

Notes de Cours LOG6308

Olivier Sirois

2017-10-10

Contents

1	Introduction et Filtres Collaboratifs	4
1.0.1	Amazon.com : très efficace!, Intérêts	4
1.0.2	Connaissance	5
1.0.3	Intentions et Tâches	5
1.1	Quelques Définition et Taxonomies	6
1.1.1	Données Utilisées	6
1.2	item-item et utilisateur-utilisateur	8
1.2.1	utilisateurs-utilisateurs	10
1.2.2	items-items	10
1.3	Approche Mémoire vs Approche Modèle	10
1.4	Algorithme Utilisateur-Utilisateur et Items-Items	10
1.4.1	Utilisateurs-Utilisateur	11
1.4.2	K nearest Neighbour	13
1.4.3	item-item	14
1.5	TF-IDF	16
1.6	Mesure de Performance	16
2	Méthode contenue, PageRank et un peu de Bayésien	18
2.1	Modèles Probabiliste	18
2.1.1	Correction de Laplace	19
2.1.2	Exemple	20
2.2	Autres Approches probabiliste	20
2.3	Modèle Par Classe (Clustering)	21
2.4	Arbre de décision	22
2.4.1	Breese et Coll.	22
2.5	PageRank (Popularité)	23
2.5.1	Calcul de manière vectoriel	25
2.5.2	Exemple	25
2.5.3	Puissance des Matrices	25
2.6	Rappel Théorie Bayésienne	26

3	Réduction de Dimensions	28
3.1	Intuition	28
3.2	Techniques de Décomposition Matricielle	29
3.3	SVD	30
3.4	Factorisation en Valeur non-négative + Valeurs manquantes . . .	32
3.5	Biais	32
4	Modèle de Connaissance de l'utilisateur	34
4.1	Factorisation des matrices	36
4.2	Utilisation d'un Modèle Bayésien	37
4.2.1	Problèmes Combinatoire	38
4.3	Modèles d'espaces (sous-ensembles)	38
4.3.1	Inférence et choix de Question	38
4.4	Autres Modèles	38
4.5	Régression Logistique	39
4.5.1	Modèles TRI	40

Chapter 1

Introduction et Filtres Collaboratifs

Selon les notes, les recommandations prennent la forme d'un filtre d'information. On essaie de filtrer l'information qu'on ne désire pas avoir. On peut séparer les filtres en deux parties, soit:

- Filtre passifs: Comme les vidéos et pièces de musique présentées avec le dernier choix fait sur un site, style youtube
- Filtres Actifs: Comme des courriels de recommandation (se faire envoyer des courriels)

On peut définir un système de recommandation comme étant une personnalisation d'interface qui font des choses comme exemples:

- curation d'articles de presse
- guide d'étude en fonction des acquis
- auto-complétion sensible au contexte et à l'historique d'utilisation

Généralement, la personnalisation à l'utilisateur nécessite trois types d'information à modéliser et à inférer:

- Les intérêts et préférences de l'utilisateur (Présent)
- Les compétences et connaissances de l'utilisateur (Passé)
- Les buts de l'utilisateur (Futur)

1.0.1 Amazon.com : très efficace!, Intérêts

Un utilisateur : seulement deux livres d'achetés

l'utilisateur à déjà deux livres sur 15 qui sont recommandés, alors c'est quand même pas mal.

Dans ces recommandations, plusieurs titres étaient fort pertinents, soit:

- Machine Learning, Tom Mitchell
- Categorical Data Analysis, Agresti
- Data Mining: Practical Machine Learning Tools and Techniques with Java Implementations, Witten & Frank

Plusieurs autres sites tel que Youtube, Itunes, Netflix, WatchOn offrent des services similaires.

1.0.2 Connaissance

Certains systèmes essaient de faire des recommandations basées sur les connaissances de certains utilisateurs. Un système comme ALEKS sert à faire des recommandations d'études basées sur les forces et faiblesses de l'utilisateur pour conformer à un certain programme d'étude. Le système pose des questions et analyse les réponses de l'utilisateur et en se basant là-dessus, il est capable de déterminer dans quel champ d'étude il devrait travailler dessus pour améliorer sa note.

Un système de ce genre donne environ 1 sigma (un écart-type) d'amélioration en moyenne chez les utilisateurs. En moyenne, du tutorat individuel améliore d'environ 2 sigma. Cette marque pourrait être mise comme étant le but de ce genre de systèmes. En moyenne, les systèmes d'aide d'apprentissage aident d'environ 0.3 sigma, ce qui n'est quand même pas négligeable.

1.0.3 Intentions et Tâches

Un exemple de système qui aide à accomplir des tâches de travail est le système de complétion d'Eclipse. En le mettant en contexte, il faut réaliser que certains des librairies contiennent des classes contenant plus de 381 méthodes. Ça peut être difficile de trouver la méthode qui fait exactement ce que l'on cherche, alors certains systèmes sont faits pour analyser ton code, et te faire des recommandations de méthodes basées sur les travaux que tu as fait et l'objectif que tu as. Évidemment, modéliser ce genre de système peut être très complexe et n'est peut-être pas nécessairement souvent correct, mais bon, c'est mieux que rien.

Petit rappel que Microsoft a longtemps essayé de faire un système de ce genre (paperclip).

1.1 Quelques Définition et Taxonomies

Systèmes de Recommandations et Filtres Collaboratifs Ce sont des systèmes qui utilisent des données statistique pour créer et regrouper des utilisateurs selon différentes dimensions (intérêt, navigation, opinions, etc). Une fois regroupé, on détermine les traits et caractéristique communes et déterminantes pour chaque individu groupe afin d'identifier les recommandations pertinentes, c'est-à-dire, des items qui sont bien rater selon les dimensions utiliser dans le groupes d'utilisateur observer.

Les recommandations peuvent aussi parfois reposer sur le contenu de ceux-ci (content-based).

On peut regrouper les systèmes de recommandations en différentes familles. Comme par exemple:

- **Personnalisation** : Des adaptations d'éléments d'interface basé sur un profil personnel
- **Accélérateurs**: Des systèmes qui ont pour but d'accélérer certains aspect. Comme par exemple de l'auto-complétion, du pré-chargement(prefetching), de l'auto-correction comme DWIM (Do what I mean)
- **Assistants et Agents**: Comme le paperclip de microsoft, Clipper.. Ce sont des systèmes de dialogue qui sont censé aider les gens avec des tâches facile/redondante. Le dialogue coopératif et d'aide est basé sur la reconnaissance des intentions et du plan, plus ou moins structuré et prédéterminé, entre un assistant et un assisté
- **Environnement d'apprentissage**: Comme vu précédemment, ALEKS, GradeSummit, Lisp Tutor, Andes Tutor. C'est systèmes sont des environnement intelligents d'apprentissage qui utilisent plusieurs approches pour fournir un encadrement à l'apprenant, notamment pour fournir des exercices et capsule d'apprentissage adaptés au besoin d'apprentissage. Pour résumé, ils aident à apprendre et à assimiler la matière.

1.1.1 Données Utilisées

Normalement, on se sert de plusieurs sources d'information pour faire nos recommandations. Évidemment, le plus de différentes source d'information qu'on à accès, le plus de liberté et de possibilité que nous pouvons avoir pour faire nos recommandations. Les sources d'informations les plus couramment utilisés sont :

- **Les actions d'utilisateurs**: Simplement prendre en mémoire les actions que les utilisateurs font, de sorte qu'on essaie soit de trouver un pattern, ou d'associer certaines actions avec certains types de recommandations

- Historique de l'utilisation et de la navigation: Comme par exemple, une matrice utilisateur-visite, qui prend en compte le temps que l'utilisateur prend sur certains items/services. On peut associer avoir une matrice utilisateur-commande, qui prend en compte quel commande l'utilisateur se sert..
- Éléments de contexte: Comme par exemple, la popularité d'un document, la proximité sémantique de termes dans les documents eux-mêmes.. Et aussi, très important, la proximité de profils des différents utilisateurs.
- Combinaison de données: utilisation d'une différent combinaison de toutes les sources vues.

Pour aller un peu plus en détail, on défini la proximité entre deux utilisateurs comme étant la proximité entre les deux vecteurs représentant ces deux utilisateurs et non la position géographique. On peut utiliser cette même technique avec les documents/URL.

d'autre chose dont on peu se servir sont les probabilités. Nous allons les voir beaucoup plus en détails, mais la probabilité conditionnel est utilisé assez souvent dans la construction des modèles probabiliste. Voir Bayésien.

Comme par exemple: On peut définir la probabilité que l'utilisateur clique sur le lien Étudiant actuels de la page d'accueil de Poly(H) étant donnée qu'il a un cooki qui l'identifie comme étudiant (E_1) et qu'il a accédé à ce lien plus de 5 fois durant les 5 derniers jours(E_2).

$$P(H|E_1, E_2, E_3....E_n)$$

En générale, les actions de l'utilisateurs sont déterminées par les buts que cet utilisateurs veut atteindre. Que sa soit perdre sont temps, ou finir ses notes, l'utilisateur va agir en conséquence. Normalement, un but se décompose en sous-buts et récursivement jusqu'à avoir un ensemble d'actions séquentiel. On peut définir comment un but se transpose en actions par une grammaire récursive ou par un automate fini.

```

(define-goal editor (?texte)
  (XOR
    (copier-texte ?texte)
    (additionner-texte-a-doc ?texte ?doc)
    (insérer-plus ?texte)
    (majuscules ?texte)
    (change-texte ?texte ?texte2)
    (effacer ?texte)
    (dessine* (node-observed m323))
    (copier-vertical ?texte)
    (deplacer-vertical-plus ?texte)
    (deplacer-texte ?texte ?pos)))

(define-goal root ()
  (XOR
    (format-texte ?texte ?prop)
    (remplace-multiple ?t1 ?t2)
    (editor ?texte)
    (positionner ?pos)
    (divers)))

```

Figure 1.1: Exemple de grammaire recursive.

Il y a certaines autres aspects à prendre en considération. Entre autre, les **facteurs ergonomiques**.

- Transparence : D'où vien le rationnel de la recommandation. C'est-à-dire, la logique derrière la recommandation.
- Crédibilité : La recommandation est-elle fiable ou biaisé ?
- Contrôle : Est-ce le choix que je ferais si j'avais accès à toutes les options? suis-je pris dans un gettho d'information ? il faut que le système soit capable de bien gérer.

1.2 item-item et utilisateur-utilisateur

Le principe d'un filtre collaboratif est de recommander ou filtre l'information pertinente afin de la montrer à l'utilisateur. Comme par exemple:

- Filtres: Éliminer ou classifier les courriels indésirés; filtrer les nouvelles selon nos intérêts.
- Recommandations : suggestion d'achats croisé; publicité personnalisée..

l'émergence du web social et les algorithmes de systèmes de recommandations sont de plus en plus répandus. Les utilisations les plus anciennes des filtres collaboratifs remontent jusqu'en 1994 avec l'utilisation du projet GroupLens, qui est un filtre de message pertinents sur UseNet.

Aujourd'hui, les grands systèmes de recommandations:

- Amazon.com: Le plus grand détaillant en ligne, fait un usage intensif de différents types de recommandations
- Netflix: Recommandation de films/séries, il y a même eu un prix netflix (1M\$). Qui avait pour but d'améliorer les recommandations de film faites par Cinematch et de passer d'un RMSE de 0.96 jusqu'à 0.86 (10%). Le prix a été lancé en 2006 et gagné en septembre 2009. Deux montréalais ont fait partie de l'équipe gagnantes.

Le filtre d'informations peut soit être passif ou actif.

Système Passif Cherche à réduire le flot d'information, comme par exemple un filtre polluel. (Cherche à éliminer du mauvais)

Système Actif Cherche à créer un flot d'information en extrayant de l'information de différentes sources pour les livrer à l'utilisateur. (Cherche à ajouter du bon)

Le filtre se trouve généralement à trois endroits différents:

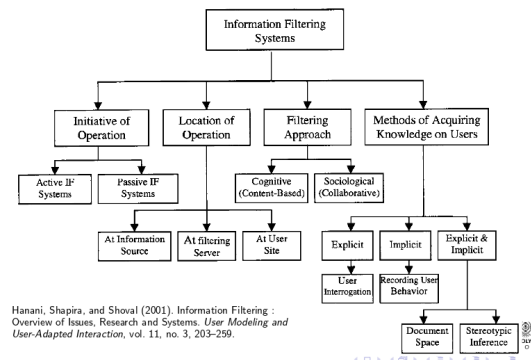


Figure 1.2: Différents Systèmes d'information

- Au poste de travail (client side)
- À la source d'information (Server Side)
- sur un serveur de filtre dédié

Un filtre passif est généralement situé sur le poste client étant donné qu'il ne cherche qu'à enlever de l'information. Par contre, un filtre actif sera normalement mis sur la source ou sur un serveur de filtre étant le grand nombre de données dont il doit traiter pour pouvoir effectuer sa recommandation.

De plus, on peut spécifier aussi la méthode d'acquisition de l'information sur l'utilisateur dans notre filtre. Normalement, on utilise des méthodes:

- Explicite: l'utilisateur spécifie ses préférences (généralement pas utiliser étant donné que ces méthodes sont très agressives et peuvent tanner l'utilisateur)
- Implicite: les préférences sont déduites du comportement de l'utilisateur
- Combinaison: l'utilisateur spécifie quelques paramètres de préférences mais un traitement est réalisé pour établir des informations implicites en même temps.

Normalement, l'approche **collaborative** du filtrage se base sur des facteurs sociologiques explicite et implicite. Comme par exemple, on utilise les votes (explicite) ainsi que des données d'achat et de consultations de pages (implicite) pour établir des similarités entre les différents utilisateurs. Par contre, l'approche collaborative peut être utilisée de différentes manières. Comme par exemple, on peut l'utiliser en comparant les items et établir des utilisateurs recommandés ou comparer des utilisateurs et recommandés des items.

1.2.1 utilisateurs-utilisateurs

On essaie de classer les gens selon leur intérêts, leurs votes, leur comportements ou toute autre dimensions qu'on juge pertinentes. On effectue des recommandations basées sur le groupes le plus près et le plus représentatif d'un individu. On recommandera ainsi un élément très caractéristique du groupe qui n'est pas dans le profil de l'individu

1.2.2 items-items

On essaie rechercher des items qui ont des profils d'intérêts similaires. On fait une recommandation en fonction de l'item affiché ou d'un historique d'items (on prend les items achetés/bien rater, on regarde les items les plus près selon certaines dimensions et on effectue une recommandation basé sur ce groupe.)

Approche Contenue Une approche différente, elle cherche à analyser les propriétés des items au lieu des différentes dimensions sociologique de l'approche collaborative (votes, temps passer). Mais cherche toujours les similarités entre les items.

1.3 Approche Mémoire vs Approche Modèle

On différencie aussi les différents algorithmes de systèmes de recommandations en fonction de la gestion de la mémoire. On différencie les algorithmes comme étant:

- Memory-Based (basé mémoire): On effectue une recherche dans une Base de Données et on l'utilise certains algorithmes avec des quantités massives d'informations pour effectuer nos recommandations
- Model-Based (basé modèle): C'est basé sur l'apprentissage d'un modèle. On utilise une base de données pour entraîner un modèle qui ensuite permet la prédiction de l'item pertinent sans avoir à faire une série de calculs intensifs. Par contre, il faut prendre en considération que ça peut être long d'entraîner un modèle.

1.4 Algorithme Utilisateur-Utilisateur et Items-Items

En regroupant l'information récoltée de nos méthodes implicites et explicites, on peut finalement les représenter sous forme d'une matrice:

On condense donc l'information des utilisateurs et des items dans une seule grande structure de données. Par contre, ce qui nous intéresse c'est le milieu de cette matrice où on voit les liens entre nos utilisateurs et nos produits. C'est

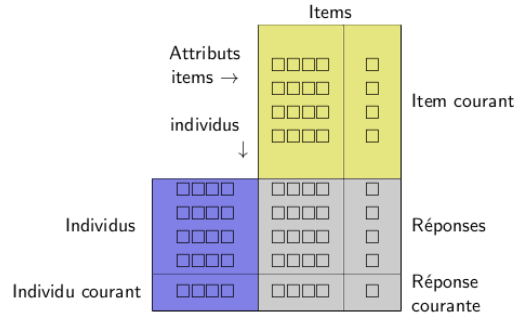


Figure 1.3: représentation de nos features dans une grande matrice

en faisant des opérations sur ces liens que nous allons pouvoir faire nos recommandations. Commençons premièrement par faire une recommandation en se basant sur les différents profils de nos utilisateurs.

1.4.1 Utilisateurs-Utilisateur

Prenons comme exemple initiales cette matrice contenant des votes de 4 utilisateurs pour 4 items différents.

Votes pour 4 utilisateurs

U	Item			
	1	2	3	4
U_1	5	1	?	2
U_2	4	1	1	3
U_3	4	2	1	2
U_4	1	4	3	2

Figure 1.4: notre matrice pour les 4 utilisateurs et 4 items

Notre objectif est de prédire les votes d'un utilisateur spécifique à partir de notre Base de Données des autres votes des utilisateurs.

La valeur estimé de l'utilisateur a pour un item j , $E(v_{a,j})$ est la somme

pondéré des votes des autres utilisateurs v_i qui ont des votes communs, soit:

$$E(v_{a,j}) = \bar{v}_a + k \sum_i^n w_{a,i} (v_{i,j} - \bar{v}_i)$$

Où a est notre utilisateur, j est notre item cibler et i est l'ensemble de tous les autres items.

Le poids w dans la formule peut représenter plusieurs chose, elle peut par exemple représenter une corrélation, une distance euclidienne, un coefficient de similarité quelconque défini par le système. la constante k est une normalisation des poids à un, qui peut être représenté par :

$$k = \frac{1}{\sum_i |w_{a,i}|}$$

Corrélation Pearson

Une estimation de poids très populaire est celle de la corrélation Pearson. la corrélation Pearson est une mesure statistique très commune qui à originalement été utilisé pour les filtres collaboratifs par le projet GroupLens (Resnick et al. 1994). La corrélation entre les **utilisateurs** a et i est :

$$w_{cor(a,i)} = \frac{cov(a,i)}{\sigma_a \sigma_i} = \frac{\sum_j (v_{a,j} - \bar{v}_a)(v_{i,j} - \bar{v}_i)}{\sqrt{\sum_j (v_{a,j} - \bar{v}_a)^2 \sum_j (v_{i,j} - \bar{v}_i)^2}}$$

Pour l'exemple d'en haut, ces calculs reviennent au résultats suivant:

U	Item				\bar{v}_j	$w_{cor(1,i)}$
	1	2	3	4		
U_1	5	1	?	2	2,67	
U_2	4	1	1	3	2,67	0,89
U_3	4	2	1	2	2,67	0,97
U_4	1	4	3	2	2,33	-0,89
κ						1/2,75

Figure 1.5: Résultats des calculs pour la réponse utilisateur-utilisateur

Ce qui nous donne une réponse de 1.34 pour $E(v_{1,3})$

voici la démarche:

On peut refaire cette même série de calcul mais au lieu d'utiliser la corrélation Pearson on pourrait utiliser la mesure du cosinus. Cela nous permet d'avoir de

$$\begin{aligned}
E(v_{1,3}) &= \bar{v}_1 + \kappa \sum_{i=2}^n w_{\cos(1,i)} (v_{i,j} - \bar{v}_i) \\
&= 2,67 + \frac{0,89(4-2,67) + 0,97(1-2,67) + -0,89(3-2,33)}{|0,89| + |0,97| + |-0,89|} \\
&= \mathbf{1,32}
\end{aligned}$$

Figure 1.6: Démarche pour utilisateur utilisateur

meilleurs résultats quand nous données tendent vers 0.

$$\begin{aligned}
w_{\cos(1,i)} &= \frac{\sum_j v_{1,j} v_{i,j}}{\sqrt{\sum_{k \in I_1} v_{1,k}^2 \sum_{k \in I_i} v_{i,k}^2}} \\
&= \frac{v_i v_i}{\|v_1\| \|v_i\|}
\end{aligned}$$

Chaque utilisateur représente ainsi un vecteur dans un espace de vote et on cherche celui qui a le vecteur le plus près (parallèle). Cette différente méthode, avec la mesure du cosinus, va nous donner les résultats suivant:

U	Item				\bar{v}_i	$w_{\cos(1,i)}$
	1	2	3	4		
U_1	5	1	?	2	2,67	
U_2	4	1	1	3	2,00	0,97
U_3	4	2	1	2	2,25	0,97
U_4	1	4	3	2	2,50	0,52
κ						1/2,45

Figure 1.7: résultats du cosinus

Le résultats de ces calculs nous donnent un résultats de 1.49 . ce qui en quand même vraiment différent.

1.4.2 K nearest Neighbour

Le principe est qu'on utilise le vote moyen, ou pondéré, des n utilisateurs le splus rapprochés dans l'espace vectoriel des votes. La distance euclidienne est généralement utilisé pour cette fin:

$$d(a, i) = \sqrt{\sum_j (v_{a,j} - \bar{v}_{i,j})^2}$$

Pour chaque item j, on calcule la racin carrée de la somme des carrés des différences entre les votes des utilisateurs a et i. Les utilisateur les plus proches

voisins sont ceux ayant les premières n valeurs. On fait un rankings des utilisateur les plus près de notre utilisateurs ciblé. On détermine alors la valeur du vote de l'utilisateur a par la moyenne, potentiellement pondérée, de ces utilisateurs. Comme on peut voir dans l'exemple ci-dessous, les deux utilisateurs les plus proches de U_1 sont U_2 et U_3 . Leurs distance avec U_1 est de $\sqrt{2}$. En ne conservant que ces deux plus proches voisin, la réponse est de $E(v)_{1,3} = 1.0$, ce qui n'est pas très précis..

U	Item				\bar{v}_i	$d(1,i)$	$w_{\cos(1,i)}$
	1	2	3	4			
U_1	5	1	?	2	2,67	0,00	
U_2	4	1	1	3	2,00	1,41	0,97
U_3	4	2	1	2	2,25	1,41	0,97
U_4	1	4	3	2	2,50	5,00	0,52
κ							1/1,94

Figure 1.8: exemple K nearest Neighbour

On peut continuer notre analyse. La valeur d'une corrélation ou d'un cosinus basée sur un plus grand nombre de votes commun devrait avoir un poids plus important qu'un valeur basée sur un plus petit nombre. Il est donc fréquent de faire la correction suivante au poids $w_{u,v}$:

$$w'_{u,v} = \frac{\max(v_{u,v}, \gamma)}{\gamma} * w_{u,v}$$

Où $v_{u,v}$ est le nombre de votes communs entre les utilisateur u et v et ou gamma est un constante représentant le nombre minimum de votes pour effectuer cette correction, par exemple, Herlocker et coll(1999) utilisent gamma = 5.

1.4.3 item-item

Les algorithmes précédent cherchent des similarités entre utilisateur pour ensuite proposer les items les plus populaires des utilisateur voisins, notamment par une somme pondérée similarité-utilisateur par item pour suggérer les items (équations 1).

L'approche item-item cherche plutôt des similarité entre les items. Dès qu'un utilisateur s'intéresse à un item, on lui suggère des items similaires.

Les similarités dans ce cas ci peuvent être estimés par le cosinus ou la corrélation entre des items, sauf qu'on transpose la matrice pour faire le calcul de la similarité item-item plutôt qu'utilisateur-utilisateur.

Enfin, finalement, on procède de façon analogue, à l'approche utilisateur-utilisateur en utilisant une somme pondérée.

comparément à l'algorithme utilisateur-utilisateur, l'item-item utilise la même méthode, mais au lieu de l'appliquer sur des vecteur d'utilisateur on l'applique sur des vecteurs d'items.

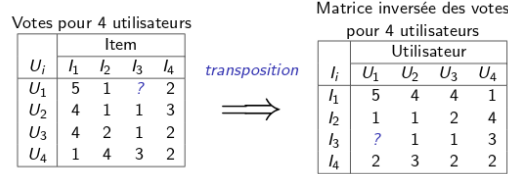


Figure 1.9: transposition de la matrice pour l'item-item

En ne gardant que les items 2 et 4 comme plus proche voisins. On peut déduire la réponse suivante à partir de cette question.

Matrice transposée des votes pour 4 utilisateurs

l_i	Utilisateur				\bar{v}_i	$d(l_3, l_i)$	$w_{cor}(l_3, l_i)$
	1	2	3	4			
l_1	5	4	4	1	3,00	4,69	-1,00
l_2	1	1	2	4	2,33	1,41	0,94
l_3	?	1	1	3	1,67	0,00	1,00
l_4	2	3	2	2	2,33	2,45	-0,50
κ							1/1,44

En ne gardant que l_2 et l_4 comme voisins rapprochés, on prédirait la valeur de $l_{3,1}$ basée sur le principe de l'équation (1) comme suit :

$$l_{3,1} = 1,67 + \frac{0,94(1 - 2,33) + -0,50(2 - 2,33)}{(|0,94| + |-0,50|)} = 0,92$$

Figure 1.10: probleme items items

On peut faire les mêmes calculs qu'à l'exemple précédent mais en utilisant le cosinus comme poids. Par contre, on obtiendrait un résultat de .78 ce qui n'est pas exactement très bon. En utilisant le cosinus normalisé, on pourrait vraiment avoir une mesure beaucoup plus précise. Pour faire sa, nous avons qu'à soustraire la moyenne des rating par utilisateur dans le calcul du poids du cosinus dans le cas d'un calcul item-item (soustraire rating par item dans le cas utilisateur=utilisateur??). Ce qui nous donnerait maintenant un résultats de 1.62, ce qui est beaucoup mieux.

Pour résumer, c'est bien de normaliser en fonction de ce qu'on compare pour prendre en compte des différences de votages, comme par exemple un utilisateur qui vote très haut comparément à un utilisateur qui vote très bas. (genre une moyenne de vote de 4 versus un moyenne de vote de 1).

Maintenant, ce qui nous intéresse de la méthode item-item c'est de faire une

recommandation d'item à un utilisateur en utilisant des items similaire à ceux dont l'utilisateur s'intéresse. Maintenant supposons un item d'intérêt i pour l'utilisateur u , un approche possible serait

- Trouver les N items le splus similaires à i
- De ces N items, choisir les R items dont on calcule le vote le plus élevé pour u .

1.5 TF-IDF

On peut utiliser la transformation TF-IDF pour la recherche d'information. On peut conclure que la similarité de votes pour un item comportant un grand nombre de votes n'a pas le même poids qu'un vote similaire pour un item comportant très peu de votes. Comme par exemple, Vu que StarWars à beaucoup de bons vote de tout le monde. Sa veut dire que les gens vont naturellement trouver le film bon, alors le recommander n'a pas un impact très intense. Par contre, un film ayant peut de vote mais qui est quand même bien rater pour un utilisateur serait un bien meilleur choix de recommandation.

On utilise la transformation TF-IDF pour prendre en compte ces différences:

Pour l'item j , son poid w_j et son nouveau poids pondéré, w'_j

$$w'_j = w_j \log\left(\frac{n}{n_j}\right)$$

Ou n est le nombre total d'individus et n_j est le nombre d'invidus qui ont exprimé un vote pour l'item spécifier.

1.6 Mesure de Performance

Pour mesurer la performance de nos algorithme, on mesure normalement l'erreur entre la valeur prédit et la valeur réelle. La manière dont on mesure l'erreur peut varier, on peut utiliser l'erreur au carré, l'erreur absolue, la racine de l'erreur au carré, c'est très flexible.

La moyenne des erreurs au carré ou plus communément, l'erreur quadratique:

$$\sqrt{\frac{\sum_i^N (\bar{x}_i - x_i)^2}{N}}$$

ou par exemple l'erreur absolue moyenne:

$$\frac{\sum_i^N |(\bar{x}_i - x_i)|}{N}$$

Une autre mesure très utile est d'utiliser le **Receiver operator Characteristic**. Entre autre, c'est de mesure le taux de vrais positif versus faux positif, vrais négatifs versus faux négatif. Vice versa. On mesure l'évolution du taux en fonction de notre algorithme.

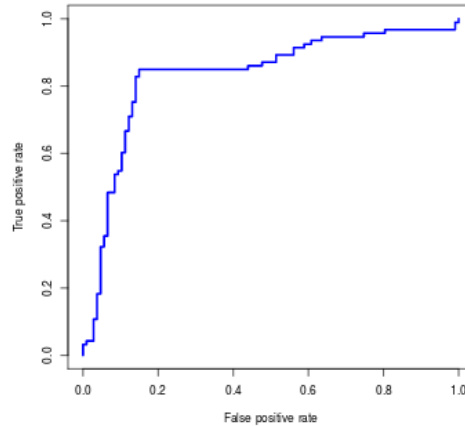


Figure 1.11: Receiver Operator Characteristic

On essaie normalement de maximiser l'aire sous la courbe, ou : **AUC, Area Under the Curve**

Lors de simulations, on utilise la validation croisée. Le principe de la validation croisée est de séparer nos données en un ensemble d'entraînement et un ensemble de validation (de test). Ce qu'on fait:

- On retire des données de notre ensemble de données original, et on les conserve pour des tests
- On cible les valeurs qu'on à enlever et on prédit les votes avec notre algorithme
- On compare les votes qu'on à prédit avec les vrais votes qu'on à sauver

Étant donnée la stochasticité de cette méthode, on essaie généralement de répéter l'expérience plusieurs fois en raison de l'impact que l'enlèvement de certaines données peuvent avoir sur le calculs des votes prédits. On peut aussi faire des variations sur notre méthode de séparer nos données. Comme par exemple, on peut seulement en enlever une, tester, la remettre, en enlever une autre, tester, etc.

Aux fins de comparaison, on élimine du bloc de validation et d'entraînement de l'individu cible, et on rapporte ensuite le score moyens.

Chapter 2

Méthode contenue, PageRank et un peu de Bayésien

Les méthode collaborative utilisent des mesures directe et indirecte de l'**intérêt** pour mesure des similarités entre des utilisateurs/items/peu importe. Cependant, les approche contenus utilisent des attributs de ces objets et utilisateur pour les comparés. On utilise communément la **sémantique démographique** ou autre attributs associés aux items ou aux utilisateurs afin d'établir la similarité. Les approche contenus ont plusieurs d'affinités avec les technique de recherche d'informations.

Normalement, les approches items-items et utilisateurs-utilisateurs sont dites **basés mémoire**. Leur ordre de complexités est de l'ordre de $O(mn^2)$ ou n est le nombre d'utilisateur et m le nombre d'items. (pour l'approche item-item, l'inverse pour u-u). Pour des applications où le temps de réponse est critique et où la Base de Données est grande, le calcul basé mémoire doit s'effectuer au préalable et le résultat stocké en mémoire pour une consultation rapide. Par contre, si l'information change souvent, ce qui est le cas généralement pour les utilisateur ayant peu d'historique, l'approche n'est pas très idéale. C'est pourquoi une approche **basé modèle**, peut être une bien meilleure idée. Le modèle probabiliste est un très bon candidats.

2.1 Modèles Probabiliste

Le principe d'un modèle probabiliste est d'utiliser des facteurs propres aux items et/ou aux utilisateurs. Soit:

- utilisateurs:

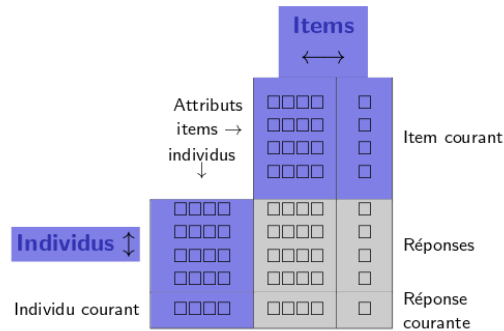


Figure 2.1: Approche contenu

- âge
- profession
- sexe
- code postal (position géographique)
- etc.
- items
 - auteur
 - mots-clés
 - année de production
 - etc.

Quand notre base de données est assez grande, certains items s'adressent à des populations spécifiques et ils sont reconnaissables par leur attributs de contenu. Les votes les plus probable peuvent s'exprimer sous la forme de **probabilité conditionnelle**. qui veut dire, la probabilité du vote v étant donnée les attribut utilisateur et items:

$$P(v_{ij} = x | A_{ik} = 'ing') = \frac{P(v_{ij}=x, A_{ik}='ing')}{P(A_{ik}='ing')}$$

ou par exemple, le film v_{ij} est le vote pour le film j de l'utilisateur i et A_{ik} est l'attribut profession de l'utilisateur i .

2.1.1 Correction de Laplace

Pour ces calculs, on peut aussi rajouter la base de ratio de fréquence. Car lorsque notre base de donnée est très petite, nos ratios peuvent donner des résultats un peu étrange, on utilisera donc la **correction de Laplace** pour pouvoir corriger ces inégalités..

$$\frac{x_1+1}{(x_1+x_2)+2}$$

Cette correction intègre une notion de probabilité antérieure et évite les biais des petits échantillons de même que les ratios singulier de 1/1, 0/1, 0/0.

2.1.2 Exemple

Maintenant, supposons le vote d'un homme ingénieur:

$$P(v_{ij} = x | A_{ik1} = 'ing', A_{ik2} = 'H') = \frac{P(v_{ij}=x, A_{ik1}='ing', A_{ik2}='H')}{P(A_{ik1}='ing', A_{ik2}='H')}$$

Le problème avec ce calcul est que la combinaison de facteurs divise les fréquences à chaque facteur que l'on rajoute. On peut rapidement trouver avec aucun cas.

Solution: On postule l'indépendance des facteurs

$$P(v_{ij} | A_{ik1}, A_{ik2}) = P(v_{ij})P(A_{ik1} | v_{ij})P(A_{ik2} | v_{ij})$$

de manière générale..

$$P(C | F_1, \dots, F_n) = \frac{1}{Z} * p(C) \prod_{i=1}^n p(F_i | C)$$

ou C est une classe et F sont des attributs. Z est une valeur qui ne dépend pas de C, de sorte que lorsqu'il s'agit d'estimer la class la plus probable, on peut l'ignorer.

2.2 Autres Approches probabiliste

Plusieurs modèles ont été proposés pour fournir une estimation probabiliste des votes. Par exemple, on peut utiliser l'espérance en fonctions des items pour estimer le votes d'un utilisateur.

La valeur attendue d'un vote (sur une échelle ordonnée) est la somme pondérée de la probabilité de chaque valeur par sa probabilité:

$$E(v) = \sum_i P(v = i) i$$

La formulation de cette équation dans le contexte d'estimer le vote de l'utilisateur actif, a pour un item j est :

$$E(v_{a,j}) = \sum_i P(v_{a,j} = i | v_{a,k}, k \in I_a) i$$

c-à-d. que $E(v_{a,j})$ est la somme pondérée des probabilités de chaque vote étant donnée les votes aux autres items pour l'utilisateur actif.

2.3 Modèle Par Classe (Clustering)

Une première approche repose sur la définition de classe d'utilisateur. Ces classes sont définies en fonction de leur capacité à discriminer les préférences entre les utilisateurs.

$$P(C = c | v_1, \dots, v_n) = P(C = c) \prod_i P(v_i | C = c)$$

Ensuite, on calcule les probabilités qu'un utilisateur appartienne à chaque classe respective à la lumière de ses votes et on utilise ses probabilité dans l'équation (1). Cette approche nécessite le recours à l'algorithme EM pour obtenir probabilités et de déterminer un nombre adéquat de classes par des techniques d'exploration statistique.

Une autre approche probabiliste serait de dériver des probabilités conditionnelle entre les noeuds eux-mêmes. Breese et Coll. utilisent par exemple un arbre de décision comme celui ci-dessous. Les noeuds terminaux représentent la probabilité de visionner 'Melrose Place' conditionnellement à avoir visionné les noeuds parents.

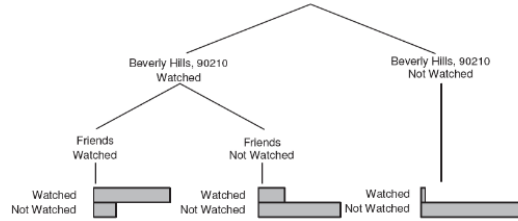


Figure 2.2: Melrose Place

Aussi, plutôt que de faire un arbre de décision, on pourrait aussi faire un calcul de probabilité postérieur:

$$P(v_i | v_j \in S) = \prod_{j \in S} P(v_j | v_i)$$

Il faut faire une sélection de noeuds pertinents, $v_j \in S$, puis établir les valeurs de $P(v_j | v_i)$ à partir des données.

Un test d'indépendance entre v_i et tous les autres noeuds permet d'effectuer une première sélection de façon simple. Cette approche est semblable aux réseaux bayésien, à la différence qu'on ne crée que des réseaux bayésiens dits naïfs.

Les calculs reposent sur une hypothèse d'indépendance des votes étant donnée l'appartenance à une classe, ou l'indépendance conditionnelle des votes entre eux. Dans le cas de l'approche des classes, le calcul nécessite un entraînement avec des données et repose sur

- un algorithme pour définir les classes optimales
- estimer les paramètres du modèle. qui peut être assez difficile.

On peut aussi utiliser une approche graphique, Ces approches indiquent les interdépendances entre les votes et les items pour inférer la probabilité d'un vote à un item donné. Breese et al. propose une approche basée sur les modèles bayésiens tandis que Heckerman et al. suggèrent une approche basée sur un modèle de réseaux de dépendance. Les deux approches sont comparables en termes d'acuité et performance de 2% à 4 % mieux que les méthodes d'espace vectoriels dans les expériences rapportées. L'approche de réseaux de dépendance est cependant moins gourmande en ressources mémoire et de calcul.

2.4 Arbre de décision

- Avantages:
 - Ces modèles sont plus précis pour prédire les votes
 - Plus rapides dans la mesure où l'on peut créer un modèle avec des paramètres plutôt que de prendre une approche où l'on doit faire des calculs basés sur l'ensemble des données
- Désavantages
 - Déploiement beaucoup plus complexe
 - Nécessite une maîtrise des techniques de modélisation bayésienne et d'estimation de paramètres relativement sophistiquée

Comme discutés précédemment, il est aussi possible d'utiliser la fréquence inverse des votes pour pouvoir donner plus de poids aux votes moins fréquents en modulant le poids par l'équivalent de la fréquence inverse des documents (TF-IDF)

Une seconde transformation du vote consisterait à pénaliser les votes qui sont plus faibles pour compenser le fait que les utilisateurs ont moins tendance à voter pour les items qu'ils n'aiment pas.

2.4.1 Breese et Coll.

L'utilisation de la fréquence inverse utilisateur améliore les résultats mais de façon variable. On peut voir une amélioration de 1 à 3 % pour la corrélation et le cosinus pour la prédiction de recommandations. et de 3 à 25 % pour la

Résultats moyens des recommandations en utilisant 5 votes, la mesure d'utilité et 10 simulations.

	Neilsen	MSWeb	EachMovie
Algorithme			
BN	42,2	59,8	42,3
CR+	43,3	58,9	42,1
VSIM	40,0	56,1	36,7
BC	18,9	54,8	33,2
POP	19,5	46,9	28,9
Rd	1,8	1,8	0,8

Figure 2.3: Résultats de l'expérience breese et col.

prédiction d'un vote individuel, plus grande amélioration pour le cosinus que la corrélation, par contre, sa ne s'applique qu'aux données EachMovie.

l'amplification du score améliore aussi les résultats de 1 à 10% mais surtout pour les données de télévision et très peu pour EachMovie.

La taille du corpus d'entraînement influence quand même grandement la performance jusqu'à atteindre un seuil maximale (convergence).

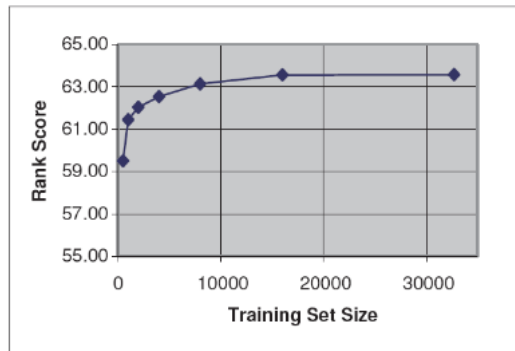


Figure 2.4: corpus d'entraînement pour breese et col.

2.5 PageRank (Popularité)

Par définition, les items plus populaire intéressent plus d'invidivus. Cependant, c'est quand même difficile de représenter la popularité d'un document en fonction d'un nombre et de pouvoir le comparé avec un autres. Par contre, à l'aide de liens entre les documents, on peut mesurés la popularité d'un document à l'aide des liens entre eux. Les références dans le cas du web sont les liens entres

les différents pages.

l'algorithme PageRank, définit par Larry Page dans le début des années de google. est particulièrement bien adapté à la mesure de popularité. Le principe est d'avoir une requête dans un monde W , qui retourne une ensemble de page s correspondant à la requête ciblé, soit par mots-clés, distance euclidienne à partir d'une référence, etc. On peut aussi définir s' comme étant les pages référencés par les pages dans notre domaine initiale s . Selon l'algorithme Pagerank, dans notre figure, (b) serait la page la plus pertinente car elle est la plus référencés de s . Cependant, en faisant l'extension du domaine à s' , on peut voir que la page (a) est bien meilleur. Il existe plusieurs variante possible selon qu'on différencie entre les types de liens ou qu'on ne tien compte que de s et s' par exemple. En plus, la définition de s et s' est complètement au choix de l'architecte du système.

l'algorithme de base de PageRank est la suivante:

$$PageRank(A) = (1 - d) + d \sum_{D_1 \dots D_n} \frac{PageRank(D_i)}{C(D_i)}$$

Ou:

- D_n sont les page qui réfèrent à A (liens entrant)
- $C(D_i)$ est le nombre de référencs de la page i (les liens sortants)
- d est un facieur d'ajustement dans l'intervalle $[0,1]$, notamment pour permettre aux pages sans liens entrants d'avoir une cote et pour faciliter la convergence pour les calculs des pages référencés par de telles page. Une valeur de base pourrait ête de 0.85

Cet algorithme est itératif et les valeurs convergent après quelques itérations (souvent en dessous de 10). On peut initialiser les PageRank et ajouter 1 à $C(D)$ pour éviter des division par 0.

Le modèle calcule la probabilité que l'on accède à une page en suivant les liens de façons aléatoire. Le facteur d'atténuation d représente la probabilité qu'un individu continu d'accéder à des pages; donc $(1-d)$ est la probabilité qu'il arrête.

Pour que la somme des PageRank donne 1, on divise le facteur d'atténuation par N , le nombre total de pages:

$$PageRank(A) = \frac{(1-d)}{N} + d \sum_{D_1 \dots D_n} \frac{PageRank(D_i)}{C(D_i)}$$

Si on considère le réseau de liens comme un chaîne de Markov ou les pages sont des états, le PageRank représente en fait la probabilité qu'un individu soit sur une page donnée par une navigation aléatoire.

2.5.1 Calcul de manière vectoriel

Pour faire le calcul de manière vectoriel, voici la meilleur manière de calculer le Pagerank:

soit:

- r : vecteur des valeurs de Pagerank (initialiser à 1)
- s : vecteur du nombre de liens sortant
- A : matrice d'adjacances (les références)
- d : le facteur d'amortissement
- d : un vecteur d'amortissement de longueur n contenant
- n : le nombre de document

La valeur de R , qui est le page de chaque articles, serait calculer de la manière suivante:

$$r = \frac{(1-d)}{n} + dA\left(\frac{r}{s}\right)$$

2.5.2 Exemple

En supposant une valeur d de 0.85, et en initialisant le PR de chaque page à 1, les valeurs des PR à chaque itération sont alors:

l'algorithme PageRank à révolutionner la recherche d'information sur le web, un des raisons pourrait être expliquer en raison du fait que Pagerank chercher à vraiment modéliser l'interactions entres les différents articles en fonctions de comment les articles interagissent entres-elles. Sa fait en sorte qu'on exprime d'une manière beaucoup plus intime les relations entre les différents articles. C'est pourquoi l'approche Pagerank est très performante et peut rivaliser les approches d'espaces vectorielles et probabiliste.

l'algorithme Pagerank peut être utiliser pour recommander des articles. Premièrement, on peut cibler un article spécifique sur lequel on veut trouver des articles voisins et faire des recommandations d'article qui ressemble à cet article cibler. On peut faire sa en spécifiant un domaine S qui prendrait toutes les articles auquel l'article en question fait références. On calculerait ensuite le PageRank pour toute ces articles, et ensuite on prendrait les articles ayant les indices Pagerank les plus haut afin de les recommandés.

2.5.3 Puissance des Matrices

Un aspect très intéressant est la relation entre les différentes puissance de matrice d'adjacance. en représentant un graphe sous la forme d'une matrice

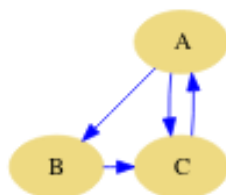
<i>Itération</i>	A	B	C
0	1,000	1,000	1,000
1	0,900	0,475	1,325
2	1.176	0.432	0.836
3	0.760	0.549	0.917
4	0.829	0.373	0.840
5	0.764	0.402	0.720
6	0.662	0.374	0.717
7	0.659	0.331	0.650
8	0.602	0.330	0.612
9	0.570	0.306	0.586
10	0.548	0.292	0.552
...	...		
30	0.393	0.217	0.403

Figure 2.5: Valeurs de PageRank pour chaque itérations de notre graphe ci-dessous

d'adjacance (ou de transisiton). Il est possible de déterminer les chemins transi-tifs entre les noeuds par la puissance de la matrice. ainsi, supposons un matrice de transition M entre quatre noeuds (a,b,c,d) et ou $(i,j) = 1$ indique l'existence d'un lien $i \rightarrow j$, les graphiques peuvent être représenter comme ceux-ci, avec les valeurs de matrice de a à c étant de deux pour la deuxième, qui compte les liens entre c et a:

2.6 Rappel Théorie Bayésienne

Voir section des notes d'INF8215



Le graphe ci-dessus
correspond à la matrice
d'adjacence :

	a	b	c
a	0	0	1
b	1	0	0
c	1	1	0

Figure 2.6: Notre matrice d'adjacence avec graphe

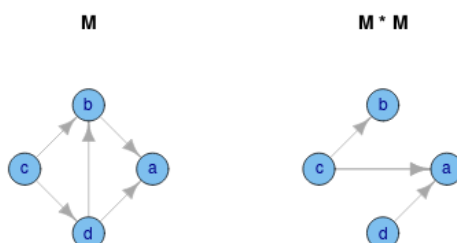


Figure 2.7: Démonstration d'une deuxième puissance d'une matrice de transition/adjacence

Chapter 3

Réduction de Dimensions

Normalement, nos systèmes essaie de faire une recommandations basé sur une multitude de films (1000+). Quoique précise, ces recommandations nécessite un nombre assez élevés de cycles CPU pour calculer. Une des méthodes que l'on propose pour réduire l'intensité des calculs est d'effectuer une réduction de dimension pour nos données.

3.1 Intuition

Le principe/intuition derrière la réduction de dimensions est de 'séparé' les films selon certaines dimensions. Par exemple, on peut regrouper les films comme étant des films d'horreur, des films d'actions, des films romantique.. etc. Par contre, chaque films est unique, ce qui veut dire que certains film peut être plus un films d'horreur que d'autre mais qui ont une aspect un peu plus films d'actions. Ce qu'on essaie de faire, c'est de regrouper les films selon une forme de catégorisation comme par exemple les genres, sauf que ce n'est pas selon les genres que nous allons les catégoriser mais biens une dimensions qui représentent plusieurs facteurs différents, que nous ne savons pas nécessairement.

Dans le cas de nos données, On représente le vote de chaque utilisateur par rapport à chaque items. Ce que nous pouvons essayer de faire, c'est de conserver le nombre d'utilisateur, mais de réduire le nombre de dimensions d'items, pour avoir par exemple, des dimensions de genre, et ensuite reconstruire la matrice pour avoir des prédictions de toutes les votes pour chaque films en fonction des relations entre les films et les dimensions.

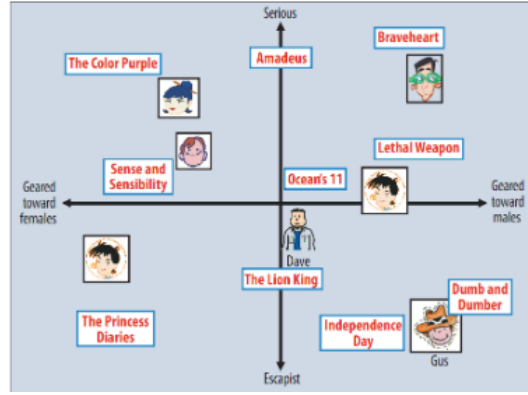


Figure 3.1: Représentation Graphique des films rapport à certaines dimension

3.2 Techniques de Décomposition Matricielle

Plusieurs techniques sont déjà disponible pour la décomposition de matrice. Normalement, on factorise à partir des vecteurs propres de MM^T ce qui entraîne l'orthogonalité. (analyse en composante principales **PCA**). On peut résumer le principe comme:

$$M_{m \times n} = Q_{m \times l} P_{l \times n}^T$$

Ou m et n sont nos dimensions, l est le nombre de dimensions latente que nous souhaitons utiliser pour faire notre décomposition. Cette factorisation produit une matrice dense, de laquelle des calculs de similarités comme faite auparavant peuvent être utiliser pour faire des recommandations. Cette factorisation représente le produit Q et P et consitue une estimation (prédiction) de nos votes M.

Maintenant, supposons une matrice de votes utilisateurs-items M, il s'agit alors de la factoriser en 2+ matrice Q et P:

$$M = QP^T$$

Cette approche permet d'identifier des facteurs latents, qui sont en fait les dimensions sur laquelle nous allons décomposer nos matrices. Les facteurs se combinent linéairement pour augmenter et diminuer le vote (somme). Dans ce calcul, Q représente la préférence des utilisateurs pour les facteurs et P représente l'appartenance des items aux facteurs. Cette méthode a donné les meilleurs résultats pour le prix Netflix en 2009. Il y a beaucoup de manière de créer les matrices Q et P. Comme par exemple la SVD ou la Décomposition en valeurs positive. Il existe plusieurs différents algorithmes qui peuvent converger vers une solution.

3.3 SVD

Cette méthode est déjà bien connue en recherche d'information. Cette méthode se décompose de la façon suivant:

$$M = U\Sigma V^T$$

ou:

- U: matrice orthonormale qui représente les vecteurs propres de MM^T
- V: matrice orthonormale qui représente les vecteurs propres de M^TM
- Σ : matrice diagonale qui représente les valeurs singulières de M (les racines des valeurs propres des vecteurs propres de MM^T et M^TM)

Les valeurs singulières de Σ sont ordonnées et indique l'amplitude relative de chaque vecteur propre. On utilise normalement les n premières valeurs singulières de Σ pour effectuer une réduction de dimensions ou n représente les facteurs latents. C'est ce qu'on nomme la **LSA** ou Latent Semantic Analysis. Voici un exemple un peu plus graphique de la décomposition SVD:

Recall the SVD form of A:

$$\begin{matrix} A & = & U & \Sigma & V^T \\ m \times n & & m \times m & m \times n & n \times n \end{matrix} \quad (1)$$

$$= \left(\begin{array}{c|c|c|c} \mathbf{u}_1 & \mathbf{u}_r & \mathbf{u}_{r+1} & \mathbf{u}_n \\ \hline & \dots & & \\ \hline & & & \end{array} \right) \left(\begin{array}{ccc} \sigma_1 & & 0 \\ & \ddots & \\ & & \sigma_r & & 0 \\ & & & \ddots & \\ 0 & & & & 0 \end{array} \right) \left(\begin{array}{c} \text{---} \mathbf{v}_1^T \\ \vdots \\ \text{---} \mathbf{v}_r^T \\ \vdots \\ \text{---} \mathbf{v}_{r+1}^T \\ \vdots \\ \text{---} \mathbf{v}_n^T \end{array} \right) \begin{matrix} \left. \vphantom{\begin{matrix} \vdots \\ \vdots \end{matrix}} \right\} \text{row}(A) \\ \left. \vphantom{\begin{matrix} \vdots \\ \vdots \end{matrix}} \right\} \text{null}(A) \end{matrix}$$

$\underbrace{\hspace{10em}}_{\text{col}(A)} \quad \underbrace{\hspace{10em}}_{\text{null}(A^T)}$

Figure 3.2: Exemple visuel de la décomposition SVD

Exemple dans le domaine de la RI Le domaine de la recherche d'information partage plusieurs techniques avec les systèmes de recommandations (en fait, ces techniques sont directement utilisables pour les approches contenues). La matrice de recherche d'information a normalement une allure comme suit:

Initialement, notre corrélations initiales des paires de terme était de -0.38 pour human-user ainsi que -0.29 pour human-minors. C'est mot veut dirent

$$\begin{matrix} & \text{documents} \\ \text{termes} & \begin{pmatrix} a_{11} & \dots & a_{1n} \\ \vdots & \ddots & \vdots \\ a_{m1} & \dots & a_{mn} \end{pmatrix} \end{matrix}$$

Figure 3.3: allure d'une matrice de recherche d'information

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

Figure 3.4: Exemple de contenu

	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

Figure 3.5: Exemple de Matrice de termes

presque la même chose, mais vu qu'il y a beaucoup de valeurs manquante la corrélation ne se fait pas bien. Par contre, en réduisant la dimensions, on ne conserve que les deux premières valeurs singulières (sachant qu'il n'y a que deux sources de documents). La matrice originale est reproduite avec les dimensions réduites à 2, alors toutes les valeurs singulières de la matrice diagonale Σ sont mise à 0.

Après avoir réduit la dimension, nos corrélations pour les deux paires sont -0.94 ainsi que -0.83, ce qui est beaucoup mieux et représentent bien les attentes qu'on a des mots.

Pour résumer, la réduction de dimensions nous permet d'assigner des valeurs sur une échelle continue la ou il n'y avait pas de valeurs. Elle offre donc une solution au problème des matrices creuse. Et parfois, elle assigne des valeurs très justes des appartenances des variables aux facteurs latents comme l'exemple de Landauer et al. le démontre.

3.4 Factorisation en Valeur non-négative + Valeurs manquantes

D'autres méthodes sont disponibles pour faire la décomposition de matrice. Une des autres méthodes très connues est la factorisation en valeurs non-négatives. elle va comme suit:

$$R_{u*i} = W_{u*l} H_{l*i}$$

On fait une factorisation en deux matrices où l représente les facteurs latents. On applique une contrainte de valeurs non-négatives. Dans certains contextes, les valeurs négatives ne sont pas naturelles. Cette approche nous permet d'avoir des solutions multiples.

Normalement, pour trouver W et H on se sert d'une recherche par gradient (descente de gradient). Cette approche est non-déterministe alors ça peut prendre quand même longtemps converger. Par contre, on doit spécifier le nombre de dimension latentes k sur lesquelles nous allons construire les matrices.

Pour l'ensemble des votes communs, k , on cherche les valeurs de W et H qui minimisent les équations suivantes:

$$\min_{p,q} \sum_{(u,i) \in k} (r_{ui} - h_i^T w_u)^2 + \lambda (\|h_i\|^2 + \|w_u\|^2)$$

où le facteur avec le λ est le facteur de régularisation qui limite les valeurs de p et q et que l'on contrôle par λ .

3.5 Biais

Une proportion importante de la variance des votes s'explique par:

- le biais utilisateur: la propension d'un individu à des votes faibles ou forts
- le biais item: la popularité ou l'impopularité d'un item

Il faut donc inclure ces termes dans l'équation objective:

$$r_{ui} = \mu + b_i + b_u + h_i^T q_u$$

On peut ajouter beaucoup d'autres choses à l'équation, comme par exemple:

- Votes implicites (traces d'utilisateurs)
- Facteurs Démographique

Certains termes sont en fonction du temps.. comme par exemple les biais qui peuvent changer avec le temps. Ou par exemple un vote qui peut avoir plus d'importance (TF-IDF??).

typiquement, les matrices de votes sont creuse. On peut proposer deux solution:

- Optimiser l'erreur quadratique avec la descente de gradient
- Estimer les valeurs manquantes afin de créer une matrice dense.

Chapter 4

Modèle de Connaissance de l'utilisateur

Comme on a vu dans l'intro, trois types d'information sont nécessaires pour une bonne modélisation/personnalisation d'un système de recommandations, soit:

- **Intérêts:** On les retrouve plus ou moins implicitement dans les modèles de filtres d'informations. Les préférences, les votes, les comportements constituent en fait, d'un modèle utilisateur
- **Connaissances:** très important pour les systèmes d'aide, les guides d'apprentissage et les systèmes tuteurs
- **Buts:** Dans une certaine mesure, les intentions et les buts recherchés par l'utilisateur constituent aussi un modèle utilisateur, mais il est très dynamique, Il peut varier d'un instant à l'autre. On le verra avec les travaux sur la reconnaissance de plans.

Dans cette section, on se concentrera beaucoup plus sur les modèles de la connaissance.

Les modèles de connaissances d'un utilisateur sont très importants dans plusieurs sortes de systèmes de recommandations, soit:

- Guide d'étude: indique les matières sur lesquelles concentrer les efforts en vue d'un examen ou dans le cadre d'un cours.
- Tuteur: fournit un encadrement intelligent de matériel et exercices adaptés aux réussites/échecs d'un étudiant
- Exerciceur: adapte la difficulté des exercices/items au niveau de l'apprenant

Par exemple, on avait un système à la poly qui s'appelait Agata Z050, ce système évaluait les compétences en mathématiques des appliquants à la Polytechnique.

On peut modéliser la connaissance de plusieurs différentes façon:

- **Stéréotypes:** l'utilisateur est classifié selon un stéréotype, une classe comme expert, novice, intermdiate.. Ce modèle à l'avantage de la simplicité et peut facilement être entraîné avec des méthodes probabiliste.
- **Hiérarchique :** Les connaissances sont organisées sous la forme d'une hiérarchie. La connaissance d'un noeud dépend de la connaissance de ses enfants.
- **Par item:** Seuls des items observables sont utilisés pour modéliser la connaissances. De ces items, on peut déduire la connaissances de concepts non observable.

On peut aussi distinguer chaque modèle selon s'il comporte des paramètres latents ou observables. et qu'ils sont construits par l'ingénierie de la connaissance ou par l'apprentissage automatique.

On peut utiliser d'autre dimensions à ce calcul, comme par exemple on peut utilisé des habiletés latentes ou observables. Les concepts sont des habiletés latentes car ils ne sont jamais directement observés (genre trigo vs calcul intégrale) mais plutôt induits à partir d'indices observables comme les réponse à des questions (items de tests).

On fait l'estimation des paramètres par apprentissage par ingénierie. L'estimation des paramètres par apprentissage avec des données empiriques permet l'automatisation et évite la subjectivité d'un modèle dont les paramètres sont déterminés par un expert du domaine.

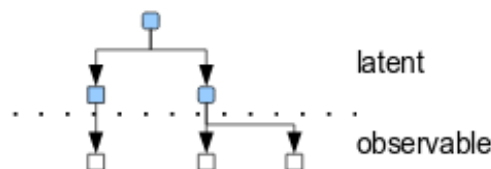


Figure 4.1: Comparaison graphique des facteurs latents vs. Observables.

Les concepts, dans le cadre d'un apprentissage plus scolaires. sont des paramètres latents puisqu'il ne sont jamais (??). Dans la représentation graphique de la fig4.1, les noeuds latents sont des concepts dans une structure hiérarchique

	Resultats	Sujet X	Sujet Y	Sujet Z	
Question 1	1	1	0	0	} Q-matrice
Question 2	0	1	0	0	
Question 3	1	0	1	0	
Question 4	1	0	0	1	
Question 5	1	0	0	1	
Question 6	0	0	0	1	

Score par sujet
 Sujet X : 1/2 — Sujet Y : 1/1 — Sujet Z : 2/3 \Rightarrow
 $\begin{bmatrix} 1 & 1 & 2 \\ 2 & 1 & 3 \end{bmatrix}^T = \mathbf{Q}^T [101110]^T$
 (1) $\mathbf{S} = \mathbf{Q}^T \cdot \mathbf{R}$
 (Les compétences, \mathbf{S} , sont le produit scalaire de \mathbf{Q} et des résultats matrix \mathbf{R} —si on normalise)
 (2) $\mathbf{R} = \mathbf{Q} \cdot \mathbf{S}$

Figure 4.2: Exemple d'une Q-matrix

dont les noeuds terminaux sont les items observables. (question/items). Pour résumer, les questions ont une appartenant à un certain concepts qu'on essaie de mesurer.

4.1 Factorisation des matrices

Normalement, on essaie de factoriser les matrices pour que les dimensions latentes représentent les différents sujets dans notre systèmes de recommandations et qu'on essaie de projeter nos questions comme ayant des composantes de nos différents sujet. Mais comment feront nous pour nous assurer quels items se rapportent à quel compétences/sujets??

On peut représenter nos questins comme étant une combinaisons de différents sujets. On appelle ce terme Q-matrix (q-matrice). Chaque question est représenter comme un coefficient (un pourcentage?) qui représentent chaque rapport à la question au différents sujet qu'on essaient de modéliser.

On peut ajouter différentes variantes à la q-matrix:

- Elle peut avoir une à **plusieurs compétences par items** (un item comportant 50% trigo, 40% combinatoire ainsi que 10% calcul intégrale..)
- On peut utiliser un **modèle conjonctifs**, c'est à dire que toutes les compétences sont nécessaires au succès (un mélange??)
- un modèle **disjonctif**, ou une seule compétence amène au succès.
- un modèle **compensatoire**, ou chaque compétences contribue linéairement aux chances de succès.

Ce sujet est très actif en recherche et soulève plusieurs questions:

- Le modèle linéaire est-il appropriés ? Est-ce qu'on peut bien modéliser les compétences d'un individu avec une comparaison/combinaison linéaire des scores?
- Un modèle conjonctif est-il plus performant ? Est-ce que sa prend vraiment une combinaison de toutes les sujets pour bien performer??
- Les prédictions sont-elles meilleurs ou non aux autres modèles ? Comparaison entre les différents modèles ?

La factorisation de matrice peut amener une meilleur performance prédictive, mais pas nécessairement se prêter à l'interprétation.

4.2 Utilisation d'un Modèle Bayésien

On peut aussi utiliser des éléments observables comme évidence pour l'appartenance à une classe donnée. Posons comme exemple trois items, X_1, X_2, X_3 pouvant être soit réussi ($X=1$) ou échoué ($X=0$). Posons aussi deux stéréotypes, maître ou apprenti que nous exprimons comme étant respectivement $\theta = 1$ ou $\theta = 0$. Le modèle bayésien consiste à évaluer:

$$P(\theta|X_1 = x_1, X_2 = x_2, X_3 = x_3)$$

ou x_i représente le succès ou l'échec à l'item correspondant. On abrège généralement l'expression précédente par :

$$P(\theta|X_1, X_2, X_3)$$

Maintenant, on peut poser comme énoncé que nos trois questions (X_i) sont des événements binomiales, c-à-d. qu'ils peuvent prendre des valeurs soit de 1 ou de 0 étant donné une réussite ou un échec respectivement. On peut représenter la probabilité d'appartenir à une classe (maître = 1, apprenti = 0) en fonction de la fréquences f d'occurrence des classes comparé aux résultats obtenus dans les trois différentes questions, qui sont toutes les différentes combinaisons binomiales des résultats de questions. Avec la **correction de Laplace**, on peut calculer la probabilité d'appartenir à une classe θ_j avec la formule suivante:

$$P(\theta_j) = \frac{f(\theta_j=1)+1}{(\sum_v f(\theta_j=v))+2}$$

en utilisant le tableau ci-haut, on peut dériver la probabilité d'appartenir à la classe $\theta = 1$ pour quelqu'un ayant eu une bonne réponse seulement qu'à la question 2:

$$P(\theta_3) = P(\theta|X_1 = 0, X_2 = 1, X_3 = 0) = \frac{8+1}{25+8+2} = 0.26$$

4.2.1 Problèmes Combinatoire

Par contre, cette méthode peut devenir beaucoup plus compliqués à mesure que sa avance. Dans notre exemple, nous avons trois question et deux classes... Pour un exemple plus réaliste ayant 10 questions, sa nous donne environ 2^{10} lignes (1024 entrées différentes) à calibrer et il nous faudrait des dizaines de milliers d'utilisateurs pour avoir une exemple pour chacun de ses différents cas, ce qui peut être quand même difficile... Il devient donc rapidement impossible de calibrer les probabilités conditionnelles.

4.3 Modèles d'espaces (sous-ensembles)

n'est pas à l'examen.. voir slides

4.3.1 Inférence et choix de Question

n'est pas à l'examen.. voir slides

4.4 Autres Modèles

Pour répondre à ce problème, certains autres modèles peuvent être utiliser pour rendre les choses beaucoup moins compliquer:

- Modèles de réseaux de Bayes: On peut aussi modéliser les liens entre concepts eux-mêmes et entre les concepts et les items par différentes simplifications comme les 'noisy OR-gates' ou tout autre structure probabiliste qui a une signification:

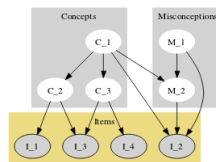


Figure 4.3: Exemple d'un réseaux de Bayes pour modéliser la connaissance

- Modèles à base de règles: Un autre modèle qui utilise des règles our établir des diagnostics précis et analyser des comportements plus complexes. Il peut être combinés à un modèle probabiliste.
- Modèles simple: Finalement, on peut aussi utiliser une matrice et une simple formule de sommation pour indiquer le niveau de maîtrise de différents concepts à l'instar d'un examen.

Les modèles graphiques bayésiens sont très répandus dans le domaine de la modélisation des connaissances des apprenants. Ils sont particulièrement utilisé pour modéliser les concepts entres eux et leur relations. Vous pouvez voir à la fig 4.3 un exemple d'un réseaux qui modélise des questions en fonction des concepts et des méconceptions classique.

Avec les réseaux bayésiens, on peut faire de l'inférence. Certaines structure comme les arbres se prête à un calcul exact. Des structure libres doivent recourir à des algorithmes d'expectation-maximisation (EM) ou plus communément, maximum-likelihood. Ce qui peut être très couteux.

ces algorithmes peuvent prendre plusieurs formes. voici certaines formes communes avec les grandes lignes:

- En commençant par les racines, on assigne des valeurs aléatoire mais selon les probabilités initiales
- On propages les assignations vers les noeuds enfants en respectant les valeurs assignées aux noeuds parents.
- On créé ainsi plusieurs échantillons à partir desquels on recalcule les probabilités conditionnelles à partir des assignations aléatoires.
- Les nouvelles probablités sont réutilisées pour créer de nouveaux échantillons.

Souvent, on utilise un modèle *ad hoc* qui simplifie les calculs mais ne respecte pas la sémantique des inférences bayésiennes.

4.5 Régression Logistique

Présumons que le succès d'un individu à répondre correctement à une série de question $(X_1, X_2, X_3, \dots, X_n)$ dépend de son habileté, θ qui serait un nombre continue dans le spectre réel. La probabilité d'observer une série spécifique de réponse, $(x_1, x_2, x_3, \dots, x_n)$ est donc:

$$P(x_1, x_2, \dots, x_n | \theta) = \prod_i P(x_i | \theta)$$

Le problème consiste donc à trouver la valeur de θ qui maximisera cette probabilité. Il nous faut un modèle de $P(X_i | \theta)$ comme celui de la figure ci-contre, qui correspond à la figure ci-dessous, qui est en fait la **fonction sigmoïde**:

$$P(X_i | \theta) = \frac{1}{1 + e^{-\theta}}$$

Prenons comme exemple un individu qui obtient ces réponse pour 7 questions étant totalement indépendantes:

$$(0, 1, 1, 0, 0, 0, 0)$$

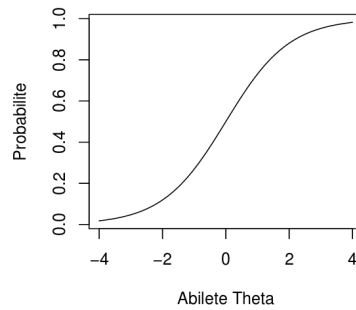


Figure 4.4: Allure de la fonction sigmoïde

La probabilité d'observer cette séquence s'il avait une habileté $\theta = 1$ serait proportionnelle à :

$$P(X_i = 1|\theta = 1) = \frac{1}{1+e^{-1}} = 0.731$$

donc, pour deux événements qui sont égaux à 1 et 5 autres qui sont égaux à 0...

$$0.731^2 * (1 - 0.731)^5 = 0.001$$

Sont les chances qu'un individu obtienne ces résultats étant donné son habileté $\theta = 1$. On peut recommencer l'expérience avec un différent $\theta = -1$ qui nous donnerait une probabilité de 0.015. on peut faire un graphique avec les différentes valeurs de theta possible et prendre le maximum comme étant l'habileté la plus probable pour notre individu.

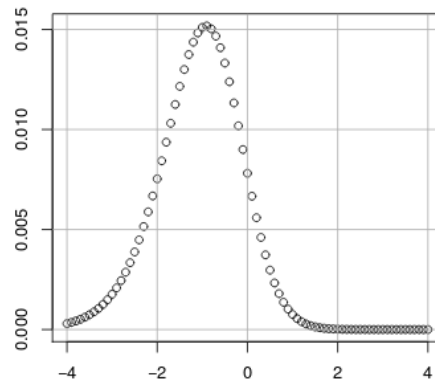


Figure 4.5: Habiletés différentes pour notre individu

4.5.1 Modèles TRI

Le modèle le plus répandus pour modéliser les réponses à des items d'un test est celui de la Théorie des Réponses aux Items, abrégés **TRI**. Ce modèle s'inspire de la fonction sigmoïde mais il comprend deux paramètres supplémentaires:

- a : la discrimination d'un item qui détermine la pente de notre fonction sigmoïde.
- b : la difficulté d'un item qui détermine l'abscisse

La fonction résultante est celle de la réponse caractéristique à un item (Item characteristic Curve - ICC):

$$P(X_i|\theta) = \frac{1}{1+e^{-a_i(\theta-b_i)}}$$

Ces fonctions donne comme résultats finaux:

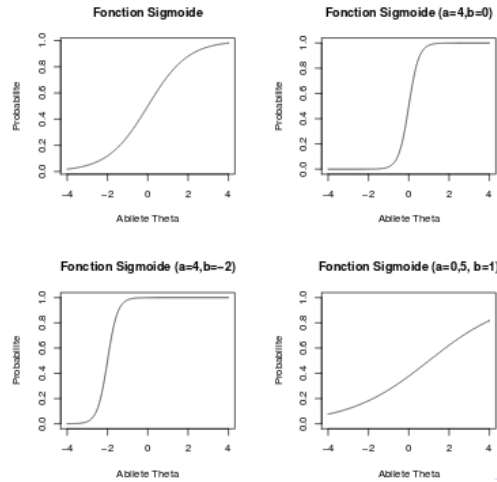


Figure 4.6: Différentes fonctions TRI