Scénario: Recommandation par Factorisation ou Complétion

Résumé

Mise en œuvre dans R de la NMF ou factorisation de matrice non négative sur un exemple jouet de système de recommandation par filtrage collaboratif. Comparaison avec la SVD et la complétion de matrice.

1 Introduction

1.1 Système de recommandation

Tous les sites marchands mettent en place des systèmes de recommandation pour déterminer les produits les plus susceptibles d'intéresser les internautes / clients en visite. Lorsque ceux-ci sont basées sur les seules informations concernant les interactions clients $\times$ produits, ils sont nommés *filtrage collaboratif*. Parmi ces derniers, les systèmes les plus aboutis sont basés sur la recherche d'un modèle de quelques *facteurs latents* susceptibles d'expliquer en faible dimension les interactions entre clients et produits.

D'autres systèmes sont basés sur des connaissances clients (*user-based*) ou sur des connaissances produits (*item-based*)ou encore sur des approches mixtes. Ils ne sont pas abordés ici.

Les données se mettent sous la forme d'une matrice $\mathbf{X}$, toujours très creuse, contenant pour chaque client i (ligne) le nombre d'achats du produit j (colonne) ou une note d'appréciation de 1 à 5 lorsqu'il s'agit de films (Netflix), musiques (itune), livres...

Attention, la valeur "0" a du sens lorsqu'il s'agit d'un nombre d'achats alors qu'elle doit signifier une donnée manquante dans le cas d'une notation.

1.2 Factorisation

Définition

La décomposition en valeurs singulières (SVD) d'une matrice ainsi que la Non Negativ Matrix Factorization (NMF) sont utilisées dans ce contexte pour rechercher les facteurs (matrices $\mathbf{W}$ et $\mathbf{H}$) reconstruisant au mieux la matrice $\mathbf{X} \approx \mathbf{WH}$ avec une contrainte de *parcimonie* ou faible rang sur les matrices $\mathbf{W}$ et $\mathbf{H}$. Contrairement à la SVD où les facteurs sont recherchés orthogonaux 2 à 2, la NMF impose la contrainte de non négativité des matrices pour construire les facteurs de la décomposition. Ces facteurs ne permettent plus de représentation comme en ACP ou en MDS mais au moins une classification non supervisée tant des objets lignes que des objets colonnes de la matrice initiale. Ces classifications sont respectivement basées sur les matrices $\mathbf{W}$ et $\mathbf{H}$ des facteurs dits latents.

Schématiquement, w_{ij} dénote l'appétence du i-ème utilisateur pour le j-ème facteur latent, tandis que h_{jk} décrit quelle part du k-ième item intervient dans le j-ème facteur latent ; le modèle suppose que la note x_{ik} est la somme, sur tous les facteurs latents j, des produits $w_{ij} * h_{jk}$.

Implémentation, convergence

La SVD, est basée sur un critère de moindre carrés (norme trace des matrices). La librairie `NMF` (Gaujoux et Seoighe, 2010)[2] de R propose plusieurs algorithmes de factorisation non négative, principalement *Multiplicative update algorithms* et *Alternate least Square (ALS)*, adaptés à deux fonctions possibles de perte : divergence de Kullback-Leibler (KL) ou moindres carrés (norme trace).

Attention, les choix d'option : fonction objectif, algorithme, rang des matrices, influencent fortement les résultats obtenus et ce d'autant plus que les algorithmes (NMF) convergent (au mieux) vers des optimums locaux. La SVD bénéficie d'une convergence "globale" mais est moins adaptée au contexte car les solutions ne sont pas cohérentes avec l'objectif recherché : des notes ou comptages nécessairement positifs.

1.3 Complétion

Lorsque, les données sont des notes d'appréciation, la valeur "0" signifie en principe une valeur manquante. L'usage de la NMF ou de la SVD est alors abusif. Cette situation a été largement popularisée avec le concours Netflix à

1M$. Il a été abordé de façon théorique par Candes et Tao (2010)[1] comme un problème de *complétion de matrice* sous contrainte de parcimonie ; problème difficile, dont de très nombreuses approximations et implémentations ont depuis été proposées. Une simple utilisation est proposée ici dont l'algorithme (Mazumder et al. 2010)[3] conduit également à une factorisation.

2 Recommandation par NMF

2.1 Les données

Des données fictives triviales sont testées afin d'illustrer la démarche. Elles contiennent des nombres d'achats de certains produits ou des notes d'appréciation et peuvent être complétées à loisir au gré de votre imagination.

Charger le fichier `recom-jouet.dat`

```
jouet=read.table("recom-jouet.dat")
jouet
```

Le visualiser.

Résumé des données :

```
boxplot(jouet)
```

Les données sont bien creuses mais les variables s'expriment dans des unités et donc avec des variances très différentes. Une forme de normalisation peut s'avérer nécessaire. Elle concerne à la fois les produits (colonnes ou variables), car certains (chocolat) sont plébiscités plus que d'autres, ainsi que les clients qui peuvent avoir des échelles différentes de notation. C'est bien connu en *analyse sensorielle*.

2.2 Factorisation

Chargement de la librairie et identification des algorithmes disponibles. Plusieurs initialisation sont possibles ; seule celle aléatoire par défaut est utilisée.

```
library(NMF)
nmfAlgorithm()
nmfAlgorithm("brunet")
nmfAlgorithm("lee")
nmfAlgorithm("snmf/l")
nmfAlgorithm("snmf/r")
```

Identifier la fonction perte ; les deux derniers algorithmes sont issus de l'ALS.

Comparer les méthodes en exécutant pour chacune d'entre elles 10 factorisations de rang 5. Les exécutions sont répétées car la convergence locale dépend de l'initialisation.

```
res.multi.method=nmf(jouet, 5,nrun=10,
  list("brunet","lee","snmf/l","snmf/r"),
        seed = 111, .options ="t")
compare(res.multi.method)
consensusmap(res.multi.method,hclustfun="ward")
```

Plusieurs critères de comparaison sont proposés. Lequel choisir ? Pourquoi ?

Choix du rang des matrices de la décomposition.

```
estim.r=nmf(jouet,2:6,method="snmf/l",
   nrun=10,seed=111)
plot(estim.r)
consensusmap(estim.r)
```

Utiliser les résultats précédents pour déterminer un rang "optimal".

Une fois méthode et rang déterminés, itérer plusieurs fois l'exécution pour retenir la "meilleure".

```
nmf.jouet=nmf(jouet,4,method="snmf/l",
   nrun=30,seed=111)
```

Extraction des résultats numériques.

```
summary(nmf.jouet)
# les matrices de facteurs
w=basis(nmf.jouet)
h=coef(nmf.jouet)
```

2.3 Représentation des classifications

Production de classifications non-supervisées et graphiques associés aux matrices `w` et `h` de la factorisation. Ceci permet d'identifier des groupes de

clients au regard de leur consommation ou préférences comme de construire des classes de produits appréciés simultanément.

```
basismap(nmf.jouet,hclustfun="ward")
coefmap(nmf.jouet,hclustfun="ward")
```

Comme c'est logique, le dendrogramme produit dans les cartes précédentes est directement issu des classifications ascendantes hiérarchiques calculées à partir des distances euclidiennes entre les lignes de `w` d'une part et les colonnes de `h` d'autre part.

```
library(class)
distmod.h=dist(t(h), method="euclidean")
hclusmod.h=hclust(distmod.h,method="ward.D")
plot(hclusmod.h)
distmod.v=dist(w, method="euclidean")
hclusmod.v=hclust(distmod.v,method="ward.D")
plot(hclusmod.v)
```

La classification des objets est représentable dans les coordonnées d'un MDS ou dans les composantes d'une ACP des "facteurs" de la NMF ; c'est équivalent en considérant la distance euclidienne définie à partir de ces facteurs.

```
mdjouet= cmdscale(distmod.h, k=2)
dN.h=dimnames(h)[[2]]
plot(mdjouet, type="n", xlab="", ylab="",main="")
text(mdjouet,dN.h)
abline(v=0,h=0)

mdjouet= cmdscale(distmod.v, k=2)
dN.v=dimnames(w)[[1]]
plot(mdjouet, type="n", xlab="", ylab="",main="")
text(mdjouet,dN.v)
abline(v=0,h=0)
```

Les produits à plus forte occurrence ou note peuvent prendre trop d'importance, les facteurs sont réduits.

```
distmod.h=dist(scale(t(h)), method="eucl")
mdjouet= cmdscale(distmod.h, k=2)
hclusmod.h=hclust(distmod.h,method="ward.D")
plot(hclusmod.h)

hclasmod.h = cutree(hclusmod.h,k=4)
plot(mdjouet, type="n", xlab="", ylab="",main="")
text(mdjouet,dN.h,col=hclasmod.h)
abline(v=0,h=0)

distmod.v=dist(scale(w), method="eucl")
mdjouet= cmdscale(distmod.v, k=2)
hclusmod.v=hclust(dist.mod,method="ward.D")
plot(hclusmod.v)

hclasmod.v = cutree(hclusmod.v,k=2)
plot(mdjouet, type="n", xlab="", ylab="",main="")
text(mdjouet,dN.v,col=hclasmod.v)
abline(v=0,h=0)
```

Il n'est pas possible comme en ACP ou AFCM de mettre en relation les deux représentations des lignes et colonnes, individus et variables de la matrice factorisée. Cela peut être fait de façon détournée à l'aide d'une *heatmap* qui intègre les classifications obtenus en réordonnant les lignes et colonnes de $\mathbf{X}$.

```
# intégration des deux classifications
aheatmap(jouet,Rowv=hclusmod.v,
   Colv=hclusmod.h,annRow=as.factor(hclasmod.v),
   annCol=as.factor(hclasmod.h))
```

2.4 Recommandation

L'objectif de rechercher les produits les plus susceptibles d'intéresser les clients. Celui-ci est atteint en reconstruisant une approximation de la matrice `x` par produit des matrices des facteurs.

Les couples (client, produit) pour lesquels les valeurs reconstruites sont le plus élevées alors qu'il n'y a pas eu d'achat ou de notation, sont ceux qui sont ciblés afin de proposer le produit identifié au client de ce couple.

Attention, le choix du rang est déterminant. En utilisant le choix optimal précédent ($r = 4$) la reconstruction est finalement "trop" bonne et aucune recommandation n'émerge de la reconstruction de x. Le choix $r = 2$, sous-optimal, fait ressortir des couples candidats.

```
# Exécution avec r=2
nmf.jouet=nmf(jouet,2,method="snmf/l",
   nrun=30,seed=111)
# Matrice reconstruite
xchap=w%*%h
# Comparer avec les données initiales
# Identifier le plus fort score reconstruit
# par client
prod=apply(xchap-10*jouet,1,function(x)
   which.max(x))
# Identifier le produit correspondant
cbind(dN.v,dN.h[prod])
```

Remarques :

- La démarche s'applique également à de simples matrices de (0,1) de présence / absence d'achat.
- Les matrices peuvent être très grandes (données massives) sur des sites marchand, il est alors nécessaire d'utiliser des librairies avec représentation adaptée de matrices creuses pour réduire l'occupation mémoire. Seules les valeurs non nulles sont stockées.
- L'initialisation, ou *cold start*, de la matrice est un problème bien identifié. Cela concerne l'introduction d'un nouveau client ou d'un nouveau produit.
- D'autres approches concurrentes sont proposées ci-après.

3 Approches concurrentes

3.1 Par SVD

La décomposition en valeurs singulières propose également une factorisation de la matrice $\mathbf{X} = \mathbf{U}\mathbf{\Lambda}\mathbf{V}'$. Celle-ci a de bien meilleure propriétés numériques dont l'unicité de la solution optimale qui est atteinte pour un rang fixé. Des contraintes de parcimonie ou de régularité peuvent également être associées à la fonction perte quadratique (*sparse SVD*).

ACP

La démarche est équivalente et découle directement de la SVD donc de l'analyse en composantes principales de $\mathbf{X}$. Remarquer une forte similitude entre les représentions obtenus par MDS des facteurs de le NMF et celles de l'ACP :

```
library(FactoMineR)
PCA(jouet)
PCA(t(jouet))
```

Recommandation

```
# approximation de rang 2 par SVD
res=svd(jouet)
# Matrice reconstruite
xchap=res$u[,1:2]%*%diag(res$d[1:2])
   %*%t(res$v[,1:2])
# Comparer avec les données initiales
# Identifier le plus fort score reconstruit
# par client
prod=apply(xchap-10*jouet,1,function(x)
   which.max(x))
# Identifier le produit correspondant
cbind(dN.v,dN.h[prod])
```

Qui conduit donc, sur cet exemple trivial, à des résultats identiques.

3.2 Par complétion de matrice

Lorsque, les données sont des notes d'appréciation, la valeur "0" signifie en principe une valeur manquante et le problème est formellement celui d'une complétion de matrice. La librairie `softImpute` de R en propose une solution par un algorithme de SVD seuillée (Hastie et Mazumder et al. 2010)[3]. Il s'agit donc d'approcher une matrice très creuse avec beaucoup de valeurs manquantes par une matrice de faible rang. L'algorithme est analogue à un algorithme EM pour imputation de données manquantes. La fonction R accepte la classe de représentation des grandes matrices creuses.

Les valeurs nulles sont remplacées par des valeurs manquantes.

```
jouet.na=jouet
jouet.na[jouet==0]=NA
```

La recommandation se fait comme avec la SVD. Une étude plus approfondie de cet algorithme et de son usage s'avère nécessaire, notamment pour régler rang et paramètre de pénalisation.

```
library(softImpute)
res=softImpute(jouet.na,rank.max=2,type="svd",
   lambda=1)
# Matrice reconstruite
xchap=res$u%*%diag(res$d)%*%t(res$v)
# Comparer avec les données initiales
# Identifier le plus fort score reconstruit
# par client
prod=apply(xchap-10*jouet,1,function(x)
   which.max(x))
# Identifier le produit correspondant
cbind(dN.v,dN.h[prod])
```

Les résultats sont à nouveau identiques sur cet exemple trivial.

Cette étude n'est qu'une brève introduction au problème du filtrage collaboratif. De nombreuses questions n'ont pas été abordées dont le *cold start* et surtout celle très importante de l'*évaluation* de tels systèmes. Un procédé simple consiste à volontairement supprimer des valeurs afin de voir si le système les retrouve et en quelle proportion ; c'était le principe du concours Netflix.

Références

[1] E.J. Candes et T. Tao, *The Power of Convex Relaxation : Near-Optimal Matrix Completion*, Information Theory, IEEE Transactions on **56** (2010), n° 5, 2053–2080.

[2] Renaud Gaujoux et Cathal Seoighe, *A flexible R package for nonnegative matrix factorization*, BMC Bioinformatics **11** (2010), n° 1, 367, http://www.biomedcentral.com/1471-2105/11/367.

[3] R. Mazumder, T. Hastie et R. Tibshirani, *Spectral Regularization Algorithms for Learning Large Incomplete Matrices*, Journal of Machine Learning Research **11** (2010), 2287–2322, http://www.stanford.edu/~hastie/Papers/mazumder10a.pdf.