

Systèmes de recommandation

Shengrui Wang

November 18, 2024



Plan

Introduction

Mesures de similarité

Filtrage collaboratif basé sur le voisinage

Evaluation d'un système de recommandation

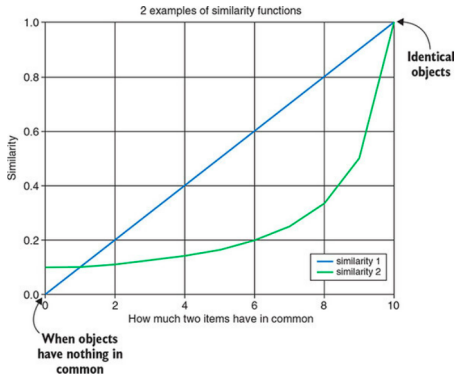
Techniques avancées

Introduction

- De manière générale nous avons,
 - ▶ Introduit la notion de système de recommandation,
 - ▶ Vu quelques techniques de recommandation en particulier celles basées sur le filtrage par contenu et sur le filtrage collaboratif,
 - ▶ Constaté qu'on ne pourrait faire de la recommandation sans parler de similarité soit entre utilisateurs soit entre items.
- Pour cette partie du cours, nous allons,
 - ▶ Davantages parler des métriques qui nous aide à faire de la recommandation.
 - ▶ Voir comment évaluer un système de recommandation.

Mesures de similarité

- Le choix de la métrique ou de la mesure de similarité a un enjeu crucial lorsqu'on fait de la recommandation,



- Il n'y a vraiment pas un choix ultime, tout dépend du jeu de données.

Coefficient de communauté: indice de Jaccard

- Applicable avec des données binaires



| | Comedy | Action | Comedy | Action | Drama | Drama |
|-----------|--------|--------|--------|--------|-------|-------|
| Sara | 1 | 1 | 0 | 0 | 0 | 0 |
| Jesper | 1 | 1 | 1 | 0 | 0 | 0 |
| Therese | 1 | 0 | 0 | 0 | 0 | 0 |
| Helle | 0 | 1 | 0 | 1 | 0 | 0 |
| Pietro | 0 | 0 | 0 | 0 | 1 | 1 |
| Ekaterina | 0 | 0 | 0 | 0 | 1 | 1 |

$Sim(U_u, U_v) = \frac{|I_u \cap I_v|}{|I_u \cup I_v|}$ avec I_u , l'ensemble d'items l'utilisateur U_u a achetés.

$Sim(I_i, I_j) = \frac{|U_i \cap U_j|}{|U_i \cup U_j|}$ avec U_i , l'ensemble d'utilisateurs qui ont acheté I_i .

- S'applique sur les *ratings*,

$$\text{dist}(U_u, U_v) = \left(\sum_{I_i \in I_u \cap I_v} |r_{u,i} - r_{v,i}|^p \right)^{1/p}$$
$$\text{dist}(I_i, I_j) = \left(\sum_{U_u \in U_i \cap U_j} |r_{u,i} - r_{u,j}|^p \right)^{1/p}$$

- Et ainsi obtenir la similarité comme suit

$$\text{sim}(\square, \star) = \frac{1}{1 + \text{dist}(\square, \star)}$$

Similarité cosinus et corrélation de Pearson

S'applique sur les *ratings*

■ Cosinus,

$$\begin{aligned} \text{sim}(U_u, U_v) &= \frac{\sum_{I_i \in I_u \cap I_v} r_{u,i} \times r_{v,i}}{\sqrt{\sum_{I_i \in I_u \cap I_v} r_{u,i}^2} \sqrt{\sum_{I_i \in I_u \cap I_v} r_{v,i}^2}} \\ \text{sim}(I_i, I_j) &= \frac{\sum_{U_u \in U_i \cap U_j} r_{u,i} \times r_{u,j}}{\sqrt{\sum_{U_u \in U_i \cap U_j} r_{u,i}^2} \sqrt{\sum_{U_u \in U_i \cap U_j} r_{u,j}^2}} \end{aligned}$$

qui pourrait être normalisé comme suit,

$$\begin{aligned} \text{sim}(U_u, U_v) &= \frac{\sum_{I_i \in I_u \cap I_v} (r_{u,i} - \bar{r}_u) \times (r_{v,i} - \bar{r}_v)}{\sqrt{\sum_{I_i \in I_u \cap I_v} (r_{u,i} - \bar{r}_u)^2} \sqrt{\sum_{I_i \in I_u \cap I_v} (r_{v,i} - \bar{r}_v)^2}} \\ \text{sim}(I_i, I_j) &= \frac{\sum_{U_u \in U_i \cap U_j} (r_{u,i} - \bar{r}_i) \times (r_{u,j} - \bar{r}_j)}{\sqrt{\sum_{U_u \in U_i \cap U_j} (r_{u,i} - \bar{r}_i)^2} \sqrt{\sum_{U_u \in U_i \cap U_j} (r_{u,j} - \bar{r}_j)^2}} \end{aligned}$$

avec \bar{r}_u la moyenne des *ratings* de l'utilisateur U_u sur les items pris dans $I_u \cap I_v$
et \bar{r}_i la moyenne des *ratings* qu'a eu l'item I_i parmi les utilisateurs pris dans $U_i \cap U_j$

Similarité cosinus et corrélation de Pearson

■ Corrélation de Pearson,

$$\begin{aligned} \text{sim}(U_u, U_v) &= \frac{\sum_{I_i \in I_u \cap I_v} (r_{u,i} - \bar{r}_u) \times (r_{v,i} - \bar{r}_v)}{\sqrt{\sum_{I_i \in I_u \cap I_v} (r_{u,i} - \bar{r}_u)^2} \sqrt{\sum_{I_i \in I_u \cap I_v} (r_{v,i} - \bar{r}_v)^2}} \\ \text{sim}(I_i, I_j) &= \frac{\sum_{U_u \in U_i \cap U_j} (r_{u,i} - \bar{r}_i) \times (r_{u,j} - \bar{r}_j)}{\sqrt{\sum_{U_u \in U_i \cap U_j} (r_{u,i} - \bar{r}_i)^2} \sqrt{\sum_{U_u \in U_i \cap U_j} (r_{u,j} - \bar{r}_j)^2}} \end{aligned}$$

avec \bar{r}_u la moyenne des *ratings* de l'utilisateur U_u sur tous les items qu'il a noté.

et \bar{r}_i la moyenne des notes qu'a eu l'item I_i .

Similarité cosinus et corrélation de Pearson

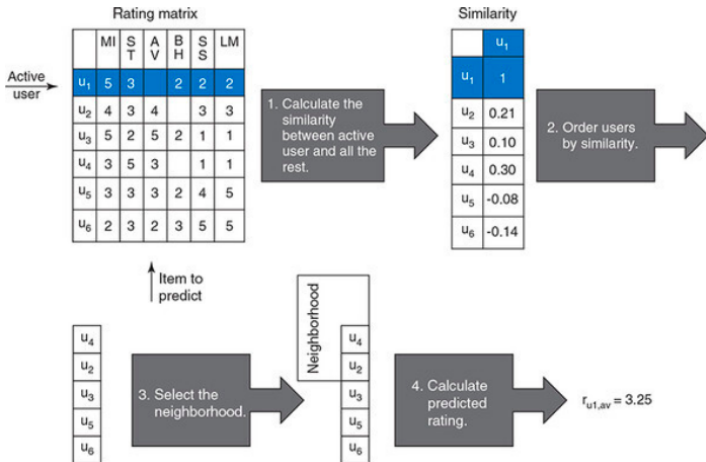
Faire le calcul des similarités cosinus, cosinus normalisé et Pearson entre utilisateurs et entre items dans le cas suivant:



| | Comedy | Action | Comedy | Action | Drama | Drama |
|-----------|--------|--------|--------|--------|-------|-------|
| Sara | 5 | 5 | | 2 | 2 | 2 |
| Jesper | 4 | 5 | 4 | | 3 | 3 |
| Therese | 5 | 3 | 5 | 2 | 1 | 1 |
| Helle | 3 | | 3 | 5 | 1 | 1 |
| Pietro | 3 | 3 | 3 | 2 | 4 | 5 |
| Ekaterina | 2 | 3 | 2 | 3 | 5 | 5 |

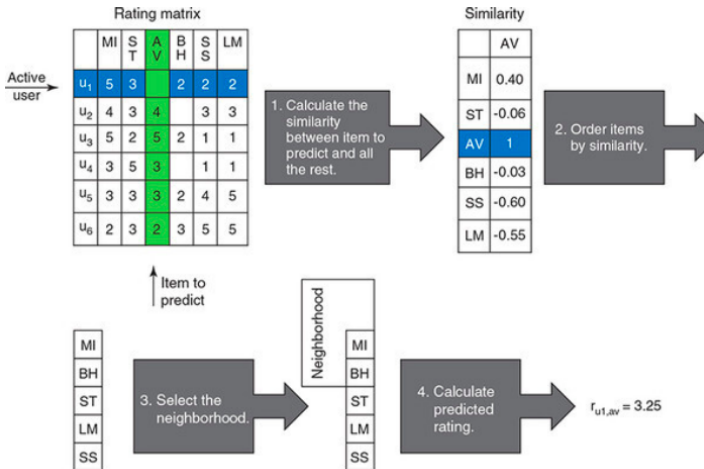
Filtrage collaboratif basé sur le voisinage

Voisinage utilisateurs



Filtrage collaboratif basé sur le voisinage

Voisinage items

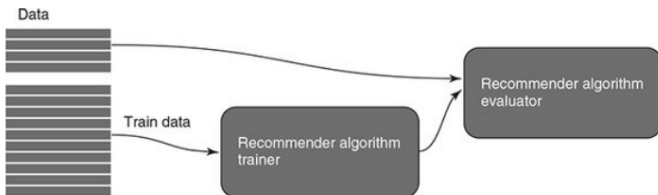


Évaluation d'un système de recommandation

- Une fois que votre algorithme de recommandation a été défini, il est important de savoir si ce dernier effectue de bonnes recommandations.
- Pour ce faire nous avons besoin de l'évaluer.
- L'évaluation d'un modèle de recommandation peut se faire de deux manières distinctes:
 1. Évaluation hors ligne
 2. Évaluation en ligne

Évaluation hors ligne

- L'évaluation dans ce cas de figure requiert une bonne quantité de données pour s'assurer que notre modèle de recommandation puisse faire une bonne généralisation.
- À partir des données existantes, on va créer un scénario où l'on prendra une partie des données comme information connue (données d'entraînement) et la partie restante comme information inconnue (données de test)



- Dans l'échantillon d'entraînement, on va apprendre notre modèle de recommandation à reconnaître les *ratings* des utilisateurs.
- Pour cela on va cacher certains *ratings* et voir comment notre modèle de recommandation s'adapte dans sa tâche de prédiction: On parle d'entraînement du modèle.
- L'entraînement du modèle recommandation requiert une métrique d'évaluation.
- L'objectif de l'entraînement est de trouver le(s) paramètre(s) de notre modèle de recommandation qui minimiserai(en)t la métrique d'évaluation.

- Formellement,
- Étant donné un modèle de recommandation défini par la fonction $f()$ (qui prédit le rating d'un utilisateur), ayant pour paramètre ω et une mesure d'évaluation $Eval()$,
- L'entraînement sur un ensemble $Train$ consiste à trouver le meilleur paramètre $\hat{\omega}$ qui minimiserait notre mesure d'évaluation.

$$\hat{\omega} = \underset{\omega}{argmin} \{ Eval(f(U_u|\omega), r_{u,i}) \mid \forall U_u, I_i \in Train \}$$

Exemple,

- Étant donné un système de recommandation où les *ratings* r peuvent prendre les valeurs: 1, 2, 3, 4 ou 5.
- Supposons que notre fonction de recommandation soit une moyenne pondérée donnée par,

$$f(I_i|U_u, \Omega) = \frac{1}{|\mathbf{U}_{v \sim u}|} \Omega \cdot \left(\sum_{U_v \in \mathbf{U}_{v \sim u}} \sum_{I_j \in \mathbf{I}_v} \sum_{r=1}^5 r_{v,j} \Delta_r(r_{v,j}) \right)^T$$

Avec $\mathbf{U}_{v \sim u} = \{U_v, U_v \neq U_u / \mathbf{I}_u \cap \mathbf{I}_v \neq \emptyset\}$

$\Omega = (\omega_1, \omega_2, \omega_3, \omega_4, \omega_5)$

$\Delta_r(r_{v,j}) = (1, 0, 0, 0, 0)$ si $r = r_{v,j} = 1$ (resp. $\Delta_r(r_{v,j}) = (0, 1, 0, 0, 0)$

si $r = r_{v,j} = 2, \dots, \Delta_r(r_{v,j}) = (0, 0, 0, 0, 1)$ si $r = r_{v,j} = 5$)

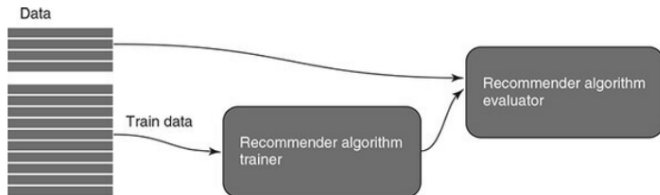
- $Eval(a, b) = |a - b|$ notre fonction de perte.
- Entraîner notre fonction de recommandation, reviendrait à chercher les meilleurs paramètres $\hat{\Omega}$ qui minimisent la fonction

$$\hat{\Omega} = \min_{\Omega} \sum_{U_u, I_i \in Train} |f(I_i | U_u, \Omega) - r_{u,i}|$$

- Ils existent plusieurs moyens d'estimation des paramètres tels que: la descente des gradients, le maximum de vraisemblance, Newton-Raphson etc. ...
- Nous n'aborderons pas les méthodes d'estimation des paramètres dans ce cours ... 😞

Évaluation hors ligne – Validation croisée

- Souvenons nous avons besoin de sélectionner une partie de nos données pour représenter l'ensemble d'entraînement et une autre partie pour représenter l'ensemble de test:

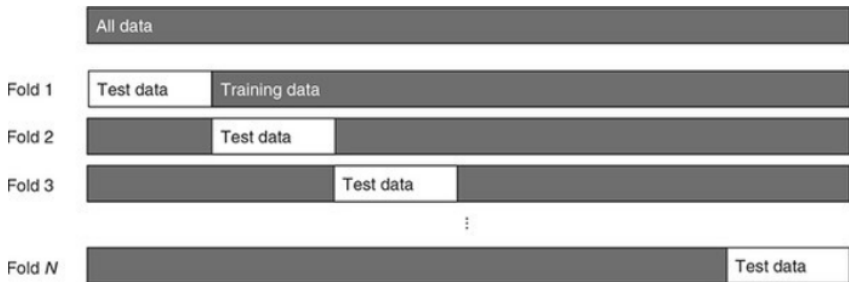


- **Il est important de noter que les paramètres estimés valent particulièrement pour ce cas de figure. On ne saurait encore conclure si notre fonction de recommandation fait une bonne recommandation.**

Évaluation hors ligne – Validation croisée

- La subdivision des données en deux sous-ensembles pourraient générer des cas de figure favorables/défavorables.
- Pour s'assurer qu'on ne favorise pas un cas en particulier, on pourrait générer à partir de nos données, plusieurs sous-ensemble entraînements et tests.
- Ensuite tester notre fonction de recommandation pour chacun des cas et prendre la moyenne.
- Ce type de procédé est généralement connu sous le nom de *validation croisée*.

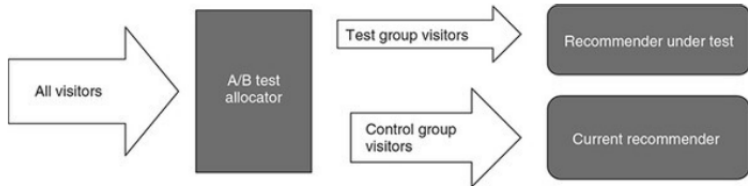
Évaluation hors ligne – Validation croisée



- Contrairement à la méthode hors ligne qui ne base que sur la connaissance des données existantes,
- l'évaluation en ligne tient compte des nouveaux cas de figures pour juger de la performance du modèle de recommandation.
- Pour une évaluation en ligne, on a très souvent besoin de deux systèmes de recommandation:
 1. Un système de recommandation existant/courant.
 2. Un nouveau système de recommandation.

- Sur la base des utilisateurs existants, on définit deux sous-ensembles: Un ensemble de *test* et un ensemble de *contrôle*.
- Les utilisateurs pris dans l'ensemble de *contrôle* continueront à être soumis au système de recommandation courant;
- tandis que, les utilisateurs pris dans l'ensemble de *test* seront soumis au nouveau système de recommandation.
- Rouler les deux systèmes de recommandation pendant une certaine durée (quelques jours, semaines ou des mois ...)
- Tirer une conclusion sur celui qui performe le mieux.

Évaluation en ligne



- Problèmes avec des approches simples
 1. Couverture limitée
 2. Sensibilité aux données éparses
- Méthodes basée sur la réduction de dimensionnalité
 1. Décomposition de la matrice de "ratings"
 2. Décomposition de la matrice de similarité
- Méthodes basées sur le parcours des graphes (**Cette partie de méthodes ne sera pas abordée**)
 1. Similarité basée sur chemins de parcours
 2. Similarité basée sur la marche aléatoire

■ Couverture limitée

1. Les mesures basées sur la corrélation des "ratings" comparent les "ratings" sur les items communs. Ainsi, les utilisateurs ne peuvent être voisins que s'ils ont évalué des items communs.
2. Dans la pratique, les utilisateurs n'ayant évalué aucun item commun peuvent néanmoins avoir des préférences similaires

■ Sensibilité aux données éparses

1. Les utilisateurs n'évaluent généralement qu'une petite proportion des items disponibles
2. Les utilisateurs ou les items nouvellement ajoutés au système peuvent n'avoir aucune "rating" du tout (démarrage à froid)
3. Donc, il est peu probable que deux utilisateurs ou items aient des "ratings" communes, ce qui peut affecter sérieusement la précision des approches basées sur le voisinage

Méthodes de réduction de dimensionnalité

- L'idée : projeter les utilisateurs (ou items) dans un espace latent compact pour capturer leurs caractéristiques les plus importantes
 1. Le sous-espace dense des caractéristiques (de haut niveau), à la place de l'espace de ratings, permet de découvrir plus de relations significatives.
 2. En particulier, une relation entre deux utilisateurs peut être trouvée, même s'ils n'ont pas évalué les mêmes items.
- Deux approches principales
 1. Décomposition de la matrice de "ratings"
 2. Décomposition de la matrice de similarité

Méthodes de décomposition de la matrice de "ratings"

■ Méthode basée sur LSI (Latent Semantic Indexing) :

- ▶ \mathbf{R} : la matrice de rating de taille $|\mathbf{U}| \times |\mathbf{I}|$
- ▶ \mathbf{R} est approchée par
$$\hat{\mathbf{R}} = \mathbf{P}\mathbf{Q}^t$$
avec
$$k := \text{rank}(\hat{\mathbf{R}}) \leq \text{rank}(\mathbf{R}) =: n$$
- ▶ \mathbf{P} : une matrice de facteurs utilisateurs de taille $|\mathbf{U}| \times k$
- ▶ \mathbf{Q} : matrice de facteurs items de taille $|\mathbf{I}| \times k$
- ▶ Le $u^{i\text{ème}}$ ligne p_u de \mathbf{P} représente la projection de l'utilisateur u dans un espace latent de k -dimensions
- ▶ Le $i^{i\text{ème}}$ ligne q_i de \mathbf{Q} représente la projection de l'item i dans un espace latent de k -dimensions

Méthodes de décomposition de la matrice de "ratings" (cont)

■ Méthode basée sur LSI (Latent Semantic Indexing) :

- ▶ $\hat{\mathbf{R}} = \mathbf{P}\mathbf{Q}^t$ résultant de la minimisation de

$$err() = \left\| \mathbf{R} - \mathbf{P}\mathbf{Q}^t \right\|^2 = \sum_{u,i} (r_{u,i} - p_u q_i^t)^2$$

- ▶ Cela équivaut à effectuer la SVD (décomposition en valeurs singulières) sur \mathbf{R}

$$\mathbf{R} = \bar{\mathbf{U}}\Sigma\bar{\mathbf{V}}^t$$

Ici, $\bar{\mathbf{U}}$ est la matrice de "vecteur singuliers gauches" de taille $|\mathbf{U}| \times r$, $\bar{\mathbf{V}}$ est la matrice de "vecteur singuliers droites" de taille $|\mathbf{I}| \times r$, et Σ est la matrice de valeurs singulières de taille $r \times r$.
 $r = \text{rang}(\mathbf{R})$ et $r \leq \min\{m, n\}$.

La convention d'écriture est : $\mathbf{R} = \mathbf{U}\Sigma\mathbf{V}^t$. $\bar{\mathbf{U}}$ et $\bar{\mathbf{V}}$ sont utilisés ici pour se différencier des \mathbf{U} et U_u utilisés dans ce chapitre.

■ Méthode basée sur LSI (Latent Semantic Indexing) :

- ▶ Étant donnée la décomposition $\mathbf{R} = \bar{\mathbf{U}}\Sigma\bar{\mathbf{V}}^t$ obtenue précédemment, la décomposition $\hat{\mathbf{R}} = \mathbf{P}\mathbf{Q}^t$ peut être obtenue en sélectionnant Σ_k (sous matrice de Σ) correspondant aux k plus grandes valeurs singulières, et $\bar{\mathbf{U}}_k$ et $\bar{\mathbf{V}}_k$ (sous matrices de $\bar{\mathbf{U}}$ et $\bar{\mathbf{V}}$) composées des vecteurs singuliers correspondants.

$$\hat{\mathbf{R}} = \bar{\mathbf{U}}_k \Sigma_k \bar{\mathbf{V}}_k^t$$

- ▶ Il suffit donc de définir $\mathbf{P} = \bar{\mathbf{U}}_k \Sigma_k^{1/2}$ et $\mathbf{Q} = \bar{\mathbf{V}}_k \Sigma_k^{1/2}$.
- ▶ Une fois \mathbf{P} et \mathbf{Q} obtenus, le (nouveau) score peut être calculé par

$$\hat{r}_{ui} = p_u q_i^t$$

où p_u est la ligne u de la matrice de \mathbf{P} et q_i est la ligne i de la matrice de \mathbf{Q} .

Exemple de LSI

- c1: **Human** machine **interface** for ABC computer applications
- c2: A **survey** of **user** opinion of computer **system response time**
- c3: The **EPS user interface** management **system**
- c4: **System** and **human system** engineering testing of **EPS**
- c5: Relation of **user** perceived **response time** to error measurement
- m1: The generation of random, binary, ordered **trees**
- m2: The intersection **graph** of paths in **trees**
- m3: **Graph** minors IV: Widths of **trees** and well-quasi-ordering
- m4: **Graph** minors: A **survey**

Exemple de LSI

- La matrice mots clés - documents : $A =$

| | c1 | c2 | c3 | c4 | c5 | m1 | m2 | m3 | m4 |
|------------------|----|----|----|----|----|----|----|----|----|
| human | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
| interface | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| computer | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| user | 0 | 1 | 1 | 0 | 1 | 0 | 0 | 0 | 0 |
| system | 0 | 1 | 1 | 2 | 0 | 0 | 0 | 0 | 0 |
| response | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 0 |
| time | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 0 |
| EPS | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 |
| survey | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| trees | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 0 |
| graph | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 |
| minors | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 |

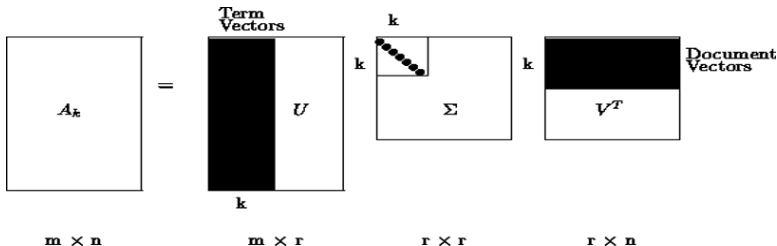
Exemple de LSI

- Décomposition des valeurs singulières (SVD) :

$$\mathbf{A} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^t$$

- SVD réduit :

$$\mathbf{A}_k = \mathbf{U}_k \mathbf{\Sigma}_k \mathbf{V}_k^t$$



Exemple de LSI

■ $U =$

| | | | | | | | | |
|------|-------|-------|-------|-------|-------|-------|-------|-------|
| 0.22 | -0.11 | 0.29 | -0.41 | -0.11 | -0.34 | 0.52 | -0.06 | -0.41 |
| 0.20 | -0.07 | 0.14 | -0.55 | 0.28 | 0.50 | -0.07 | -0.01 | -0.11 |
| 0.24 | 0.04 | -0.16 | -0.59 | -0.11 | -0.25 | -0.30 | 0.06 | 0.49 |
| 0.40 | 0.06 | -0.34 | 0.10 | 0.33 | 0.38 | 0.00 | 0.00 | 0.01 |
| 0.64 | -0.17 | 0.36 | 0.33 | -0.16 | -0.21 | -0.17 | 0.03 | 0.27 |
| 0.27 | 0.11 | -0.43 | 0.07 | 0.08 | -0.17 | 0.28 | -0.02 | -0.05 |
| 0.27 | 0.11 | -0.43 | 0.07 | 0.08 | -0.17 | 0.28 | -0.02 | -0.05 |
| 0.30 | -0.14 | 0.33 | 0.19 | 0.11 | 0.27 | 0.03 | -0.02 | -0.17 |
| 0.21 | 0.27 | -0.18 | -0.03 | -0.54 | 0.08 | -0.47 | -0.04 | -0.58 |
| 0.01 | 0.49 | 0.23 | 0.03 | 0.59 | -0.39 | -0.29 | 0.25 | -0.23 |
| 0.04 | 0.62 | 0.22 | 0.00 | -0.07 | 0.11 | 0.16 | -0.68 | 0.23 |
| 0.03 | 0.45 | 0.14 | -0.01 | -0.30 | 0.28 | 0.34 | 0.68 | 0.18 |

Exemple de LSI

■ $\Sigma =$

3.34

2.54

2.35

1.64

1.50

1.31

0.85

0.56

0.36

Exemple de LSI

■ $V =$

| | | | | | | | | |
|-------|-------|-------|-------|-------|-------|-------|------|-------|
| 0.20 | 0.61 | 0.46 | 0.54 | 0.28 | 0.00 | 0.01 | 0.02 | 0.08 |
| -0.06 | 0.17 | -0.13 | -0.23 | 0.11 | 0.19 | 0.44 | 0.62 | 0.53 |
| 0.11 | -0.50 | 0.21 | 0.57 | -0.51 | 0.10 | 0.19 | 0.25 | 0.08 |
| -0.95 | -0.03 | 0.04 | 0.27 | 0.15 | 0.02 | 0.02 | 0.01 | -0.03 |
| 0.05 | -0.21 | 0.38 | -0.21 | 0.33 | 0.39 | 0.35 | 0.15 | -0.60 |
| -0.08 | -0.26 | 0.72 | -0.37 | 0.03 | -0.30 | -0.21 | 0.00 | 0.36 |
| 0.18 | -0.43 | -0.24 | 0.26 | 0.67 | -0.34 | -0.15 | 0.25 | 0.04 |
| -0.01 | 0.05 | 0.01 | -0.02 | -0.06 | 0.45 | -0.76 | 0.45 | -0.07 |
| -0.06 | 0.24 | 0.02 | -0.08 | -0.26 | -0.62 | 0.02 | 0.52 | -0.45 |

- Problème avec la méthode basée sur LSI :
 - ▶ Pour appliquer SVD, $\mathbf{R} = \bar{\mathbf{U}}\bar{\Sigma}\bar{\mathbf{V}}^t$, \mathbf{R} doit être définie partout, ce qui n'est souvent pas le cas.
 - ▶ Une solution pratique : affectant les valeurs par défaut à r_{ui} . Cela crée un biais
- Une solution plus juste : créer l'espace latent par optimisation

$$err(P, Q) = \sum_{r_{u,i} \in \mathbf{R}} \left((r_{u,i} - p_u q_i^t)^2 + \lambda (\|p_u\|^2 + \|q_i\|^2) \right)$$

La minimisation de $err(P, Q)$ se fait pas une approche itérative (ne sera pas abordée dans ce cours. Voir Bell, R. et al (KDD 2007), Koren (KDD 2008), Takacs (JMLR)).

- Le sous-produit de la projection est que les coordonnées de projection de u et i peuvent ensuite être utilisées pour estimer la similarité entre les utilisateurs et entre les éléments.

$$w_{u,v} = \text{sim}(U_u, U_v) = p_u p_v^t$$

$$w_{i,j} = \text{sim}(I_i, I_j) = q_i q_j^t$$

Voir Billsus (ICML 98).

- Méthode basée sur la diagonalisation de la matrice de similitude \mathbf{W} .

$$err(P) = \left\| \mathbf{W} - \mathbf{P}\mathbf{P}^t \right\|^2 = \sum_{u,v} (w_{u,v} - p_u p_v^t)^2$$

L'optimisation de cette fonction d'erreur donne lieu à la solution suivante :

$$\mathbf{W} = \mathbf{V}\mathbf{\Lambda}\mathbf{V}^t$$

où $\mathbf{\Lambda}$ est la matrice diagonale composée des valeurs propres de \mathbf{W} , et \mathbf{V} est une matrice orthogonale composée des vecteurs propres de \mathbf{W} .

(cont)

- Similaire à l'approche SVD, \mathbf{W} peut être approximé par :

$$\hat{\mathbf{W}} = \mathbf{P}_k \mathbf{P}_k^t$$

où \mathbf{P}_k est obtenue en utilisant les k plus grandes valeurs propres et les vecteurs propres correspondants, c'est-à-dire :

$$\mathbf{P}_k = \mathbf{V}_k \Lambda_k^{1/2}$$

Voir Goldberg, K. et al (2001) qui ont construit le système Eigentaste qui recommande des blagues.