RECSYS Formula Sheet

Global Effects

1. Global Bias μ — $\mu = \frac{\sum_u \sum_i r_{ui}}{N+C}$
   $N = \#$non-zero ratings
2. Normalization r'_{ui} — $r'_{ui} = r_{ui} - \mu$
3. Item Shrink Average Rating b_i — $b_i = \frac{\sum_u r'_{ui}}{N_i+C}$
   $N_i =$ number of Users who rated item i
4. Normalization (Again) — $r''_{ui} = r'_{ui} - b_i \ \forall u \in U, i \in I$
5. User Shrinked Average Rating b_u — $b_u = \frac{\sum_{i \in I} r''_{ui}}{N_u}$

Rating Estimation

We can estimate a rating in a NON-PERSONALIZED way using the global effects:

$$r_{ui} = \mu + b_u + b_i$$

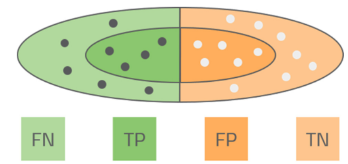
Evaluation Techniques

Online Evaluation	
Direct Feedback	User questionnaires (high bias)
A/B Testing	Compare RS_1 vs RS_2 with unaware users
Controlled Exp.	Small aware group, mock-up testing
Crowdsourcing	Large volunteer group with compensation
Offline Evaluation	
Tasks	Rating Prediction,Top-N
Dataset Split	Training Set $\rightarrow$ Model Creation User Profile $\rightarrow$ Rating Generation Testing Set $\rightarrow$ Evaluation
Dataset Partitioning	
Hold out of Ratings	Random % of ratings for testing, Risk of overfitting
Hold out of Users	Exclude users for training, Split excluded users' ratings between profile/testing

Quality Metrics

Metric	Description
Relevance	Ability of recommending items that the user likes
Diversity	Ability of recommending different items
Serendipity	Ability of recommending unexpected items
Coverage	Ability of recommending items that the user has not seen
Novelty	Ability of recommending unknown items
Consistency	Ability to give consistent recommendations
Confidence	Measure how much the model is sure about its recommendations
Scalability	Time required for training
Serving Time	Time required for serving recommendations
Fairness	Fair recommendation for all users and for content providers

Classification Metrics



Classification Metrics	Formula
Recall	$\frac{TP}{FN+TP}$
Precision	$\frac{TP}{TP+FP}$
Fallout	$\frac{FP}{FP+TN}$

Ranking Metrics	
AUC	$\frac{\sum_k Recall(k)\cdot \Delta Fallout}{N_i}$
AP	$\sum_k Precision(k) \cdot [Recall(k) - Recall(k-1)]$
MAP	$\frac{\sum_u AP_u(k)}{N_u}$

Content-Based Filtering

Similarity Metrics	
Basic Similarity	$s_{ij} = \vec{i} \cdot \vec{j} = \#$common attributes
Cosine Similarity	$s_{ij} = \frac{\vec{i}\cdot\vec{j}}{\lVert\vec{i}\rVert \cdot \lVert\vec{j}\rVert}$
Shrinked Cosine	$s_{ij} = \frac{\vec{i}\cdot\vec{j}}{\lVert\vec{i}\rVert \cdot \lVert\vec{j}\rVert + C}$
Rating Estimation	
Single Rating	$\tilde{r}_{ui} = \frac{\sum_j r_{uj} \cdot s_{ij}}{\sum_j s_{ij}}$
Matrix Form	$\tilde{R} = R \cdot S$
k-Nearest Neighbors (kNN)	
Definition	Keep only k highest similarity values per item
Effect	Reduces noise and improves computation speed
Selection	k too small: unreliable estimates k too large: noisy recommendations
Formula	$\tilde{r}_{ui} = \frac{\sum_{j, i \in N_k(j)} r_{uj} \cdot s_{ji}}{\sum_{j, i \in N_k(j)} s_{ji}}$
TF-IDF Weighting	
Term Frequency	$TF_{i,a} = \frac{N_{i,a}}{N_i}$ $N_{i,a}$: occurrences of attribute a in item i N_i: attributes in item i
Inverse Doc Freq	$IDF_a = \log_2 \frac{N_{items}}{N_a}$ N_{items}: total items N_a: items with attribute a

Collaborative Filtering

User-based CF aims to find similar users and recommend items based on their preferences.

Similarity	Formula	Context
Cosine Similarity	$s_{ij} = \frac{\vec{i} \cdot \vec{j}}{\lVert\vec{i}\rVert \cdot \lVert\vec{j}\rVert}$	Implicit ratings
Jaccard Similarity	$s_{ij} = \frac{\vec{i} \cap \vec{j}}{\vec{i} \cup \vec{j}}$	Implicit ratings
Pearson Correlation	$s_{ij} = \frac{\sum_{i \in I}(r_{iu}-\bar{r}_u)\cdot(r_{iv}-\bar{r}_v)}{\sqrt{\sum_{i \in I}(r_{iu}-\bar{r}_u)^2}\sqrt{\sum_{i \in I}(r_{iv}-\bar{r}_v)^2}}$	Explicit ratings

Focus on Pearson Correlation

Pearson correlation computes similarity between a rating delta. Therefore the similarity is used to predict the delta of the rating!

$$\tilde{r}_{ui} - \bar{r}_u = \frac{\sum v \in KNN(u)(r_{vi}-\bar{r}_v)\cdot s_{uv}}{\sum v \in KNN(u) s_{uv}}$$

The Top-N Scenario

Normalizing the predicted rating is necessary to improve rating prediction. In a Top-N scenario, we can compute the rating $\tilde{r}_{ui}$ without normalization to save computation!

Item-based CF aims to find similarity between items based how many users have the same opinion about them. The similarity is obtained in the same way as for user-based CF, considering the items instead of the users.

Memory-Based vs Model-Based

Memory-Based:

- Requires user profile in URM used to build model
- Only works for "known" users
- Must rebuild model for new users
- Example: User-Based CF (uses user neighborhood)

Model-Based:

- Works with any user profile
- Supports both "known" and "unknown" users
- No model recomputation needed for new users
- Example: Item-Based CF (uses item similarities)

Association Rules

Association rules explore relationships between items using conditional probability:

$$P(i|j) = \frac{\text{\# appearances of i and j}}{\text{\# appearances of j} + C}$$

where C is a shrinkage term to avoid biases. The similarity is asymmetric: $P(i|j) \neq P(j|i)$

Machine Learning Item-based CF

Loss Functions	
Error Metrics	MAE,MSE
Accuracy Metrics	Precision, Recall
Ranking Metrics	AUC, MAP
SLIM (opt. Error Metric)	
Closed-Form Solution Objective	$S^* = \arg\min_S \lVert R - RS \rVert_2$
Constraints on S	$\text{diag}(S) = 0$
Lasso Regression Regularization	$S^* = \arg\min_S \left(\lVert R - RS \rVert_2 + \lambda \lVert S \rVert_1\right)$
Ridge Regression Regularization	$S^* = \arg\min_S \left(\lVert R - RS \rVert_2 + \lambda \lVert S \rVert_2\right)$
Elastic Net Regularization	$S^* = \arg\min_S \left(\lVert R - RS \rVert_2 + \lambda_1 \lVert S \rVert_1 + \lambda_2 \lVert S \rVert_2\right)$
BPR (opt. Ranking)	
(BPR) Probability Function	$P(\tilde{r}_{ui} > \tilde{r}_{uj} \mid \text{user } u) = \sigma(x) = \frac{1}{1+e^{-x}}$
Pairwise Difference for BPR	$x_{uij} = \tilde{r}_{ui} - \tilde{r}_{uj}$
Loss Function	$\arg\max_\theta \prod_{(u,i,j)} P(x_{uij}(\theta) > 0 \mid u)$
Log-Likelihood:	$\arg\max_\theta \sum_{(u,i,j)} \log(P(x_{uij}(\theta) > 0 \mid u))$
Loss Minimization:	$\arg\min_\theta - \sum_{(u,i,j)} \log(\sigma(x_{uij}(\theta))) + \lambda \lvert\lvert\theta\rvert\rvert_2 \ldots$
LightGCN	

BPR optimization

It can be demonstrated that optimizing the BPR objective function is equivalent to maximizing the AUC metric. Thus, BPR is an optimization method for ranking metrics.

$$\begin{aligned} P(\tilde{r}_{ui} > \tilde{r}_{uj} \mid \text{user } u) &= P(\tilde{r}_{ui} - \tilde{r}_{uj} > 0 \mid \text{user } u) &(1)\\ &= P(x_{uij} > 0 \mid \text{user } u) &(2)\\ &= \sigma(x_{uij}) = \frac{1}{1+e^{-x_{uij}}} &(3) \end{aligned}$$

Where $\sigma(x)$ is the sigmoid function to optimize.

- x_{uij} should tend to 1 if i is a relevant item for user u and j is not
- x_{uij} should tend to 0 if both items are not relevant for user u or either both relevant

NB: BPR suffers from popularity bias.

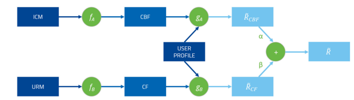
Matrix Factorization

User Rating Matrix (URM)

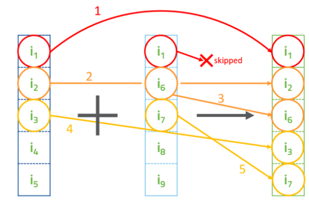
User Preference	x_{uk}: Preference of user u for feature k
Item Description	y_{ik}: Description of item i for feature k
Predicted Rating	$\tilde{r}_{ui} = \sum_k x_{uk} \cdot y_{ik}$
Dimensionality Constraint	$N_k < \frac{N_u \cdot N_i}{N_u + N_i}$
Matrix Factorization	$R \approx X \cdot Y$
Dimensions	$X \in \mathbb{R}^{N_u \times N_f}, Y \in \mathbb{R}^{N_f \times N_i}, R \in \mathbb{R}^{N_u \times N_i}$
Loss Function	$\min_{X,Y} \|R - XY\|_2$
Regularization	$\min_{X,Y} \|R - XY\|_2 + \lambda_1 \|X\|_2 + \lambda_2 \|Y\|_2$
SGD for MF	- Sample (u, i, r_{ui}) - $\frac{\delta E(X,Y)}{\delta x_u} = -2 \cdot (r_{ui} - x_u y_{i*}) \cdot y_{i*} + 2\lambda_1 \cdot x_u$ - $\frac{\delta E(X,Y)}{\delta y_{i*}} = -2 \cdot (r_{ui} - x_u y_{i*}) \cdot x_u + 2\lambda_2 \cdot y_{i*}$
Missing Ratings	◦ MAR: Missing As Random • MAN: Missing As Negative
ALS Algorithm	`While not converged do:` `Fix` X`, Learn` Y `Fix` Y`, Learn` X
◦ FunkSVD Algorithm	`set` $N_k = 0$ `Initialize` X, Y `While not converged do:` `Increment` N_k `Apply ALS for current` N_k
SVD++ (train with SGD)	- $\tilde{r}_{ui} = \mu + b_u + b_i + \sum_k x_{uk} \cdot y_{ki}$ - $\mu^*, b_u^*, b_i^*, X^*, Y^* = \min_{\mu, b_u, b_i, X, Y} E(\dots)$
Asymmetric SVD (m-b)	$\tilde{R} = RZY$, $X = RZ$
• pure SVD (m-b)	$\tilde{R} = U_k \Sigma_k V_k^T = R V_k V_k^T$

Hybrid Recommenders

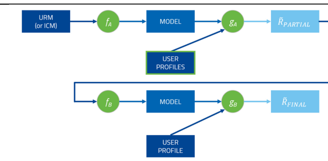
Linear Combination



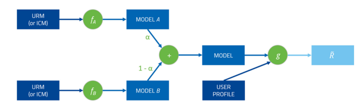
List Combination



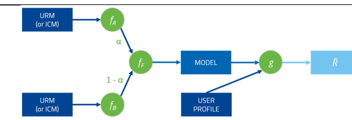
Pipelining



Model Merging



Co-Training



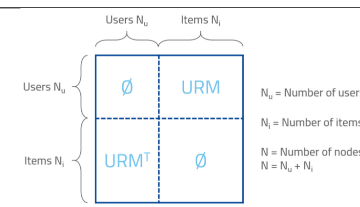
SSLIM (Co-Training Technique)

Optimization	$S^* = \min_S \alpha \|\|R - RS\|\|_F^2 + (1-\alpha)\|\|F - SF\|\|_F^2$
Rating Prediction	$\hat{r}_{ui} = \frac{\sum_j s_{ji} r_{uj}}{\sum_j s_{ji}}$
Feature Prediction	$\hat{f}_{ik} = \sum_j s_{ij} f_{jk}$

Graph-Based Recommenders

Random Walk Probability	$p_{ij} = \frac{g_{ij}}{\sum_j g_{ij}}$ $d_i = \sum_j g_{ij}$
Next Step Probability	$\Pi_i^{k+1} = \sum_j \Pi_j^k \cdot p_{ji}$
Matrix Form	$\Pi^{k+1} = P \cdot \Pi^k$
Steady State	$\Pi = P \cdot \Pi$
Probability Constraint	$\sum \Pi_i = 1$

Matrix G



Graph-Based - 2

PageRank

Random Walk and Restart	$\prod = \gamma \prod \cdot P + (1-\gamma)\prod_0$

P3Alpha $P_3\alpha$

Metapath	$U \rightarrow I \rightarrow U \rightarrow I$
Probability	$P_{UI} = (diag(\frac{1}{d_u}) \cdot R)^\alpha$ $P_{IU} = (diag(\frac{1}{d_i}) \cdot R^T)^\alpha$
Recommendation	$\prod = \gamma \prod \cdot P^3$ $= \gamma \prod \cdot P_{UI} \cdot P_{IU} \cdot P_{UI}$ $= \gamma \prod \cdot P_{UI} \cdot S$
Disadvantages	Strong popularity bias

RP3Beta $RP_3\beta$ (penalize popular items)

Similarity	$S_{ij} = \frac{1}{d_j^\beta} \sum_{u \in U} (\frac{r_{ui} r_{uj}}{d_i d_j})^\alpha$

Cosine Similarity Correlation

As seen above, without the parameter α, the random walk will end up building the same similarity matrix S as the one obtained by the cosine similarity (for **implicit ratings**).

$$S_{ij} = [P_{IU} \cdot P_{UI}]_{ij} = \sum_{u \in U} \frac{r_{ui} r_{uj}}{d_i d_j}$$

DL for RECSYS

Binary Cross Entropy $\arg\min_\theta =$	$-\frac{1}{N} \sum^N [r_{ui} \log(\tilde{p}_{ui}(\theta)) + (1 - r_{ui}) \log(\tilde{p}_{ui}(\theta))]$
Sampling	× Cannot use ground truth (too many negative samples) × Cannot use just positive samples (no learning) ✓ Subsample among + and - interaction with probability $p = 0.5$

Autoencoder

Reconstruction Loss	MSE, BCE
Steps	– Sample a user profile r_u – Ecode it $e_u = g_e(r_u)$ – Decode it $\tilde{r}_u = g_d(e_u)$ – Rank the items

EaseR (Item-Based similarity CF model)

Loss Function $S^* =$	$\arg\min_S \|\|R - RS\|\|_F + \lambda \|\|S\|\|_F + 2\vec{\gamma} \odot diag(S)$ $\vec{\gamma} \in \mathbb{R}^{\|I\|}$
Constraints	$diag(S) = 0$
Similarity Matrix	$P = (R^T \cdot R + \lambda I_{\|I\|})^{-1}$ $S^* = I_{\|I\|} - P \cdot diag(\mathbf{1} \oslash diag(P))$
Pros and Cons	✓ Fast and highly efficient ✓ Due the Frobenius norm, it tries to compute $R = RS$, thus repoducing the input as output such as an autoencoder. × Computing P is memory intensive

Autoencoders and Item-Item Similarity Correlation

Given a shallow autoencoder with no hidden layers and embedding size K, if $f = I$ and $b_e, b_d = 0$ then:

$$e_u = f_e(r_u \cdot W_e + b_e) = r_u \cdot W_e$$
$$\tilde{r}_u = f_d(e_u \cdot W_d + b_d) = e_u \cdot W_d = r_u \cdot W_e \cdot W_d$$

Since $W_e \in \mathbb{R}^{|I| \times K}, W_d \in \mathbb{R}^{|K| \times |I|}$
We can derive the assymmetric (or symmetric, if encoder and decoder share parameters) similarity matrix S as: $S = W_e \cdot W_d$

DL for RECSYS - 2

Denoising Autoecoders

Risks

- × The encoder might create a poor embedding for new user profiles
- × The decoder could lack on reconstruct correctly portion of the embedding space

Denoising Salt & Pepper

- Dropout, remove a number of positive interactions
- Random add a number of positive interactions

Variational Autoencoders (Mult-VAE)

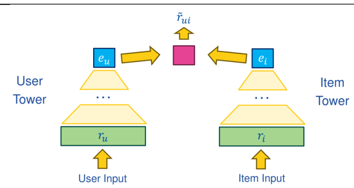
Encoding a input as a distribution

Idea

Encoder: encode the input as $\vec{\mu}, \vec{\sigma}$ of a Gaussian
Decoder: sample from the Gaussian and decode it $\vec{e}\tilde{N}(\vec{\mu}, \vec{\sigma})$
Learn the probability distribution $P(\theta|z)$.

Reparametrization Trick $\vec{e} = \vec{\mu} + \vec{\sigma} \odot \vec{\epsilon}$ Where $\vec{\epsilon} \sim \mathcal{N}(0,1)$

Two Tower Models



Pros and Cons

- ✓ Can use any loss function (it does not have to reconstruct the input)
- ✓ User and item input can be of different types
- × Need to compute several ranking predictions $\tilde{r}_{ui}$
- × If the input is a one-hot encoding, it is **memory based**

Two tower models as Matrix Factorization

Consider a two-tower model with no hidden layers, embedding size K and both user and item input one-hot encoded x_u , x_i. If $f = I$ and $b_u, b_i, I = 0$: $\tilde{r}_{ui} = W_u^U \cdot W_i^I$ Where $W_u^U \in \mathbb{R}^{|U| \times K}, W_i^I \in \mathbb{R}^{|I| \times K}$ Then we have a Matrix Factorization model with K latent factors, $X = W_u^U$ and $Y = W_i^I$

Graph Convolutional Networks

Training

Initialize user and item embeddings $E^{(0)}$

- Sample a data point (depends on loss function)
- Apply h hops of graph convolution on the nodes $E^{(h)}$
- Using $E^{(h)}$ compute the prediction and gradients
- Repeat!

learn is $E^{(0)}$ predict is $E^{(h)}$. The "message" is the node embedding.

LightGCN

- weighted mean as aggregation function
- Loss function is BPR
- Does not include self-connections

$E_u^{(h)} = \sum_{i:u,i \in R^+} \frac{1}{\sqrt{d_u \cdot d_i}} \cdot E_i^{(h-1)}$
$E_i^{(h)} = \sum_{u:u,i \in R^+} \frac{1}{\sqrt{d_u \cdot d_i}} \cdot E_u^{(h-1)}$
normalized adjacency matrix $\hat{G}$:
$\hat{g}_{xy} = \frac{g_{xy}}{\sqrt{d_x d_y}}$

Training

LightGCN

Initialize the embeddings $E^{(0)}$

1. Apply h convolution steps $E^{(h)} = \hat{G}^h \cdot E^{(0)}$
2. Draw a BPR sample u, i, j such that $u, i \in R^+$ and $u, j \in R^-$
3. Compute prediction $\tilde{r}_{ui} = E_u^{(h)} \cdot E_i^{(h)}$ and $\tilde{r}_{uj}$
4. Apply gradient of BPR

Pros and Cons

- ✓ Can work on different types of graphs
- ✓ Can accommodate different aggregationfunctions that may have parameters themselves
- ✓ Can accommodate different lossfunctions
- × Have enormous computational cost
- × Can exhibit high popularity bias

Popularity Bias

$\hat{G}$ with SVD

$$\begin{aligned} E^{(h)} &= \hat{G}^h \cdot E^{(0)} \\ &= (V \cdot \Sigma \cdot V^T)^h \cdot E^{(0)} \\ &= V \cdot \Sigma^h \cdot V^T \cdot E^{(0)} \end{aligned}$$

- Large singular values, strongly popularity-based
- Small singular values, fine-grained signals

Filter Function

f modifies Σ $E^{(h)} = \hat{G}^h \cdot E^{(0)} = V \cdot f(\Sigma)^h \cdot V^T \cdot E^{(0)}$

Graph-Filter Collaborative Filtering

GF-CF

- Observation:small singular values disappear
- Idea: use truncated SVD to compute the similarity.

Normalize: $\tilde{r}_{ui} = \frac{r_{ui}}{\sqrt{d_u d_i}}$

Compute item-item S: $S = \hat{R}^T \cdot \hat{R} + \alpha D_I^{-\frac{1}{2}} \cdot V_k \cdot V_k^T \cdot D_I^{+\frac{1}{2}}$

Pros and Cons

- ✓ Fast computation of the similarity and very effective
- ✓ Flexible, change the exponent of the degree matrix
- × The similarity is dense (high memory requirement, slow prediction)
- × Applying KNN is difficult, large # of neighbors

Factorization Machines

Input Data:

user	item	rating
1, 0, 0	1, 0, 0	3
0, 1, 0	0, 1, 0	1
⋮	⋮	⋮
0, 0, 1	0, 0, 1	2

One hot encoding.

Rating Estimation: $\tilde{r}^{(k)} = \omega_0 + \sum_{i=1}^{n} \omega_i x_i^{(k)} + \sum_{i=1}^{n} \sum_{j=i+1}^{n} \omega_{ij} x_i^{(k)} x_j^{(k)}$

Vector Form: $\tilde{r}^{(k)} = \omega_0 + \vec{\omega} \cdot \vec{x}^{(k)} + \vec{x}^{(k)T} \cdot W \cdot \vec{x}^{(k)}$

Loss Function: $\arg\min_{\vec{\omega}} E(\omega) = \arg\min_{\vec{\omega}} ||r^{(k)} - \tilde{r}^{(k)}||$

W factorization: $\omega_{ij} = \vec{v}_i \cdot \vec{v}_j = \sum_{h=0}^{f} v_{i,h} v_{j,h}$ $f << n$ latent factors

N parameters: $\begin{cases} 1 + n + \frac{n^2 - n}{2} & \text{if not factorized} \\ 1 + n + nf & \text{if factorized} \end{cases}$

Imbalance Problem

- × If ratings are implicit, lead to predict only 1s
- ✓ Random select non rated items for every user
- ✓ Use the same number of positive and negative samples

Factorization Machines as SVD++

Using only collaborative data, the factorization machine is equivalent to SVD++, since one-hot encoding we can rewrite the factorization machine as:

$$\tilde{r}^{(k)} = \omega_0 + \omega_i + \omega_u + V_i \cdot V_u^T \tag{4}$$

Equivalent to SVD++.
We need to add **Context** data, for example from a ICM model.

Etics

Concentration

Concentration Effect: The recommender made popular items more popular.

Gini Coefficient: $G(y) = \frac{\sum_{i=1}^{N} \sum_{j=1}^{N} |y_i - y_j|}{2N^2 \bar{y}}$

$\begin{cases} G(y) \approx 0 \implies \text{even distribution} \\ G(y) \approx 1 \implies \text{concentration} \end{cases}$

In our case: $\begin{cases} G_R(y) < G_I(y) \implies \text{dispersion} \\ G_R(y) > G_I(y) \implies \text{concentration} \end{cases}$

Diversification

Reranking

MMR: $\arg\max_{i \in R \setminus S} [\lambda Sim^{us}(i,u) - (1-\lambda) \max_{j \in S} Sim^{it}(i,j)]$

$\begin{cases} \lambda \ large \implies \text{only care about relevance} \\ \lambda \ small \implies \text{only care about diversity} \end{cases}$

Adomavicius & Kwon: $rank'_u(i,t) = \begin{cases} rank^{(pop)}(i) \text{ if } r(u,i) \geq t \\ \alpha_u + rank_u(i) \text{ otherwise} \end{cases}$

Spotify

Computing if user is specialist

User representation: $\gamma_u = \frac{1}{|H|} \sum_{j=1}^{|H|} \gamma_{H_j}$

Generalist - Specialist: $GS(u) = \frac{1}{|H|} \sum_{j=1}^{|H|} \frac{\gamma_{H_j} \cdot \gamma_u}{||\gamma_{H_j}||||\gamma_u||}$

$\begin{cases} GS(u) \approx 0 \implies \text{generalist} \\ GS(u) \approx 1 \implies \text{specialist} \end{cases}$

Inactive users tends to be specialist.

Calibration

Calibrate similar attribute proportions

KL divergence: $KL(p,q) = \sum_g p(g|u) \log \frac{p(g|u)}{q(g|u)}$

Where:

- p is the historical distribution of attribute g for user u
- q is the current distribution of attribute g for user u.

Means: p and q should match.

Beyond CF

CARS	Context-Aware Recsys
Rating Tensor	$\tilde{R} = X \cdot Y \cdot Z \quad Z \in \mathbb{R}^{\|F\| \times \|K\|}$
Loss function	$\arg\min_{X,Y,Z} \|\|R - \tilde{R}\|\|_2 + \lambda_1 \|\|X\|\|_2 + \lambda_2 \|\|Y\|\|_2 + \lambda_3 \|\|Z\|\|_2$

Session-based

- Anonymous users
- can only optimize short-term preferences
- Is **Session-aware**, if past sessions are available.
  - can optimize long-term preferences

Knowledge-based

- Explicitly encode additional knoledge, requires knoledge engineering
- Useful for certain domains (chatbots, conversational systems)

Sequence-Aware When sequence matters

Input

- if CF, is a set of user interactions
- Rely on a specific sequence of interactions, not on a session.
- × cannot use URM!

Goal: Predict the next item in the sequence

Approach

- Basics: CO-occurrence, Markov Chains, Heuristic
- KNN: find past sessions similar.
- Sequence-learning: Embedding, RNN, Attention.