidden #1 |  0.1025 -0.3892  1.8640  0.0281  0.4085 -1.3205 -1.2462 -1.2886
Hidden #2 | -3.9266 -3.0627  0.4654  2.2113  3.7435 -1.2448 -1.2558 -1.0873
Hidden #3 |  2.9467  1.7045  0.8540 -2.4054 -3.1775 -1.1406 -1.2271 -1.1100
```

Vstupní data jsou zadávána interaktivně ve formě vektoru čísel, kdy jednotlivé položky jsou odděleny čárkou a jakákoliv číselná nenulová hodnota je považována za aktivní, naopak prázdné pole či nečíselná hodnota je interpretována jako neaktivní:

```
Test data
------------------------------------------------------------------------------
Please enter the sample (comma separated): +
Sample [0, 0, 0, 0, 0, 0, 0, 0]
--> Result [0.441964, 0.140818, 0.987572, 0.268943, 0.593680, 0.007137, 0.007168,
0.008246]
Please enter the sample (comma separated): 1,,1
Sample [1, 0, 1, 0, 0, 0, 0, 0]
--> Result [0.976743, 0.649332, 0.997064, 0.023276, 0.033107, 0.002684, 0.002644,
0.002987]
Please enter the sample (comma separated): ,,,1,1,1,,1
Sample [0, 0, 0, 1, 1, 1, 0, 1]
--> Result [0.023392, 0.021443, 0.925973, 0.798700, 0.971016, 0.024326, 0.024313,
0.028853]
```

Program ukončíte voláním `<CTRL-C>`.

Závěr

Implementoval jsem neuronovou síť typu omezený Boltzmannův stroj, která je schopná v uživatelem zvolené konfiguraci natrénovat své odpovědi a poté poskytovat klasifikaci daného vstupu. Trénování je provedeno pomocí Contrastive Divergence. K pochopení problematiky mi velmi pomohla série videí od Huga Larochelle[2], který velmi dobře vysvětluje matematiku v RBM se skrývající.

[2] https://www.youtube.com/watch?v=p4Vh_zMw-HQ