CONTENT-BASED RS

CONTENT-BASED RS

- Idea :
  - Base on description, content of each item and profile of user's preferences to recommend other item to the user.
  - Item is represented as a vector $x = (x_1, x_2, ...x_n)$, each feature describes a properties of the item.
  - The level of the user's concern about an item is described as a function $y = f(x)$.

CONTENT-BASED RS

- Idea :
  - Base on description, content of each item and profile of user's preferences to recommend other item to the user.
  - Item is represented as a vector $x = (x_1, x_2, ...x_n)$, each feature describes a properties of the item.
  - The level of the user's concern about an item is described as a function $y = f(x)$.
- Pros
  - CBRS does not need data about other users. This makes it easier to scale to a large number of users.
  - The model can record the specific interests of a user, and recommend items that very few users are interested in.

CONTENT-BASED RS

- Idea :
  - Base on description, content of each item and profile of user's preferences to recommend other item to the user.
  - Item is represented as a vector $x = (x_1, x_2, ...x_n)$, each feature describes a properties of the item.
  - The level of the user's concern about an item is described as a function $y = f(x)$.
- Pros
  - CBRS does not need data about other users. This makes it easier to scale to a large number of users.
  - The model can record the specific interests of a user, and recommend items that very few users are interested in.
- Cons
  - The model can only make recommendations based on existing interests of the user.
  - The model does not use the preferences of other users.

How to encode data?

How to encode data?

- One-hot encoding
- Word embeddings
- Term frequency - inverse document frequency (TF-IDF) encoding

TF - IDF representation :

TF - IDF representation :

- TF - IDF is used to weigh a keyword in any documents and assign the importance to that keyword
- The higher the TF-IDF score, the more important the term/keyword

TF - IDF representation :

- TF - IDF is used to weigh a keyword in any documents and assign the importance to that keyword
- The higher the TF-IDF score, the more important the term/keyword
- TF(term frequency) :
  - binary : $tf = 0, 1$
  - raw count : $tf = f_{t,d}$
  - term frequency : $tf = \dfrac{f_{t,d}}{\sum_{t' \in d} f_{t',d}}$
  - log normalization : $tf = \log(1 + f_{t,d})$
  - double normalization k : $tf = k + (1-k)\dfrac{f_{t,d}}{\max_{t' \in d} f_{t',d}}$

TF - IDF representation :

- IDF(Inverse document frequency)

TF - IDF representation :

- IDF(Inverse document frequency)
  - Unary : $idf = 1$
  - Inverse document frequency : $idf = \log \dfrac{N}{n_t}$
  - Inverse document frequency smooth: $idf = \log \dfrac{N}{1 + n_t}$
  - Inverse document frequency max: $idf = \log \left(\dfrac{\max_{t' \in d} n_{t'}}{1 + n_t} \right)$
  - Probabilistic inverse document frequency: $idf = \log \dfrac{N - n_t}{n_t}$

TF - IDF representation :

- IDF(Inverse document frequency)
  - Unary : $idf = 1$
  - Inverse document frequency : $idf = \log \dfrac{N}{n_t}$
  - Inverse document frequency smooth: $idf = \log \dfrac{N}{1 + n_t}$
  - Inverse document frequency max: $idf = \log \left(\dfrac{\max_{t' \in d} n_{t'}}{1 + n_t} \right)$
  - Probabilistic inverse document frequency: $idf = \log \dfrac{N - n_t}{n_t}$
- The product $tf * idf$ represents the TF-IDF score.

TF - IDF representation :

Loss function for Linear Regression

TF - IDF representation :

Loss function for Linear Regression

- Linear regression : $y_{mn} = x_m w_n + b_n$

TF - IDF representation :

Loss function for Linear Regression

- Linear regression : $y_{mn} = x_m w_n + b_n$
- Loss function :

$$\mathbb{L} = \frac{1}{2} \sum_{m:r_{mn}=1} (x_m w_n + b_n - y_{mn})^2 + \frac{\lambda}{2} \|w_n\|_2^2$$

TF - IDF representation :

Loss function for Linear Regression

- Linear regression : $y_{mn} = x_m w_n + b_n$
- Loss function :

$$\mathbb{L} = \frac{1}{2} \sum_{m:r_{mn}=1} (x_m w_n + b_n - y_{mn})^2 + \frac{\lambda}{2} \|w_n\|_2^2$$

- Take the average :

$$\mathbb{L} = \frac{1}{2s_n} \sum_{m:r_{mn}=1} (x_m w_n + b_n - y_{mn})^2 + \frac{\lambda}{2s_n} \|w_n\|_2^2$$

NEIGHBORHOOD-BASED COLLABORATIVE FILTERING

NEIGHBORHOOD-BASED COLLABORATIVE FILTERING

- Idea
  - Make prediction about the interests of a user by collecting preferences from many users.
  - Each user (or item) is represented as a vector $x = (x_1, x_2, ..., x_n)$, each feature describes a level of user's concern to the item.
  - The level of a user's concern to a item can be predicted by calculating the vector similarity between the given user and other.

NEIGHBORHOOD-BASED COLLABORATIVE FILTERING

- Idea
  - Make prediction about the interests of a user by collecting preferences from many users.
  - Each user (or item) is represented as a vector $x = (x_1, x_2, ..., x_n)$, each feature describes a level of user's concern to the item.
  - The level of a user's concern to a item can be predicted by calculating the vector similarity between the given user and other.
- Pros
  - Better RMSE (more exactly) than content-based RS
  - Use preferences of other users/items to make prediction.

NEIGHBORHOOD-BASED COLLABORATIVE FILTERING

- Idea
  - Make prediction about the interests of a user by collecting preferences from many users.
  - Each user (or item) is represented as a vector $x = (x_1, x_2, ..., x_n)$, each feature describes a level of user's concern to the item.
  - The level of a user's concern to a item can be predicted by calculating the vector similarity between the given user and other.
- Pros
  - Better RMSE (more exactly) than content-based RS
  - Use preferences of other users/items to make prediction.
- Cons
  - Hard to scale a large number of users/items.

NEIGHBORHOOD-BASED COLLABORATIVE FILTERING

How to know two users are similar?

NEIGHBORHOOD-BASED COLLABORATIVE FILTERING

How to know two users are similar?

- Cosine Similarty

NEIGHBORHOOD-BASED COLLABORATIVE FILTERING

How to know two users are similar?

- Cosine Similarty
- Person corelation

NEIGHBORHOOD-BASED COLLABORATIVE FILTERING

How to know two users are similar?

- Cosine Similarty
- Person corelation
- Jaccard similarty : $similarity = \dfrac{|A \cap B|}{|A \cup B|}$

NEIGHBORHOOD-BASED COLLABORATIVE FILTERING

How to know two users are similar?

- Cosine Similarty
- Person corelation
- Jaccard similarty : $similarity = \dfrac{|A \cap B|}{|A \cup B|}$
- Mean Measure of Divergence

NEIGHBORHOOD-BASED COLLABORATIVE FILTERING

Cosine Similarity

NEIGHBORHOOD-BASED COLLABORATIVE FILTERING

Cosine Similarity

- Cosine similarity is used to calculate the similarity between two vectors.

NEIGHBORHOOD-BASED COLLABORATIVE FILTERING

Cosine Similarity

- Cosine similarity is used to calculate the similarity between two vectors.
- Formula :

$$\text{cosine\_similarity}(u_1, u_2) = \cos(u_1, u_2) = \frac{u_1^T u_2}{\|u_1\|_2 . \|u_2\|_2}$$

NEIGHBORHOOD-BASED COLLABORATIVE FILTERING

Cosine Similarity

- Cosine similarity is used to calculate the similarity between two vectors.
- Formula :

$$\text{cosine\_similarity}(u_1, u_2) = \cos(u_1, u_2) = \frac{u_1^T u_2}{\|u_1\|_2 . \|u_2\|_2}$$

- Predict rating of user u to item i :

$$\hat{y}_{i,u} = \frac{\sum_{u_j \in N(u,j)} \bar{y}_{i,u_j} \text{sim}(u, u_j)}{\sum_{u_j \in N(u,j)} |\text{sim}(u, u_j)|}$$

NEIGHBORHOOD-BASED COLLABORATIVE FILTERING

Cosine Similarity

- Cosine similarity is used to calculate the similarity between two vectors.
- Formula :

$$\text{cosine\_similarity}(u_1, u_2) = \cos(u_1, u_2) = \frac{u_1^T u_2}{\|u_1\|_2 \cdot \|u_2\|_2}$$

- Predict rating of user u to item i :

$$\hat{y}_{i,u} = \frac{\sum_{u_j \in N(u,j)} \bar{y}_{i,u_j} \text{sim}(u, u_j)}{\sum_{u_j \in N(u,j)} |\text{sim}(u, u_j)|}$$

- Loss function :

$$\mathbb{L} = \|y - \hat{y}\|_2^2$$

MATRIX FACTORIZATION COLLABORATIVE FILTERING

MATRIX FACTORIZATION COLLABORATIVE FILTERING

- Idea :
  - Approximating the user-item reaction matrix into the product of two lower dimensionality matrix $Y = XW$.
  - By evaluating the product of two matrix, we get a complete matrix of user-item reaction

MATRIX FACTORIZATION COLLABORATIVE FILTERING

- Idea :
  - Approximating the user-item reaction matrix into the product of two lower dimensionality matrix $Y = XW$.
  - By evaluating the product of two matrix, we get a complete matrix of user-item reaction
- Pros :
  - Able to discover some new data, based on latent feature between two matrices.
  - Simple inference by evaluating the matrix product.
  - Save memory.

MATRIX FACTORIZATION COLLABORATIVE FILTERING

- Idea :
  - Approximating the user-item reaction matrix into the product of two lower dimensionality matrix $Y = XW$.
  - By evaluating the product of two matrix, we get a complete matrix of user-item reaction
- Pros :
  - Able to discover some new data, based on latent feature between two matrices.
  - Simple inference by evaluating the matrix product.
  - Save memory.
- Cons :
  - Take much time to train the model.

MATRIX FACTORIZATION COLLABORATIVE FILTERING

How to train the model?

MATRIX FACTORIZATION COLLABORATIVE FILTERING

How to train the model?

- Gradient descent

MATRIX FACTORIZATION COLLABORATIVE FILTERING

How to train the model?

- Gradient descent
- Alternating least square

MATRIX FACTORIZATION COLLABORATIVE FILTERING

How to train the model?

- Gradient descent
- Alternating least square
- Generalized low rank models

MATRIX FACTORIZATION COLLABORATIVE FILTERING

How to train the model?

- Gradient descent
- Alternating least square
- Generalized low rank models
- Singular value decommposition (SVD)

MATRIX FACTORIZATION COLLABORATIVE FILTERING

Gradient descent to optimize the linear-regression loss function

MATRIX FACTORIZATION COLLABORATIVE FILTERING

Gradient descent to optimize the linear-regression loss function

- Loss function :

$$\mathbb{L}(X, W) = \frac{1}{2s} \sum_{n=1}^{N} \sum_{m:r_{mn}=1} (y_{mn} - x_m w_n)^2 + \frac{\lambda}{2} \left(\|X\|_F^2 + \|W\|_F^2 \right)$$

MATRIX FACTORIZATION COLLABORATIVE FILTERING

Gradient descent to optimize the linear-regression loss function

- Loss function :

$$\mathbb{L}(X, W) = \frac{1}{2s} \sum_{n=1}^{N} \sum_{m:r_{mn}=1} (y_{mn} - x_m w_n)^2 + \frac{\lambda}{2} \left(\|X\|_F^2 + \|W\|_F^2 \right)$$

- Update W :

$$w_n = w_n - \eta \left(-\frac{1}{s} \hat{X}_n^T (\hat{y}^n - \hat{X}_n w_m) + \lambda w_n \right)$$

MATRIX FACTORIZATION COLLABORATIVE FILTERING

Gradient descent to optimize the linear-regression loss function

- Loss function :

$$\mathbb{L}(X, W) = \frac{1}{2s} \sum_{n=1}^{N} \sum_{m:r_{mn}=1} (y_{mn} - x_m w_n)^2 + \frac{\lambda}{2} \left(\|X\|_F^2 + \|W\|_F^2 \right)$$

- Update W :

$$w_n = w_n - \eta \left(-\frac{1}{s} \hat{X}_n^T (\hat{y}^n - \hat{X}_n w_m) + \lambda w_n \right)$$

- Update X :

$$x_m = x_m - \eta \left(-\frac{1}{s} (\hat{y}^m - x_m \hat{W}_m) \hat{W}_m^T + \lambda x_m \right)$$