APPC - Recommender systems
TP Movie Lens

Stéphane Canu

scanu@insa-rouen.fr, scanu.pages.insa-rouen.fr/pages_perso/

December 10, 2023

Practical session description

This practical session aims at showing how to build a simple recommender system on the movie lens data set

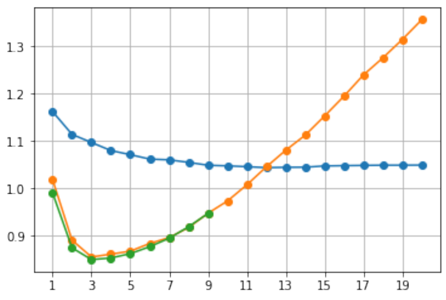

Figure 1: Result of practical session.

Ex. 1 — SVD and weighted SVD on Movie lens

1. The MovieLens100k dataset
   a) What is the Movie Lens data set [1]?
   b) Why is it preferable to begin with the MovieLens 100K Dataset?
   c) download the MovieLens100k dataset that is the ml-100k.zip fiule (size: 5 MB) from `https://grouplens.org/datasets/movielens/100k/`

```
import pandas as pd
import numpy as np
from numpy import random
import scipy
import scipy.sparse

data_dir = "ml-100k/"
data_shape = (943, 1682)

df = pd.read_csv(data_dir + "u.data", sep="\t", header=None)
values = df.values
values[:, 0:2] -= 1
M = scipy.sparse.csr_matrix((values[:, 2], (values[:, 0], values[:, 1])), dtype
    =numpy.float, shape=data_shape)
```

   d) What are the differences between the table `df` and the matrix M?
2. Pre process the data
   a) what is the type sparse?
   b) split the data into two matrices. Use 90% for training and 10 % for testing.

```
ind = np.random.permutation(range(100000))
inda = ind[0:90000]
indt = ind[90000:]
M = scipy.sparse.csr_matrix((values[:, 2], (values[:, 0], values[:, 1])), dtype
    =float, shape=data_shape)
Ma = scipy.sparse.csr_matrix((values[inda, 2], (values[inda, 0], values[inda,
    1])), dtype=float, shape=data_shape)
Mt = scipy.sparse.csr_matrix((values[indt, 2], (values[indt, 0], values[indt,
    1])), dtype=float, shape=data_shape)
```

[1]`https://grouplens.org/datasets/movielens/`

c) compute the mean ratings given by the users to the movies.

```
moy = np.sum(Ma)/np.sum(Ma>.5)
```

d) Compute the mean test error E_t when predicting missing ratings by the mean.

$$E_t = \frac{1}{n_t} \sum_{i=1}^{n_t} (M_t(i) - \hat{\mu})^2$$

where n_t denotes the number of ratings of the test set, M_t the ratings of the test set and $\hat{\mu}$ the mean value on the ratings on training set.

e) Center the data

3. Recommend using SVD
   a) Predict the missing test values using the SVD with an increasing number of component (up to 20). Evaluate the performance of this approach on the test matrix and plot the resulting performance as a function of the number of factors of the SVD used to perform the reconstruction.
   b) What is, in this case, the optimal number of factors?
4. Check if you can improve the predictions using the weighted SVD, that is solving for different k

$$\min_{U \in R^{n\times k}, V \in R^{p \times k}} J_w(U,V) \quad \text{with} \quad J_w(U,V) = \sum_{i=1}^{n} \sum_{j=1}^{p} w_{ij}(M_{ij} - u_i v_j^\top)^2$$

where $w_{ij} = 0$ if M_{ij} is unknown (and else $w_{ij} = 1$)

a) First use the approach proposed by Marlin or Srebro and Jaakkola implementing the following sequence:

```
loop
    $F = W \odot M + (1 - W) \odot Z^k$
    $U, V = svd(F)$
    $Z^{k+1} = UV^\top$
```

where $\odot$ denote the element wise product of two matices.
For details see for instance:

- Marlin, B. (2004). Collaborative filtering: A machine learning perspective (pp. 2239-2239). Toronto: University of Toronto.
- Srebro, N., & Jaakkola, T. (2003). Weighted low-rank approximations. In Proceedings of the 20th International Conference on Machine Learning (ICML-03) (pp. 720-727).

5. Solve the penalized weighted SVD problem

$$\min_{U,V} \|M - UV^\top\|_W^2 + \lambda\|U\|^2 + \lambda\|V\|^2$$

with $\lambda = 5$ by implementing the penalized Alternating Least Square(ALS) described as follows: Initialize U and V with the SVD on the full matrix and compute the following sequence

```
loop
    compute $U^\star = \text{argmin}_U \|M - UV^\top\|_W^2 + \lambda\|U\|^2$ with a fix $V$
    compute $V^\star = \text{argmin}_V \|M - UV^\top\|_W^2 + \lambda\|V\|^2$ with a fix $U$
```

For details see for instance:

- Y. Hu, Y. Koren, and C. Volinsky. Collaborative filtering for implicit feedback datasets. In Proc. of ICDM'08, pages 263–272, Pisa, Italy, 2008.

6. Try with Stochastic gradient descent (SGD) using pytorch. See for instance :

- `https://blog.fastforwardlabs.com/2018/04/10/pytorch-for-recommenders-101.html`
- `https://www.kaggle.com/shihabshahriar/pytorch-movielens`

Pick a rating $m_{i,j}$ for user i and for movie j. Given two matrices $U, n \times k$ and $V, n \times k$, is $u_{i\bullet}$ denotes line i of matrix U while $v_{\bullet j}$ is the transpose of V's column j, the prediction for $m_{i,j}$ is given by

$$\hat{m}_{i,j} = u_{i\bullet} v_{\bullet j} + \mu.$$

The associated error is

$$err = \frac{1}{2}\sum_{ij}(\hat{m}_{i,j} - m_{i,j})^2 + \frac{\lambda}{2}\|u_{i\bullet}\|^2 + \frac{\lambda}{2}\|v_{\bullet j}\|^2,$$

so that gradient updates are given by, for a stepsize ρ,

$$u_i \leftarrow u_i - \rho\big((\hat{m}_{i,j} - m_{i,j})v_j^\top + \lambda u_i\big) v_j \leftarrow v_j - \rho\big((\hat{m}_{i,j} - m_{i,j})u_i \ + \lambda v_j\big).$$

The efficiency of the algorithm depends on the parameters:

- ρ the gradient stepsize
- λ, the regularization parameter
- k, the number of factors
- b, the size of the minibatches
- the initialization of U and V

7. Compare with the SVD of the surprise framework: `https://surprise.readthedocs.io`