Part 3: k-nearest neighbours

Robert Kwieciński

OLX Group and Adam Mickiewicz University

May 22, 2020

k-nearest neighbours algorithm

k-nearest neighbours algorithm

In a basic version:

- represent all your training samples in n-dimensional space,
- define a distance function between points (for example Euclidean distance),
- choose $k \in \mathbb{N}$,
- for a given observation in the test set find k observations (neighbours) from the train set which distance from your observation is the smallest,
- predict value/class of your observation based on values/classes of your neighbours.

The algorithm is widely used for classification and regression purposes. In the simplest form there is no parameters and only one hiperparameter - number of neighbours.

KNN in recommender systems

User-based KNN in recommender systems

Suppose we have m users and n items and we want to predict a rating r_{ui}.

- each **user** is represented by his ratings,
- we define a **similarity measure** $\mathrm{sim}(u, v)$ **between users** - it is common to take cosine similarity or Pearson coefficient of the vectors ratings of items rated by both users,
- we choose $k \in \mathbb{N}$,
- we are looking for a set $N_i^k(u)$ of k **the most similar users** to the user u who rated movie i
- the final prediction is:

$$\hat{r}_{ui} = \frac{\sum_{v \in N_i^k(u)} \mathrm{sim}(u, v) r_{vi}}{\sum_{v \in N_i^k(u)} \mathrm{sim}(u, v)}.$$

KNN in recommender systems

Item-based KNN in recommender systems

Suppose we have m users and n items and we want to predict a rating r_{ui}.

- each **item** is represented by received ratings,
- we define a **similarity measure** $\mathrm{sim}(i, j)$ **between items** - it is common to take cosine similarity or Pearson coefficient of the vectors ratings of items restricted to users who rated both items,
- we choose $k \in \mathbb{N}$,
- we are looking for a set $N_u^k(i)$ of k **the most similar items** to the item i which were rated by user u,
- the final prediction is:

$$\hat{r}_{ui} = \frac{\sum_{j \in N_u^k(i)} \mathrm{sim}(i, j) r_{uj}}{\sum_{j \in N_u^k(i)} \mathrm{sim}(i, j)}.$$

Difference between user-based KNN and item-based KNN

User/Item	i_1	i_2	i_3	i_4	i_5
u_1	1	0	1	1	0
u_2	0	1		1	0
u_3		1	0	1	0
u_4		1	1	1	1
u_5	1	0	1	?	?

Figure: User-item rating matrix

User-based approach:

$$s(u_5, u_1) = \cos([1,0,1],[1,0,1]) = 1$$
$$s(u_5, u_2) = \cos([0,1],[1,0]) = 0$$
$$s(u_5, u_3) = 0$$
$$s(u_5, u_4) = \frac{1}{\sqrt{2}}$$
$$\hat{r}_{u_5,i_4} = r_{u_1,i_4} = 1$$
$$\hat{r}_{u_5,i_5} = r_{u_1,i_5} = 0$$

Item-based approach:

$$s(i_4, i_1) = \cos([1,1],[0,1]) = \frac{1}{\sqrt{2}} \approx 0.71$$
$$s(i_4, i_2) = \frac{3}{2\sqrt{3}} \approx 0.87$$
$$s(i_4, i_3) = \frac{2}{\sqrt{6}} \approx 0.82$$
$$\ldots$$
$$\hat{r}_{u_5,i_4} = r_{u_5,i_2} = 0$$
$$\hat{r}_{u_5,i_5} = r_{u_5,i_3} = 1$$

Some KNN problems

KNN problem - lack of similar users/items

In KNN models it is possible that none of similar users has rated target film and predicted rating is computed based on not very similar users.

Some KNN problems

KNN problem - lack of similar users/items

In KNN models it is possible that none of similar users has rated target film and predicted rating is computed based on not very similar users.

KNN problem - memory demanding

In practice number of users and items might be huge (several millions) and preserving similarity matrix in memory is expensive. Unfortunately preserving only top k similar users/items for each item is not enough (check carefully formulas to answer why).

Some KNN problems

KNN problem - lack of similar users/items

In KNN models it is possible that none of similar users has rated target film and predicted rating is computed based on not very similar users.

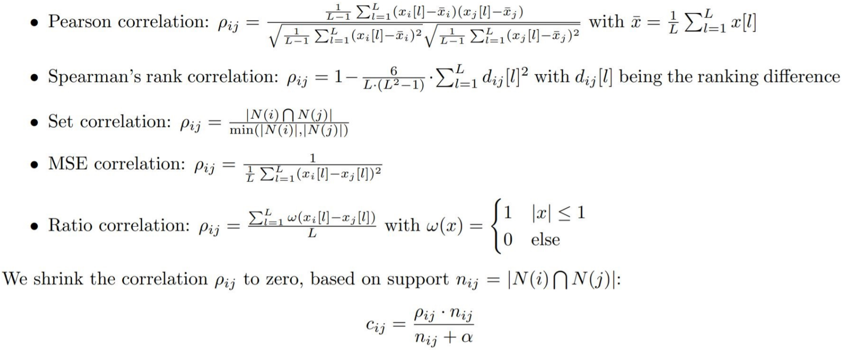
KNN problem - memory demanding

In practice number of users and items might be huge (several millions) and preserving similarity matrix in memory is expensive. Unfortunately preserving only top *k* similar users/items for each item is not enough (check carefully formulas to answer why).

Solution

Preserve only the most similar users. If they have not rated target movie, use the prediction of their ratings instead. In [1] MF model was used for predictions.

Similarity measures



- Pearson correlation: $\rho_{ij} = \frac{\frac{1}{L-1}\sum_{l=1}^{L}(x_i[l]-\bar{x}_i)(x_j[l]-\bar{x}_j)}{\sqrt{\frac{1}{L-1}\sum_{l=1}^{L}(x_i[l]-\bar{x}_i)^2}\sqrt{\frac{1}{L-1}\sum_{l=1}^{L}(x_j[l]-\bar{x}_j)^2}}$ with $\bar{x} = \frac{1}{L}\sum_{l=1}^{L} x[l]$
- Spearman's rank correlation: $\rho_{ij} = 1 - \frac{6}{L\cdot(L^2-1)}\cdot\sum_{l=1}^{L} d_{ij}[l]^2$ with $d_{ij}[l]$ being the ranking difference
- Set correlation: $\rho_{ij} = \frac{|N(i)\bigcap N(j)|}{\min(|N(i)|,|N(j)|)}$
- MSE correlation: $\rho_{ij} = \frac{1}{\frac{1}{L}\sum_{l=1}^{L}(x_i[l]-x_j[l])^2}$
- Ratio correlation: $\rho_{ij} = \frac{\sum_{l=1}^{L}\omega(x_i[l]-x_j[l])}{L}$ with $\omega(x) = \begin{cases} 1 & |x| \le 1 \\ 0 & \text{else} \end{cases}$

We shrink the correlation ρ_{ij} to zero, based on support $n_{ij} = |N(i)\bigcap N(j)|$:

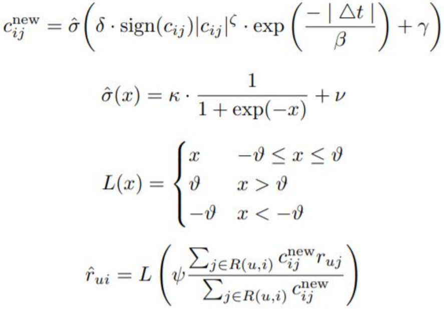
$$c_{ij} = \frac{\rho_{ij}\cdot n_{ij}}{n_{ij}+\alpha}$$

Similarity measures used by the winners of Netflix Prize [2]

KNNMovieV3-2

Models used by the winners of the Netflix Prize competition

Here is an example of k-NN model used in 33 predictors (25 predictors with simpler approach ($\zeta = \kappa = \psi = 1, \nu = 0, \vartheta = \infty$)).



$$c_{ij}^{\text{new}} = \hat{\sigma}\left(\delta \cdot \text{sign}(c_{ij})|c_{ij}|^{\zeta} \cdot \exp\left(\frac{-\mid \Delta t \mid}{\beta}\right) + \gamma\right)$$

$$\hat{\sigma}(x) = \kappa \cdot \frac{1}{1 + \exp(-x)} + \nu$$

$$L(x) = \begin{cases} x & -\vartheta \leq x \leq \vartheta \\ \vartheta & x > \vartheta \\ -\vartheta & x < -\vartheta \end{cases}$$

$$\hat{r}_{ui} = L\left(\psi \frac{\sum_{j \in R(u,i)} c_{ij}^{\text{new}} r_{uj}}{\sum_{j \in R(u,i)} c_{ij}^{\text{new}}}\right)$$

KNN models used by the winners of Netflix Prize [2]

Python implementation

To do (especially for absent students):

- Go through - *P3. k-nearest neighbours* notebook to:
  - check simplified version of I-KNN (where we sum over all neighbours instead of top k)
  - observe evaluation measures
  - run ready-made KNN algorithm implemented in Surprise
  - read Surprise docs about KNN algorithms here, it is described really clear
  - **project task 4: use a version of your choice of Surprise KNN algorithm**

References I

[1] A. Töscher, M. Jahrer, and R. Legenstein, “Improved neighborhood-based algorithms for large-scale recommender systems,”, Jan. 2008. DOI: 10.1145/1722149.1722153.

[2] A. Töscher and M. Jahrer, “The bigchaos solution to the netflix grand prize,”, Sep. 2009, http://https://www.netflixprize.com/assets/GrandPrize2009_BPC_BigChaos.pdf/.