# RECSYS Formula Sheet

## Global Effects

1. Global Bias $\mu$      $\mu = \frac{\sum_{u,i \in T} r_{ui}}{N_T + C}$
2. Normalization $r'_{ui}$      $r'_{ui} = r_{ui} - \mu$
3. Item Shrink Average Rating $b_i$    $b_i = \frac{\sum_{u,i \in T} r'_{ui}}{N_T + C}$
4. Normalization (Again)    $r''_{ui} = r'_{ui} - b_i \ \forall u \in U, i \in I$
5. User Shrinked Average Rating $b_u$    $b_u = \frac{\sum_{i \in I} r''_{ui}}{N_T}$

### 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 |
| Coverage | Ability of recommending items that the user has not seen |
| Novelty | Ability of recommending unknown items |
| Diversity | Ability of recommending different items |
| Consistency | Ability to give consistent recommendations |
| Confidence | Measure how much the model is sure about its recommendations |
| Serendipity | Ability of recommending unexpected items |

## Classification Metrics
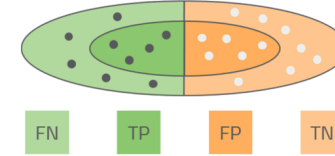


| | | | |
|---|---|---|---|
| FN | TP | FP | TN |

| Metrics | Formula |
|---|---|
| Recall | $\frac{TP}{FN + TP}$ |
| Precision | $\frac{TP}{TP + FP}$ |
| Fallout | $\frac{FP}{FP + TN}$ |
| 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 | $s_{ij} = \frac{\vec{i} \cdot \vec{j}}{\|\vec{i}\| \cdot \|\vec{j}\|}$ |
| Shrinked Cosine | $s_{ij} = \frac{\vec{i} \cdot \vec{j}}{\|\vec{i}\| \cdot \|\vec{j}\| + C}$ |

### Rating Estimation

| | |
|---|---|
| Single Rating | $\tilde{r}_{ui} = \frac{\sum_{j \in N_k(i)} r_{uj} \cdot s_{ij}}{\sum_{j \in N_k(i)} s_{ij}}$ |
| 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 | $\tilde{r}_{ui} = \frac{\sum_{j \in N_k(i)} r_{uj} \cdot s_{ji}}{\sum_{j \in N_k(i)} s_{ji}}$ |

### 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 | $s_{ij} = \frac{\vec{i} \cdot \vec{j}}{\|\vec{i}\|_2 \cdot \|\vec{j}\|}$ | Implicit ratings |
| Jaccard Similarity | $s_{ij} = \frac{i \cap j}{i \cup j}$ | 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!

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

**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 | $S^* = \arg\min_S \|R - RS\|_2$ |
| Constraints on S | $diag(S) = 0$ |
| Lasso Regression Regularization | $S^* = \arg\min_S (\|R - RS\|_2 + \lambda \|S\|_1)$ |
| Ridge Regression Regularization | $S^* = \arg\min_S (\|R - RS\|_2 + \lambda \|S\|_2)$ |
| Elastic Net Regularization | $S^* = \arg\min_S (\|R - RS\|_2 + \lambda_1 \|S\|_1 + \lambda_2 \|S\|_2)$ |

## 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 | $x_{uij} = \tilde{r}_{ui} - \tilde{r}_{uj}$ |

### 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.

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

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

# 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)**
- $\tilde{r}_{ui} = \mu + b_u + b_i + \sum_k x_{uk} \cdot y_{ki}$
- $\mu^*, b_u^*, b_i^*, X^*, Y^* = \min_{\mu, b_u, b_i, X, Y} E(\dots)$

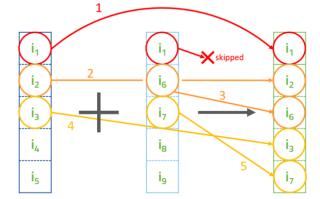| | |
|---|---|
| Asymmetric SVD (m-b) | $\tilde{R} = RZY , X = RZ$ |
| ● pure SVD (m-b) | $\tilde{R} = U_k \Sigma_k V_k^T = R V_k V_k^T$ |

# Hybrid Recommenders
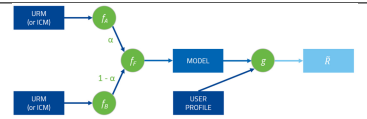
Linear Combination



List Combination



Pipelining
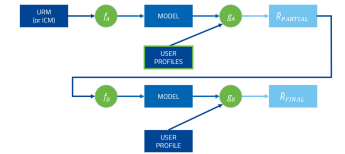


Model Merging
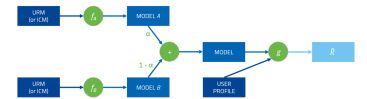


Co-Training



## 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**
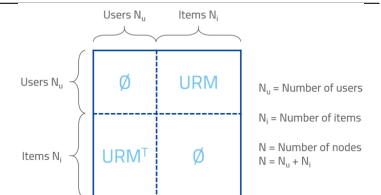
## Graph-Based - 2

**PageRank**

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

**P3Alpha** $P_3\alpha$

| | |
|---|---|
| Metapath | $U \to I \to U \to I$ |
| 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 $\alpha$, 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 | |

    $\times$ Cannot use ground truth (too many negative samples)

    $\times$ Cannot use just positive samples (no learning)

    $\checkmark$ 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 | |

    $\checkmark$ Fast and highly efficient

    $\checkmark$ Due the Frobenius norm, it tries to compute $R = RS$, thus repoducing the input as output such as an autoencoder.

    $\times$ 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: $S = W_e \cdot W_d$

## 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

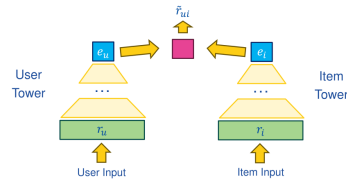### Variational Autoencoders (Mult-VAE)   Encoding a input as a distribution

### 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 $Y = W_i^I$