Implementace metod získávání znalostí

Řešení

Jakub Zárybnický (xzaryb00), Jan Hammer (xhamme00)

19. 12. 2019

1 Zadání

Kohonenova mapa (self-organizing map, SOM) je typ neuronové sítě trénovaný pomocí učení bez učitele (unsupervised learning). Používá se pro snížení dimenzionality dat, často pro projekci do 2D/3D prostoru za účelem vizualizace. Také se používá pro klasifikační úkoly, kdy výsledkem učení jsou shluky. Podobnými metodami jsou k-means clustering nebo support vector machines, které mohou dostávat lepší výsledky v klasifikačních úkolech, ale zato nemají tak čitelné vizualizační výstupy.

Cílem našeho projektu bylo implementovat Kohonenovu mapu i vizualizace nad ní a demonstrovat funkčnost nad několika sadami dat.

2 Přístup

Mapu jsme implementovali v jazyce Python pomocí knihovny pro práci s maticemi Numpy. Pro vizualizaci jsme použili další knihovnu - ne maplotlib, jak bylo původně v plánu, ale hlavně kvůli animacím jsme použili graphics-py. Poslední externí knihovna, použitá čistě z pohodlnosti, je Click, knihovna pro tvorbu rozhraní příkazové řádky.

Vytvořili jsme knihovnu a aplikaci skládající se ze dvou modulů `som.map` a `som.graphics` a ze spustitelné aplikace `som.py`. Aplikace obsahuje tři oddělené příkazy - `generate` pro generování náhodných dat (volitelně ve shlucích), `train` pro natrénovaní mapy s řadou nastavitelných parametrů a `visalize` pro zobrazení již natrénovaného modelu.

Dále jsme vytvořili dva skripty `mnist.py` a `iris.py`, které demonstrují použití API vytvořené knihovny.

2.1 Algoritmus

Jeden krok učení Kohonenovy mapy má dva podkroky:

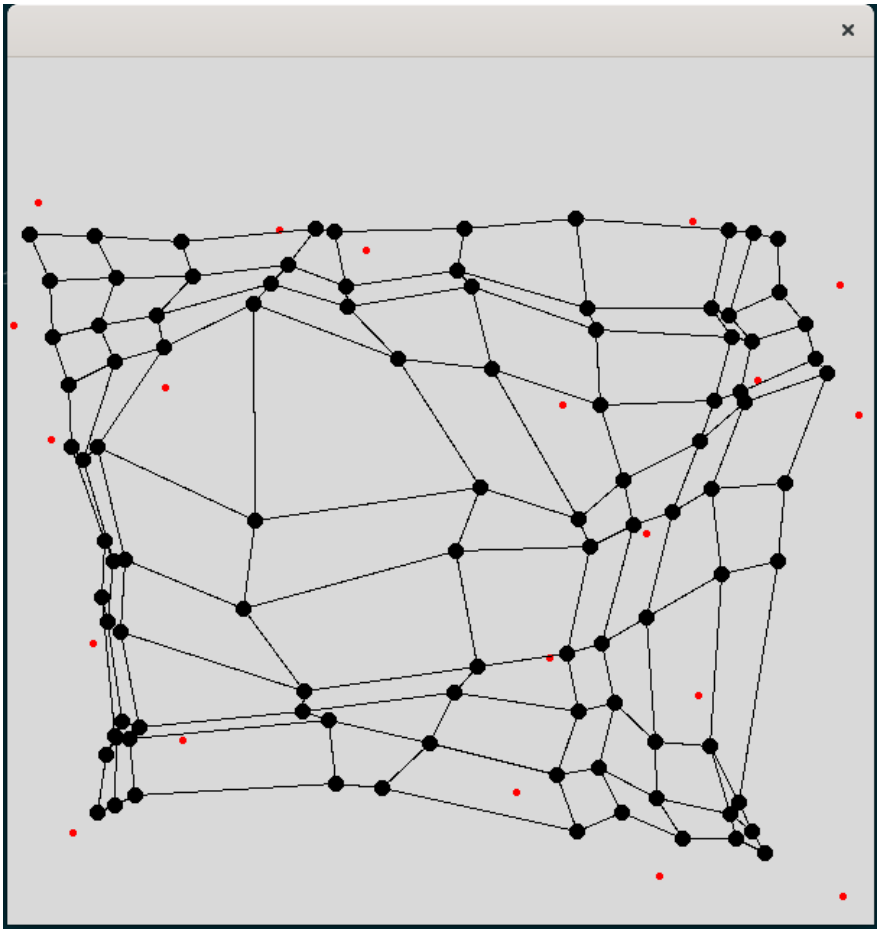
- najít uzel, jehož váhové parametry jsou nejbližší aktuálně trénovanému vzorku, tzv. BMU (Best Matching Unit)
- a posunout takto nalezený uzel a jeho sousedy blíže k trénovanému vzorku s vahou odpovídající obrácené vzdálenosti souseda od BMU (používáme tzv. *Manhattan distance*).

3 Výsledky

3.1 Random

```
./som.py generate 20 > random.csv
./som.py train --animate --animate-interval 5 \
  --size 10x10 --rate 0.2 --epochs 15 -m random.npz \
  random.csv
./som.py visualize random.npz
```

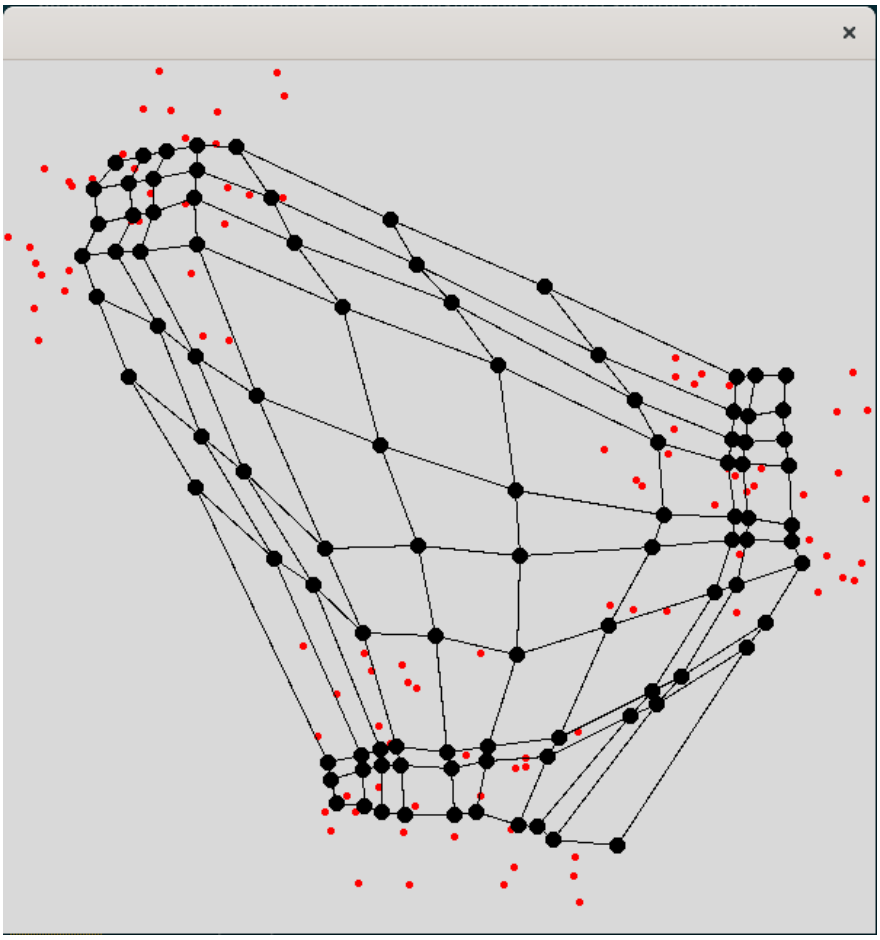
První příkaz vygeneruje 20 náhodných vzorků o délce 2, druhý nad nimi natrénuje síť o velikosti 10x10 a zároveň při tréninku zobrazuje animaci, jak se síť mění při učení. Natrénovanou síť uloží do souboru, odkud ji pak načte poslední příkaz, který vizualizuje už netrénovanou síť.



3.2 Clusters

```
./som.py generate --clusters 3 300 > clusters.csv
./som.py train --animate --animate-interval 1 \
  --size 10x10 --rate 0.2 --epochs 5 -m clusters.npz \
  clusters.csv
```

Sada příkazů generuje sadu 300 vzorků o délce 2 seskupenou do tří shluků, nad nimi natrénuje síť a během tréninku vizualizuje, jak se síť postupně učí.



3.3 MNIST

```
./mnist.py -S 20x20 --skip-rows 1 train.csv
```

Program rozdělí vstupní data na testovací a trénovací množinu a pak paralelně trénuje Kohonenovu mapu a klasifikátor na ní založený - každý uzel má ještě váhový vektor s one-hot zakódovaným výsledkem klasifikace, který se trénuje stejně jako mapa samotná, s propagací do sousedních uzlů.

Na závěr se program pokusí klasifikovat data testovací množiny. Nejvyšší procento správně klasifikovaných vzorků, co jsem viděl, bylo 76.52% - zamícháním tréninkových dat před začátkem učení jsou ale výsledky nedeterministické, pravidelně dosahujeme výsledků kolem 68% a další úpravou parametrů nebo třeba zvětšením by bylo možné dosáhnout ještě lepších výsledků.

```
[inuits@nixos:~/Projects/zzn-self-organizing-maps]$ ./mnist.py -S 20x20 --skip-rows 1 train.csv
Loading CSV... done.
Initializing map... done.
Training..................................................................................................
.......................................................... done.
Classifying... done.
Success rate: 68.73% (5773 out of 8400)
Map:
1  1  1  1  7  9  9  9  9  9  9  9  9  9  9  9  9  4  4
1  1  1  1  7  9  9  9  9  9  9  9  9  9  9  9  9  4  4
1  1  1  1  4  4  9  9  9  9  7  7  7  7  7  7  9  4  4
1  1  1  4  4  4  4  4  9  7  7  7  7  7  7  7  7  4  4
1  1  5  5  4  4  4  4  7  7  7  7  7  7  7  7  7  4  4
5  5  5  5  5  4  4  4  7  7  7  7  7  7  7  7  7  4  4
5  5  5  5  5  5  4  4  7  7  7  7  7  7  7  7  4  4  4
5  5  5  5  5  5  5  5  7  7  7  7  7  7  7  9  4  4  4
5  5  5  5  5  5  5  5  5  8  8  8  8  7  2  2  9  4  4  4
1  5  5  0  0  0  5  5  5  8  8  8  8  8  2  2  2  4  6  6
1  1  1  0  0  0  5  5  5  8  8  8  8  8  2  2  2  6  6  6
1  1  6  6  6  6  5  5  5  8  8  8  8  2  2  2  2  6  6  6
1  1  1  6  6  6  6  5  5  5  8  8  8  2  2  2  2  6  6  6
1  1  1  6  6  6  6  5  3  3  3  3  3  2  2  2  2  2  6  6
1  1  1  6  6  6  6  5  3  3  3  3  3  2  2  2  2  2  6  6
1  1  1  6  6  6  6  3  3  3  3  3  3  2  2  2  2  2  6  6
1  1  1  6  6  6  3  3  3  3  3  3  3  3  2  2  2  0  0  0
1  1  1  1  1  1  8  8  3  3  3  3  3  3  3  3  0  0  0  0  0
1  1  1  1  1  1  8  8  8  3  3  3  3  3  3  3  0  0  0  0  0
1  1  1  1  1  1  8  8  8  3  3  3  3  3  3  3  0  0  0  0  0
```

3.4 Iris

```
./iris.py -epochs 50 --size 10x10 iris-preprocessed.csv
```

Program pracuje stejně jako `./mnist.py`, jen s lehce upraveným zpracováním označení vzorků (labels). Při učení zobrazuje rozložení mapy v 2D prostoru - první okno rozložení na základě lístků kališních (první dva rozměry dat), druhé na základě listů korunních (druhé dva rozměry).

Zde s uvedenými parametry učení dosahujeme úspěšnost ověření nad testovacími daty přes 90% - nejlepší výsledek, co jsem viděl byl 100%.

```
[inuits@nixos:~/Projects/zzn-self-organizing-maps]$ ./iris.py --epochs 50 -S 10x10 iris-preprocessed.csv
Loading CSV... done.
Initializing map... done.
Training.................................................. done.
Classifying... done.
Success rate: 96.67% (29 out of 30)
Map:
  setosa      setosa      setosa    versicolor  versicolor  versicolor  versicolor  versicolor  virginica   virginica
  setosa      setosa      setosa    versicolor  versicolor  versicolor  versicolor  versicolor  virginica   virginica
  setosa      setosa      setosa    versicolor  versicolor  versicolor  versicolor  versicolor  virginica   virginica
  setosa      setosa      setosa    versicolor  versicolor  versicolor  versicolor  versicolor  virginica   virginica
  setosa      setosa      setosa    versicolor  versicolor  versicolor  versicolor  virginica   virginica   virginica
  setosa      setosa      setosa    versicolor  versicolor  versicolor  versicolor  virginica   virginica   virginica
  setosa      setosa      setosa    versicolor  versicolor  versicolor  versicolor  virginica   virginica   virginica
  setosa      setosa      setosa    versicolor  versicolor  virginica   virginica   virginica   virginica   virginica
  setosa      setosa      setosa    virginica   virginica   virginica   virginica   virginica   virginica   virginica
  setosa      setosa      setosa    virginica   virginica   virginica   virginica   virginica   virginica   virginica
Press any key to continue ...
```

4 Závěr

Adaptovat tyto techniky pro jiné sady dat by mělo být triviální. Pro vizualizaci vícerozměrných dat (např. MNIST) ale chybí implementace tzv. U-matrix vizualizace, která zobrazuje ne absolutní umístění uzlu podle jednotlivých souřadnic ale průměrnou vzdálenost uzlu od jeho sousedů, což je technika použitelná pro libovolně vysoce dimenzionální data.

Naše implementace Kohonenovy mapy ale pracuje správně pro vizualizaci 2D dat a pro klasifikaci libovolně rozměrných dat.

Zdrojový kód pro demonstrované programy je k dispozici na `https://github.com/zarybnicky/zzn-self-organizing-maps`.