Homework 3: Representation Learning & Recommender Systems

Mobile Data Mining

Spring 2019

Goal

- **Representation Learning**
  - Use representation learning techniques in mobile big data mining
    - Learning the embeddings of apps based on the app usage traces
  - Familiar with some simple classification algorithms
    - Classify apps into different categories using the obtained embeddings as the feature
- **Recommender Systems**
  - Use collaborative filtering to recommend apps to users
    - User-user collaborative filtering
    - Item-item collaborative filtering
    - Matrix-factorization (MF)-based collaborative filtering

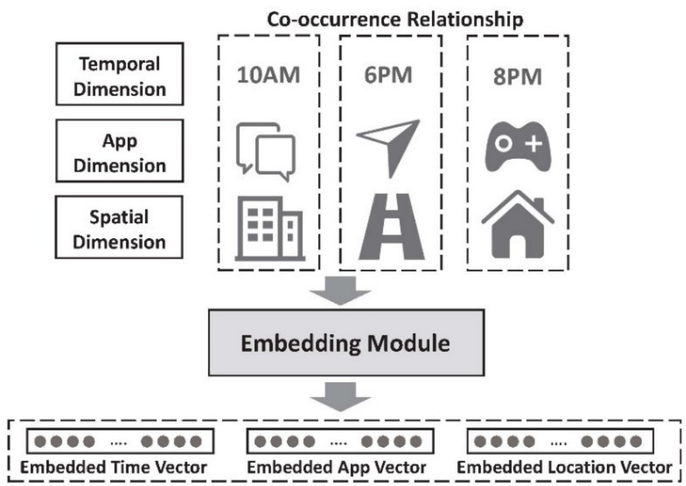
Data

- code/input/AppUsageTrace.txt
  - App usage traces in Shanghai
    - Involve over 2,000 apps and 10,875 locations (cellular base stations)
    - Duration about one week
  - Format (each line)
    - User ID||Location ID||Time (in hour)||ID of Used App

- code/App2Category.txt
  - Category of each app
  - Format (each line)
    - App ID||Category ID

- code/Categories.txt
  - English name of each category
  - Format (each line)
    - Category ID||English Name

Representation Learning: Reconstruction embedding



- Utilize the co-occurrence relationship between units (include time-bins, locations, and apps) to learning their embeddings
- Optimization Target:

损失函数

$$\mathcal{O} = -\sum_{r \in R} \sum_{e \in r} \log P(e|r_{-e})$$

  - R is the set of all app usage records. Each record contains 3 units, i.e., the time-bin, the base station, and the used app. In addition, $r_{-e} = \{o \neq e | o \in r\}$.

Representation Learning: Reconstruction embedding

- We model the likelihood of observing each unit e in each record r given its context r_{-e} by
$$P(e|r_{-e}) = \exp\big(s(e, r_{-e})\big) \,/ \sum_{o \in X} \exp\big(s(o, r_{-e})\big)$$
    - X represents all with the same type of e. For example, for an app e, X is the set of all apps.

- $s()$ is a score function reflecting the similarity between the unit e and its context r_{-e}, defined by:
$$s(e, r_{-e}) = < I_e, h_e >$$
$$h_e = \frac{1}{|r_{-e}|} \sum_{o \in r_{-e}} I_o$$
    - where $I_e \in \mathcal{R}^D$ is the D-dimensional embedding of unit e.

Representation Learning: Reconstruction embedding

Negative Sampling:

- Calculating $P(e|r_{-e})$ requires the summation over the entire set of units X, which leads to high computational complexity

- We approximate $-\log P(e|r_{-e})$ with

$$J_r = -\log \sigma\big(s(e, r_{-e})\big) - \sum_{k=1}^{K} \log \sigma(-s(o_k, r_{-e}))$$

  - where $o_1 \ldots o_K$ are the K negative samples randomly sampled from X
  - $\sigma(x) = \frac{\exp(x)}{1+\exp(x)}$ is the sigmoid function.

Representation Learning: Reconstruction embedding

Stochastic gradient descent:

- Iteratively randomly sample a record r and a unit e. Then change variables in the direction of the gradient of J_r

- For example, we have
$$\frac{\partial J_r}{\partial I_e} = \left(\sigma\left(s(e, r_{-e})\right) - 1\right)h_e$$
- Thus, each time we will update I_e with
$$I_e \leftarrow I_e - \alpha \frac{\partial J_r}{\partial I_e}$$
  - where α is the learning rate.

Recommender Systems: Preprocessing

- Construct the user-app matrix $R = \{r_{ul}\}_{U \times L}$
  - U is the set of all users, and L is the set of all apps.
  - r_{ul} represents whether user u has used app l.

Recommender Systems: User-User Collaborative Filtering

- Predict r_{ul} by

$$r_{ul} = \frac{\sum_{v \in N(u)} S_{uv}\, r_{vl}}{\sum_{y \in N(u)} S_{uv}}$$

- where $S_{uv} = sim(u, v)$, $N(u)$ is the set of k users most similar to u.

- Cosine similarity measure

$$sim(u, v) = \cos(\boldsymbol{r}_u, \boldsymbol{r}_v) = \frac{\boldsymbol{r}_u \cdot \boldsymbol{r}_v}{||\boldsymbol{r}_u|| \cdot ||\boldsymbol{r}_v||}$$

- where $\boldsymbol{r}_u$ is the uth row in R.

Recommender Systems: Item-Item Collaborative Filtering

- Predict r_{ul} by

$$r_{ul} = \frac{\sum_{j \in N(l)} S_{lj} r_{uj}}{\sum_{j \in N(l)} S_{lj}}$$

- where $S_{lj} = sim(l, j)$, $N(l)$ is the set of k apps most similar to l.

- Cosine similarity measure

$$sim(l, j) = \cos(\boldsymbol{r}_l, \boldsymbol{r}_j)$$

- where $\boldsymbol{r}_l$ is the lth column in R.

Recommender Systems: Matrix-Factorization (MF)-based Collaborative Filtering

- Predict r_{ul} by

$$r_{ul} = q_u \cdot p_l$$

- where q_u and p_l are k-dimensional latent vectors for user u and app l, respectively

- q_u and p_l are learnt from solving the optimization problem

$$\min_{\{P,Q\}} \sum_{\text{Training set}} (r_{ul} - q_u \cdot p_l)^2 + \lambda_1 \sum_u ||p_u||^2 + \lambda_2 \sum_l ||\mathrm{q}_l||^2$$

- which can be solved by stochastic gradient descent (SGD).
- For detailed algorithm, refer to page 79 of the slides of recommender systems in course documents.

Experiments - 1

- Embedding
  - A line in “embed.py” is missing. Please complete it.
  - Use the code to learn the embedding of apps
  - Run: “python train.py”
  - Output: code/output/embeddings/
- Virtualization
  - Use the code “DimReduce.py” to map all the obtained embedding into 2-dimensional space for virtualization
  - Plot the embeddings of different time-bins (hours)
  - Plot the embeddings of apps of the category “Video”, “Finance”, “Music”
  - Bonus: Tune the parameter ‘negative‘ and ‘dim’, try to find some interesting observations.
- Classification
  - Focus on apps of “video” and “Finance”
  - Use SVM or Decision Trees to divide them based on their embeddings
    - You can directly call existing functions of these algorithms
    - Use 70% as the training set and 30% apps the test set
  - Calculate the obtained accuracy

Experiments - 2

- Randomly remove 20% user-app pair in the dataset, and use collaborative filtering to predict them
  - prediction whether corresponding users have used corresponding apps
  - Calculate the RMSE
    - $RMSE = \sqrt{\frac{1}{n}\sum(r_{ul} - \hat{r}_{ui})^2}$
    - $\hat{r}_{ui}$ is the predicted value, n is the number of predicted user-app pair.
  - Implement user-user, item-item, MF-based collaborative filtering.
  - Investigate the influence of k. Plot the RMSE-k curve.

Submission

- Submit this homework before June 16th. (Hard Deadline, please keep in mind)

- Submit as .zip file, including:
  - 1) A word document, Including:
    - Brief summary about the algorithms
    - All the results you obtained, presented in table or figure (using figure more, and show the results clearly and beautifully)
    - Interpretation/discussion for each result
    - Do not need to copy the code into this document
  - 2) Source code, Including:
    - Data processing code
    - Collaborative filtering code
    - Other analysis code

Thank you!