Multi-purpose Library of Recommender System Algorithms for the Item Prediction Task

Presentation of my Bachelor Thesis

Julius Kolbe

L3S Research Center / Leibniz University of Hanover
Hanover, Germany

July 5, 2013

Contents

Contents

Implicit Feedback

	Anna	Berta	Claudia	Dagmar
The Shawshank Redemption	1		1	
The Godfather		1	1	
The Godfather: Part II		1		1
Pulp Fiction	1	1		1
The Good, the Bad and the Ugly	1		1	

Item Prediction Task

	Anna	Berta	Claudia	Dagmar
The Shawshank Redemption	1		1	?
The Godfather		1	1	?
The Godfather: Part II		1		1
Pulp Fiction	1	1		1
The Good, the Bad and the Ugly	1		1	?

Notation

	Anna	Berta	Claudia	Dagmar
The Shawshank Redemption	1		1	
The Godfather		1	1	
The Godfather: Part II		1		1
Pulp Fiction	1	1		1
The Good, the Bad and the Ugly	1		1	

Items

Users

Interactions

Basket of u

Related Work

- **MyMediaLite** C#, several recommender algorithms
- **PREA** (Personalized Recommandation Algorithms Toolkit) Java, recommender algorithms and evaluation metrics
- **Apache Mahout** Java, machine learning on top of Apache Hadoop
- **Duine Framework** Java, combination of multiple recommender algorithms
- **Cofi** C++, centers on one recommender algorithm
- **Lenskit** Java, toolkit for recommendation and evaluation
- **Scikit-learn** Python, machine learning, no recommender algorithms

Contents

Motivation for recsyslab

Motivation for recsyslab

- Python for easy readable source code

Motivation for recsyslab

- Python for easy readable source code
- Simple to use

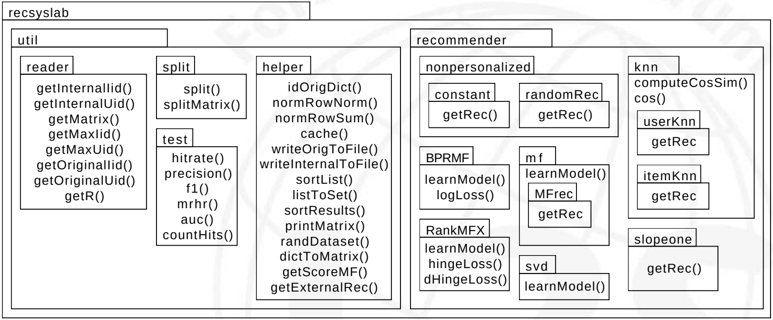
Motivation for recsyslab

- Python for easy readable source code
- Simple to use
- State of the art recommender algorithms

Motivation for recsyslab

- Python for easy readable source code
- Simple to use
- State of the art recommender algorithms
- Open source license: GPLv3

General Structure



Get recsyslab

github.com/Foolius/recsyslab

github.com/Foolius/recsyslab/archive/master.zip

```
$ git clone
    https://github.com/Foolius/recsyslab.git
```

Recommender Algorithms in recsyslab

Recommender Algorithms in recsyslab

- Matrix Factorization [mat, 2013]
  - Bayesian Personalized Ranking (BPRMF) [Rendle et al., 2009]
  - RankMFX [Diaz-Aviles et al., 2012]
  - Ranking SVD [Jahrer and Töscher, 2011]

Recommender Algorithms in recsyslab

- Matrix Factorization [mat, 2013]
  - Bayesian Personalized Ranking (BPRMF) [Rendle et al., 2009]
  - RankMFX [Diaz-Aviles et al., 2012]
  - Ranking SVD [Jahrer and Töscher, 2011]
- k-Nearest-Neighbor[Karypis, 2001]
  - Item-Based
  - User-Based

Recommender Algorithms in recsyslab

- Matrix Factorization [mat, 2013]
  - Bayesian Personalized Ranking (BPRMF) [Rendle et al., 2009]
  - RankMFX [Diaz-Aviles et al., 2012]
  - Ranking SVD [Jahrer and Töscher, 2011]
- k-Nearest-Neighbor[Karypis, 2001]
  - Item-Based
  - User-Based
- Slope One [Lemire and Maclachlan, 2007]

Recommender Algorithms in recsyslab

- Matrix Factorization [mat, 2013]
  - Bayesian Personalized Ranking (BPRMF) [Rendle et al., 2009]
  - RankMFX [Diaz-Aviles et al., 2012]
  - Ranking SVD [Jahrer and Töscher, 2011]
- k-Nearest-Neighbor[Karypis, 2001]
  - Item-Based
  - User-Based
- Slope One [Lemire and Maclachlan, 2007]
- Non-Personalized
  - Constant
  - Random

Matrix Factorization [mat, 2013]

	Anna	Berta	Claudia	Dagmar
The Shawshank Redemption	1	0	1	0
The Godfather	0	1	1	0
The Godfather: Part II	0	1	0	1
Pulp Fiction	1	1	0	1
The Good, the Bad and the Ugly	1	0	1	0

Find W and H so: $\hat{M} = W\ H^\top$.

$$\text{Score}(u, i) = W_u I_i^\top. \tag{1}$$

Matrix Factorization, Training

```
repeat until conversion of W and H:
    u=randomly chosen user
    i=randomly chosen item U interacted with
    j=randomly chosen item U did not interact
      with

    X=H[i] - H[j]
    wx=dot product of W[u] and X
    dloss=(derivative of the loss
           function of wx and 1) * learningRate

    W[u]+=dloss*(H[i] - H[j]) #These three lines
    H[i]+=dloss*W[u]          #have to be
    H[j]+=dloss*-W[u]         #executed at once
```

```
for i in xrange(0, iterations):
    u = random.choice(R.keys())
    userItems = [x[0] for x in R[u]]
    # the positive example
    i = userItems[np.random.random_integers(0, len(userItems) - 1)]
    # the negative example
    j = np.random.random_integers(0, m_items)
    # if  j is also relevant for u we continue
    # we need to see a negative example to contrast the positive one
    while j in userItems:
        j = np.random.random_integers(0, m_items)

    X = H[i] - H[j]
    wx = np.dot(W[u], X)
    dloss = dlossF(wx, y)

    # temp
    wu = W[u]
    hi = H[i]
    hj = H[j]
    if dloss != 0.0:
        # Updates
        eta_dloss = learningRate * dloss
        W[u] += eta_dloss * (hi - hj)
        H[i] += eta_dloss * wu
        H[j] += eta_dloss * (-wu)
        W[u] *= scaling_factorU
        H[i] *= scaling_factorI
        H[j] *= scaling_factorJ
```

k-Nearest-Neighbor [Karypis, 2001]

k-Nearest-Neighbor [Karypis, 2001]

1. Compute similarity of each (item, item) pair

$$\text{sim}(i,j) = \cos(\vec{i},\vec{j}) = \frac{\vec{i}\cdot\vec{j}}{||\vec{i}||_2||\vec{j}||_2} \tag{2}$$

k-Nearest-Neighbor [Karypis, 2001]

1. Compute similarity of each (item, item) pair

$$\text{sim}(i,j) = \cos(\vec{i},\vec{j}) = \frac{\vec{i}\cdot\vec{j}}{||\vec{i}||_2||\vec{j}||_2} \tag{2}$$

2. For each item, save the k items with the highest similarity (= neighbors)

k-Nearest-Neighbor [Karypis, 2001]

1. Compute similarity of each (item, item) pair

$$\text{sim}(i, j) = \cos(\vec{i}, \vec{j}) = \frac{\vec{i} \cdot \vec{j}}{||\vec{i}||_2 ||\vec{j}||_2} \tag{2}$$

2. For each item, save the k items with the highest similarity (= neighbors)
3. Compute the union of the neighbors of the basket of u

k-Nearest-Neighbor [Karypis, 2001]

1. Compute similarity of each (item, item) pair

$$\text{sim}(i,j) = \cos(\vec{i},\vec{j}) = \frac{\vec{i}\cdot\vec{j}}{||\vec{i}||_2||\vec{j}||_2} \tag{2}$$

2. For each item, save the k items with the highest similarity ($=$ neighbors)
3. Compute the union of the neighbors of the basket of u
4. For each item in this set compute the sum of similarities to the basket of u

k-Nearest-Neighbor [Karypis, 2001]

1. Compute similarity of each (item, item) pair

$$\text{sim}(i,j) = \cos(\vec{i},\vec{j}) = \frac{\vec{i}\cdot\vec{j}}{||\vec{i}||_2||\vec{j}||_2} \tag{2}$$

2. For each item, save the k items with the highest similarity (= neighbors)
3. Compute the union of the neighbors of the basket of u
4. For each item in this set compute the sum of similarities to the basket of u
5. Sort by this score and return the first N items

k-Nearest-Neighbor [Karypis, 2001]

	Anna	Berta	Claudia	Dagmar
The Shawshank Redemption	1	0	1	0
The Godfather	0	1	1	0
The Godfather: Part II	0	1	0	1
Pulp Fiction	1	1	0	1
The Good, the Bad and the Ugly	1	0	1	0

$$\mathsf{sim}(i,j) = \cos(\vec{i},\vec{j}) = \frac{\vec{i}\cdot\vec{j}}{||\vec{i}||_2||\vec{j}||_2} = \frac{0}{2} = 0$$

k-Nearest-Neighbor [Karypis, 2001]

	Anna	Berta	Claudia	Dagmar
The Shawshank Redemption	1	0	1	0
The Godfather	0	1	1	0
The Godfather: Part II	0	1	0	1
Pulp Fiction	1	1	0	1
The Good, the Bad and the Ugly	1	0	1	0

$$\mathsf{sim}(i,j) = \cos(\vec{i},\vec{j}) = \frac{\vec{i}\cdot\vec{j}}{||\vec{i}||_2||\vec{j}||_2} = \frac{2}{\sqrt{2}\sqrt{3}}$$

Leave-one-out Protocol [lea, 2013]

Leave-one-out Protocol [lea, 2013]

1. Randomly choose one interaction per user and hide them

Leave-one-out Protocol [lea, 2013]

1. Randomly choose one interaction per user and hide them
2. Train the recommender system with the remaining interactions

Leave-one-out Protocol [lea, 2013]

1. Randomly choose one interaction per user and hide them
2. Train the recommender system with the remaining interactions
3. Get recommendations for every user

Leave-one-out Protocol [lea, 2013]

1. Randomly choose one interaction per user and hide them
2. Train the recommender system with the remaining interactions
3. Get recommendations for every user
4. Compute the chosen evaluation metric with the hidden items and the recommendations

Evaluation Metrics in recsyslab

- Hitrate/Recall@N [Karypis, 2001, Sarwar et al., 2000]
- Precision [Sarwar et al., 2000]
- F1 [Sarwar et al., 2000]
- Mean Reciprocal Hitratea (MRHR) [Ning and Karypis, 2011]
- Area under the ROC (AUC) [Rendle et al., 2009]

Hitrate/Recall@N [Karypis, 2001, Sarwar et al., 2000]

$$\text{Recall@N} = \frac{\sum_{u \in U} H_u \cap \text{topN}_u}{|H|} \qquad (3)$$

H	hidden interactions
H_u	the hidden interaction of u
U	set of users
$topN_u$	N recommendations for u

Contents

MovieLens Dataset

- 100 000 interactions
- 943 users
- 1682 items
- Text file with columns separated with a single tab
- Provided by the GroupLens research lab

Movielens Dataset Excerpt

user	item	rating	time stamp
196	242	3	881250949
186	302	3	891717742
22	377	1	878887116
244	51	2	880606923
166	346	1	886397596
298	474	4	884182806
115	265	2	881171488
253	465	5	891628467
305	451	3	886324817
6	86	3	883603013
62	257	2	879372434
286	1014	5	879781125
200	222	5	876042340

Contents

Outlook

Outlook

- More algorithms

Outlook

- More algorithms
- Refine hyperparameters with more experiments

Outlook

- More algorithms
- Refine hyperparameters with more experiments
- Update function for the models

Outlook

- More algorithms
- Refine hyperparameters with more experiments
- Update function for the models
- Incorporate user feedback about the library

Conclusions

- Python for easy readable source code
- Simple to use
- State of the art recommender algorithms
- Open source license: GPLv3

Conclusions

- Python for easy readable source code ✓
- Simple to use
- State of the art recommender algorithms
- Open source license: GPLv3

Conclusions

- Python for easy readable source code ✓
- Simple to use ✓
- State of the art recommender algorithms
- Open source license: GPLv3

Conclusions

- Python for easy readable source code ✓
- Simple to use ✓
- State of the art recommender algorithms ✓
- Open source license: GPLv3

Conclusions

- Python for easy readable source code ✓
- Simple to use ✓
- State of the art recommender algorithms ✓
- Open source license: GPLv3 ✓

Have Fun!

github.com/Foolius/recsyslab

github.com/Foolius/recsyslab/archive/master.zip

```
$ git clone
    https://github.com/Foolius/recsyslab.git
```

(2013).
The leave-one-out protocol.
http://en.wikipedia.org/wiki/Cross-validation_%28statistics%29#Leave-one-out_cross-validation.

(2013).
Matrix factorization.
http://en.wikipedia.org/wiki/Matrix_factorization.

Diaz-Aviles, E., Drumond, L., Gantner, Z., Schmidt-Thieme, L., and Nejdl, W. (2012).
What is happening right now... that interests me?: online topic discovery and recommendation in twitter.
In *Proceedings of the 21st ACM international conference on Information and knowledge management*, pages 1592–1596. ACM.

- Jahrer, M. and Töscher, A. (2011).
  Collaborative filtering ensemble for ranking.
  In *Proc. of KDD Cup Workshop at 17th ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining, KDD*, volume 11.

- Karypis, G. (2001).
  Evaluation of item-based top-n recommendation algorithms.
  In *Proceedings of the tenth international conference on Information and knowledge management*, CIKM '01, pages 247–254, New York, NY, USA. ACM.

- Lemire, D. and Maclachlan, A. (2007).
  Slope one predictors for online rating-based collaborative filtering.
  CoRR, abs/cs/0702144.

Ning, X. and Karypis, G. (2011).
Slim: Sparse linear methods for top-n recommender systems.
In Cook, D. J., Pei, J., Wang, W., Zaïane, O. R., and Wu, X., editors, *ICDM*, pages 497–506. IEEE.

Rendle, S., Freudenthaler, C., Gantner, Z., and Schmidt-Thieme, L. (2009).
Bpr: Bayesian personalized ranking from implicit feedback.
In *Proceedings of the Twenty-Fifth Conference on Uncertainty in Artificial Intelligence*, UAI '09, pages 452–461, Arlington, Virginia, United States. AUAI Press.

Sarwar, B. M., Karypis, G., Konstan, J. A., and Riedl, J. T. (2000).
Application of dimensionality reduction in recommender system – a case study.

In *IN ACM WEBKDD WORKSHOP*.