Von Mises-Fisher based (co-)clustering for high-dimensional sparse data : application to text and collaborative filtering data

Aghiles Salah

DOCTORAL SCHOOL OF COMPUTER SCIENCE, TELECOMMUNICATION AND ELECTRONICS (EDITE PARIS)

Von Mises-Fisher based (Co-)Clustering for High-dimensional Sparse data

Application to Text and Collaborative Filtering data

by

Aghiles Salah

THIS DISSERTATION IS SUBMITTED FOR THE DEGREE OF
DOCTOR OF COMPUTER SCIENCE
AT THE UNIVERSITY OF PARIS DESCARTES

Committee:

Pr. Mohamed Nadif	University of Paris Descartes, Supervisor
Pr. Christophe Ambroise	University of Évry Val d'Essonne
Pr. Fabrice Rossi	University of Panthéon-Sorbonne
Pr. Talel Abdessalem	Telecom ParisTech
Pr. Josiane Mothe	ESPE, University of Toulouse
Dr. Nicoleta Rogovschi	University of Paris Descartes, Co-supervisor

ÉCOLE DOCTORALE INFORMATIQUE, TÉLÉCOMMUNICATIONS ET ÉLECTRONIQUE
(EDITE DE PARIS)

Von Mises-Fisher based (Co-)Clustering for High-dimensional Sparse data

Application to Text and Collaborative Filtering data

Par

Aghiles Salah

THÈSE PRÉSENTÉE EN VUE DE L'OBTENTION DU TITRE DE
DOCTEUR EN INFORMATIQUE
À L' UNIVERSITÉ DE PARIS DESCARTES

Jury composé de :

Mohamed Nadif	PU, Université Paris Descartes	Directeur
Christophe Ambroise	PU, Université Évry Val d'Essonne	Rapporteur
Fabrice Rossi	PU, Université Panthéon-Sorbonne	Rapporteur
Talel Abdessalem	PU, Telecom ParisTech	Examinateur
Josiane Mothe	PU, ESPE, Université de Toulouse	Examinateur
Nicoleta Rogovschi	MCU, Université Paris Descartes	Co-encadrant

I would like to dedicate this thesis to my loving parents, my uncle Lakhdar and my grandmother Khadija …

Remerciements

Nombreuses sont les personnes qui ont contribué de près ou de loin à la réussite de cette thèse. Tout d'abord, un grand merci à mon directeur de thèse Pr. Mohamed Nadif pour son soutien, ses orientations et son implication active. J'ai beaucoup appris de lui et profité de sa grande compétence en modélisation et apprentissage statistique. Le Pr. Nadif a toujours montré une grande confiance en mes capacités et m'a permis de travailler de manière autonome, tout en fournissant des conseils précieux et en me poussant à explorer plus en profondeur certaines questions de recherche. Enfin, ses qualités humaines, sa rigueur scientifique, sa disponibilité et ses encouragements constants sont les facteurs clés qui m'ont aidé à mûrir en tant que chercheur et mener à terme ce projet de thèse.

Je remercie aussi le Dr. Nicoleta Rogovschi pour sa participation en tant que co-encadrant. Elle a montré une grande foie en moi et m'a encouragé tout au long de cette thèse. Ses conseils sur des questions de rédaction et de présentation m'ont poussé à apprendre d'autres aspects d'être un bon chercheur.

Je souhaite aussi exprimer mon respect et ma gratitude au Dr. François Role avec qui j'ai eu l'occasion de collaborer et d'avoir beaucoup d'échanges fructueux sur des aspects techniques, de fouille de données et bien d'autres. J'ai grandement apprécié ses qualités humaines et scientifiques.

J'ai la chance de bénéficier d'un jury de thèse de grande qualité, en particulier je remercie les rapporteurs Pr. Christophe Ambroise et Pr. Fabrice Rossi pour leurs lectures approfondies de mon manuscrit ainsi que leurs remarques et recommandations à la fois pertinentes et constructives. Je voudrais aussi remercier les autres membres de mon jury, Pr. Josiane Mothe et Pr. Talel Abdessalem, de m'avoir fait l'honneur d'accepter d'examiner mon travail.

Mon expérience en tant que doctorant ne serait sans doute pas aussi gratifiante sans mes amis et collègues de bureau, Melissa Ailem, Charlotte Laclau, Kais Allab et Adel Benaissa, que je tiens à remercier pour leurs sympathie, gentillesse et générosité.

D'un côté un peu plus personnel j'aimerais remercier un certain nombre de personnes qui m'ont toujours encouragé et contribué de manière indirecte au bon déroulement de cette thèse, à savoir mes cousins Lounes et Lyes, ma tante Djamila, mes deux frères Yacine et

Tarik, mes oncles Rachid et Madjid, mes amis Younès et Karim et mes cousins Faredj et Mourad.

Enfin, il n'y a pas de mots pour exprimer ma reconnaissance à ma mère Fatiha, mon père Hacène, mon oncle Lakhdar et ma grand-mère Khadija pour leurs soutiens et encouragements indéfectibles ainsi que tous les sacrifices qu'ils ont fait pour mon épanouissement sur tous les plans de la vie.

Abstract

Cluster analysis or *clustering*, which aims to group together similar objects, is undoubtedly a very powerful unsupervised learning technique. With the growing amount of available data, clustering is increasingly gaining in importance in various areas of data science for several reasons such as automatic summarization, dimensionality reduction, visualization, outlier detection, speed up research engines, organization of huge data sets, etc. Existing clustering approaches are, however, severely challenged by the high dimensionality and extreme sparsity of the data sets arising in some current areas of interest, such as Collaborative Filtering (CF) and text mining. Such data often consists of thousands of features and more than 95% of zero entries. In addition to being high dimensional and sparse, the data sets encountered in the aforementioned domains are also directional in nature. In fact, several previous studies have empirically demonstrated that directional measures—that measure the distance between objects relative to the angle between them—, such as the *cosine* similarity, are substantially superior to other measures such as *Euclidean* distortions, for clustering text documents or assessing the similarities between users/items in CF. This suggests that in such context only the direction of a data vector (e.g., text document) is relevant, not its magnitude. It is worth noting that the *cosine* similarity is exactly the scalar product between unit length data vectors, i.e., L_2 normalized vectors. Thus, from a probabilistic perspective using the *cosine* similarity is equivalent to assuming that the data are directional data distributed on the surface of a unit-hypersphere.

Despite the substantial empirical evidence that certain high dimensional sparse data sets, such as those encountered in the above domains, are better modeled as directional data, most existing models in text mining and CF are based on popular assumptions such as Gaussian, Multinomial or Bernoulli which are inadequate for L_2 normalized data. In this thesis, we focus on the two challenging tasks of text document clustering and item recommendation, which are still attracting a lot of attention in the domains of text mining and CF, respectively. In order to address the above limitations, we propose a suite of new models and algorithms which rely on the von Mises-Fisher (vMF) assumption that arises naturally for directional data lying on a unit-hypersphere. The object of this thesis is fourfold.

First, we propose a novel efficient incremental variant of the spherical k-means algorithm tailored for collaborative filtering, and which is able to handle effectively the frequent changes in the CF data: occurrence of new users, items, ratings and update of existing ratings. Two situations are considered: static situation where data are kept unchanged, and dynamic situation where new data are being added as they become available.

Second, with the advent of social networks, social CF approaches have proven to be very effective in alleviating the sparsity related issues and, thereby, improving the recommendations. Such approaches are motivated by the fact that, in real life people often turn to their friends to ask for a nice movie to watch, book to read, etc. A major shortcoming of existing models to social collaborative filtering is that they are inadequate for directional data. On one hand we have the benefits of modeling CF data as directional, and on the other hand we have the benefits of incorporating information from social network into CF. It is therefore reasonable to expect that we can alleviate the sparsity problem and improve recommendations even better if we leverage simultaneously the directional characteristics of CF data and the social interactions among users. This is precisely what we do in this thesis; we develop a new directional model-based CF approach that accounts for social interactions among users. Empirical results on various real-world data sets suggest that not only the directional nature of CF data, but also the social interactions among users should be taken into account to reach a high quality of recommendation.

Third, in the context of high dimensionality and sparsity, co-clustering, or simultaneous clustering of rows and columns of data matrices, turns out to be more beneficial than traditional one-sided clustering even if we are interested in clustering along one dimension, only. In this thesis, we develop a general co-clustering model that is based on the vMF assumption. Unlike existing co-clustering models, the proposed one successfully accounts for the intrinsic directional characteristics of some data sets such as text. We study the theoretical connections of the proposed model with existing ones, and we derive a suite of new vMF-based co-clustering algorithms that turn out to be very scalable and effective for analyzing high dimensional sparse document-term matrices, as illustrated in our experiments.

Fourth, due to high dimensionality and sparsity, co-clustering approaches, like one-sided clustering methods, tend to generate highly skewed solutions with very unbalanced or even empty clusters when the number of required clusters is large. In order to overcome this difficulty, we develop a novel directional co-clustering approach which successfully integrates a “conscience” mechanism that penalizes the clusters according to their sizes. The proposed algorithm is scalable and monotonically increases a spherical k-means like criterion by intertwining row and column clusterings at each step. Moreover, empirical results provide strong support for the effectiveness of the proposed approach.

Résumé

La classification automatique, qui consiste à regrouper des objets similaires au sein de groupes, également appelés classes ou *clusters*, est sans aucun doute l'une des méthodes d'apprentissage non-supervisé les plus utiles dans le contexte du *Big Data*. En effet, avec l'expansion des volumes de données disponibles, notamment sur le web, la classification ne cesse de gagner en importance dans le domaine de la science des données pour la réalisation de différentes tâches, telles que le résumé automatique, la réduction de dimension, la visualisation, la détection d'anomalies, l'accélération des moteurs de recherche, l'organisation d'énormes ensembles de données, etc. De nombreuses méthodes de classification ont été développées à ce jour, ces dernières sont cependant fortement mises en difficulté par les caractéristiques complexes des ensembles de données que l'on rencontre dans certains domaines d'actualité tel que le Filtrage Collaboratif (FC) et de la fouille de textes. Ces données, souvent représentées sous forme de matrices, sont de très grande dimension (des milliers de variables) et extrêmement creuses (ou *sparses*, avec plus de 95% de zéros).

En plus d'être de grande dimension et *sparse*, les données rencontrées dans les domaines mentionnés ci-dessus sont également de nature directionnelles. En effet, plusieurs études antérieures ont démontré empiriquement que les mesures directionnelles, telle que la similarité cosinus, sont supérieurs à d'autres mesures, telle que la distance Euclidiennes, pour la classification des documents textuels ou pour mesurer les similitudes entre les utilisateurs/items dans le FC. Cela suggère que, dans un tel contexte, c'est la direction d'un vecteur de données (e.g., représentant un document texte) qui est pertinente, et non pas sa longueur. Il est intéressant de noter que la similarité cosinus est exactement le produit scalaire entre des vecteurs unitaires (de norme 1). Ainsi, d'un point de vue probabiliste l'utilisation de la similarité cosinus revient à supposer que les données sont directionnelles et réparties sur la surface d'une hypersphère unité.

En dépit des nombreuses preuves empiriques suggérant que certains ensembles de données *sparses* et de grande dimension sont mieux modélisés sur une hypersphère unité, la plupart des modèles existants dans le contexte de la fouille de textes et du FC s'appuient sur des hypothèses populaires : distributions Gaussiennes ou Multinomiales, qui sont malheureusement inadéquates pour des données directionnelles.

Dans cette thèse, nous nous focalisons sur deux challenges d'actualité, à savoir la classification des documents textuels et la recommandation d'items, qui ne cesse d'attirer l'attention dans les domaines de la fouille de textes et celui du filtrage collaborative, respectivement. Afin de répondre aux limitations ci-dessus, nous proposons une série de nouveaux modèles et algorithmes qui s'appuient sur la distribution de von Mises-Fisher (vMF) qui est plus appropriée aux données directionnelles distribuées sur une hypersphère unité. Les contributions majeures de cette thèse s'articulent autour de quatre axes principaux.

Dans un premier temps nous proposons une nouvelle variante incrémentale de l'algorithme k-means sphérique, qui s'appuyant sur la similarité cosinus permet de gérer efficacement les changements fréquents dans les données du filtrage collaboratif, à savoir l'apparition de nouveaux utilisateurs, items, notes et mise à jour de notes existantes. Les résultats expérimentaux confirment l'efficacité et la rapidité de notre approche dans les deux situations suivantes: statique où les données restent inchangées, et dynamique où de nouvelles données sont intégrées au fur et à mesure de leurs disponibilités.

Dans un second temps nous nous intéressons à l'exploitation des réseaux sociaux dans les systèmes de filtrage collaboratif. Avec l'avènement des réseaux sociaux les approches de FC sociale se sont avérées très efficaces pour atténuer les problèmes liés à la *sparsité* des données, ce qui permet d'améliorer la qualité des recommandations de manière notable dans la plus part des cas. D'une part, nous avons les avantages de la modélisation des données sur une hypersphère unité, et d'autre part, nous avons les avantages de l'intégration des informations des réseaux sociaux dans le FC. L'exploitation simultanée des caractéristiques directionnelles des données du FC et les informations provenant des réseaux sociaux permettra d'améliorer les recommandations de façon significative. Pour ce faire, nous développons une nouvelle approche basée sur un mélange de distributions de vMF tout en tenant compte des interactions sociales entre les utilisateurs. Les résultats empiriques sur divers ensembles de données réels suggèrent que, pour formuler des recommandations de bonne qualité, il est non seulement nécessaire de prendre en compte la nature directionnelle des données du FC mais aussi les interactions sociales entre les utilisateurs.

Dans un troisièmes temps, nous nous intéressons à la classification croisée qui consiste, cette fois-ci, à partitionner simultanément les deux dimensions d'une matrice de données afin de faire ressortir une structure en blocs latents dans les données initiales. Cette dernière s'avère être plus bénéfique que la classification simple pour le traitement de matrices creuses à grande dimension, et cela même si l'utilisateur est intéressé uniquement par le partitionnement sur une seule dimension. Les approches de classification croisées existantes s'appuient, cependant, sur des hypothèses qui sont inappropriées pour des données projetées sur une hypersphère unité. Afin de surmonter cette limite, nous proposons un nouveau modèle

de mélange pour la classification croisée qui s'appuie sur un mélange de distributions vMF. Ce modèle qui est parcimonieux a l'avantage d'exploiter simultanément les bénéfices de la classification croisée et de la modélisation des données sur une hypersphère unité. Nous étudions les connexions théoriques du modèle proposé avec d'autres modèles, et nous dérivons une série de nouveaux algorithmes qui s'avèrent évolutifs et efficaces pour la classification croisée des matrices documents-termes qui ont la particularité d'être très creuses et de grande dimension.

La dernière contribution de cette thèse consiste à contrer les problèmes liés à l'obtention de classes vides ou très déséquilibrées. En effet, du fait de la grande dimensionnalité et de la *sparsité* caractérisant les données, les méthodes de classification croisée, comme les méthodes de classification simple, ont tendance à générer des classes vides ou très déséquilibrées, en particulier lorsque le nombre de classes souhaité est grand. Afin de surmonter cette difficulté, nous développons une nouvelle approche de classification croisée directionnelle qui intègre avec succès un mécanisme de *conscience*. Ce dernier permet de pénaliser les clusters en fonction de leurs tailles. L'algorithme proposé est évolutif et garantit de faire croître, à chaque itération, de façon monotone un critère de type k-means sphérique en alternant la classification des lignes et des colonnes. En outre, sur des données réelles, les résultats empiriques obtenus appuient fortement l'efficacité de l'approche proposée.

Table of contents

List of figures **xvii**

List of tables **xxi**

Introduction **1**

I.1. Motivation and Contribution . . . 2

I.1.1. Why von Mises-Fisher based models ? . . . 2

I.1.2. Contributions . . . 3

I.2. Overview . . . 5

I.3. Notations . . . 6

1 Clustering approaches **9**

1.1 Hierarchical clustering . . . 9

1.2 Density-based clustering . . . 11

1.3 Graph-based clustering . . . 11

1.4 Partitional clustering . . . 13

1.4.1 Centroid-based approaches . . . 13

1.4.2 Model-based approaches . . . 14

1.5 Conclusion . . . 16

2 Incremental directional clustering for a dynamic collaborative filtering system **17**

2.1 Motivation . . . 17

2.2 Introduction . . . 18

2.3 Related work and contribution . . . 19

2.4 Online spherical k-means . . . 20

2.5 Efficient Incremental Collaborative Filtering system . . . 22

2.5.1 Training step . . . 22

2.5.2 Prediction step . . . 24

2.5.3 Incremental training step . . . 25

2.6 Experimental results . . . 26
2.6.1 Datasets . . . 26
2.6.2 Methods and environment . . . 27
2.6.3 Evaluation metrics . . . 27
2.6.4 Performance comparison . . . 28
2.7 Conclusion and perspectives . . . 35

3 Directional Clustering with Social-Network information for Recommendation 37
3.1 Introduction . . . 37
3.2 Related work . . . 39
3.3 Preliminaries . . . 40
3.3.1 The Multivariate von Mises-Fisher (vMF) Distribution . . . 40
3.3.2 The mixture of von Mises-Fisher Distributions (*movMFs*) . . . 41
3.4 Social von Mises-Fisher Mixture Model . . . 43
3.4.1 Maximum Likelihood estimates . . . 44
3.5 Experimental study . . . 48
3.5.1 Datasets . . . 48
3.5.2 Evaluation Metrics . . . 49
3.5.3 Experimental Settings . . . 51
3.5.4 Empirical results and discussion . . . 52
3.5.5 movMFs vs Social-movMFs on cold start users . . . 55
3.5.6 Impact of the number of clusters and the regularization . . . 56
3.6 Conclusion and perspectives . . . 57

4 Co-clustering 59
4.1 Introduction . . . 59
4.2 Metric-based co-clustering . . . 61
4.3 Graph-based co-clustering . . . 62
4.4 Model-based co-clustering . . . 63
4.5 Non-Negative Matrix Factorization-based approaches . . . 65
4.6 Conclusion . . . 66

5 Von Mises-Fisher based Co-clustering 69
5.1 Introduction . . . 69
5.2 Related work . . . 71
5.3 A Mixture of von Mises-Fisher Distributions for Co-clustering . . . 73
5.3.1 Connection to existing models . . . 75

5.3.2 Maximum Likelihood estimates and the EM_b algorithm 76
5.3.3 Classification Maximum Likelihood estimates and the CEM_b algorithm 78
5.4 Diagonal Block Spherical Kmeans (dbSkmeans) 80
5.4.1 Definition 80
5.4.2 Connection to other algorithms 82
5.4.3 Analysis of column assignment 83
5.5 Stochastic Variants 84
5.5.1 Stochastic *dbmovMFs* and the $SEM_{\mathbf{b}}$ algorithm 84
5.5.2 Simulated annealing *dbmovMFs*-based algorithms 84
5.6 Computational Complexity in the Worst Case 87
5.7 Experimental results 88
5.7.1 Simulated data sets 89
5.7.2 Real-world datasets 90
5.7.3 Assessing the number of clusters 98
5.8 Conclusion and perspectives 102

6 Frequency Sensitive Co-clustering 105
6.1 Introduction 105
6.2 Frequency Sensitive Co-clustering 107
6.3 Experimental results 110
6.3.1 Datasets 111
6.3.2 Evaluation measures 112
6.3.3 Empirical results and discussion 112
6.4 Conclusion and perspectives 115

Conclusion and Perspectives 117

List of Publications 121

References 123

Appendix A Parameters Estimation 135

List of figures

1.1 An indexed hierarchy represented as a dendrogram 10
1.2 Illustration of graph clustering in social networks. The graph corresponds to the well known Zachary's karate club network; friendships between 34 members of a karate club at a US university in the 1970s (Zachary, 1977). . 12
1.3 Gaussian Mixture in $\mathbb{R}^2$ 15

2.1 Assessing the number of clusters for `EICF` on different datasets, using F1 metric. 29
2.2 Evaluation of several recommender systems on ML-100k dataset (static situation). 30
2.3 Evaluation of several recommender systems on the ML-1M dataset (static situation). 30
2.4 Evaluation of several recommender systems on Epinions dataset (static situation). 31
2.5 Comparison of training + prediction time on ML-1M subsets. 31
2.6 Evaluation of several RS on ML-100k dataset (Dynamic situation). 33
2.7 Evaluation of several recommender systems on ML-1M dataset (Dynamic situation). 33
2.8 Evaluation of several CF systems on the Epinions dataset (Dynamic situation). 33

3.1 Three sample of 100 points according to three 2-vMF distributions, with different concentration parameters, and the same mean direction $\mu = (1/\sqrt{2}, 1/\sqrt{2})$. The segment starting from the circle centre denotes the mean direction of each vMF distribution. 41
3.2 *movMFs* as a graphical model. 42
3.3 Comparison of average nDCG over different datasets. 53
3.4 Comparison of average MRR over different datasets. 53
3.5 Comparison of average Precision@10 over different datasets. 53

3.6 Comparison of average Recall@10 over different datasets. 53
3.7 % of cold start users in the user-item matrix (Preference) and social-network (Social). 54
3.8 Cold start users : distribution of out degree, i.e., the number of social interactions per cold start user, over the different datasets 56
3.9 Impact of the number of clusters g. 56
3.10 Impact of the regularization parameter λ. 57

4.1 Binary table reorganized according to the row and column partitions 60
4.2 Binary data and its associated bipartite graph 63
4.3 LBM's graphical model 64

5.1 A co-clustering obtained by using $\text{EM}_{\mathbf{b}}$ (soft-dbmovMF): (left) original data, (middle) data reorganized according to $\mathbf{z}$, (right) data reorganized data according $(\mathbf{z}, \mathbf{w})$ 72
5.2 Graphical models: left *movMFs* (Banerjee et al., 2005b), right *dbmovMFs* (proposed model). 74
5.3 Von Mises-Fisher mixture models-based clustering (left) and co-clustering (right) algorithms. 87
5.4 Simulated datasets reorganized according to row and column partitions: (a) sdata1, (b) sdata2, (c) sdata3, (d) sdata4 (e) sdata5. 89
5.5 Behaviour of log-likelihood over iterations, on the different datasets. 94
5.6 Best NMI and ARI over all datasets for each method. Our algorithms provide good performances in terms both NMI and ARI while *movMFs*-based approach sometimes provide good NMI but low ARI. For instance, on SPORTS $\text{EM}_{\mathbf{b}}$ provides NMI = 0.62 and ARI = 0.61 while SKmeans provides NMI = 0.63 and ARI = 0.44. (see confusion matrices of Table 5.7) 95
5.7 Visualization of SPORTS dataset: (a) original, (b) reorganized according to Skmeans's row partition, (c) reorganized according to $\text{EM}_{\mathbf{b}}$'s row and column partitions. 95
5.8 Distribution of concentration parameters 96
5.9 Behaviour of the log-likelihood, AIC, AIC3 and BIC when varying the number of clusters on the CSTR dataset (True number of classes: 4). 100
5.10 Behaviour of the log-likelihood, AIC, AIC3 and BIC when varying the number of clusters on the CLASSIC4 dataset (True number of classes: 4). . 100
5.11 Behaviour of the log-likelihood, AIC, AIC3 and BIC when varying the number of clusters on the WEBACE dataset (True number of classes: 20). . 101

5.12 Behaviour of the log-likelihood, AIC, AIC3 and BIC when varying the number of clusters on the NG20 dataset (True number of classes: 20). . . . 101
5.13 Behaviour of the log-likelihood, AIC, AIC3 and BIC when varying the number of clusters on the SPORTS dataset (True number of classes: 7). . . 101
5.14 Behaviour of the log-likelihood, AIC, AIC3 and BIC when varying the number of clusters on the TDT2 dataset (True number of classes: 30). . . . 101

6.1 NG2 dataset (Balance = 1). Left: original, middle: reorganized according to `Skmeans` result(NMI = 0.57, ARI = 0.67, Balance = 0.73), right: Reorganized according to `FSdbSkm` results(NMI = 0.74, ARI = 0.83, Balance = 0.98). . . 113
6.2 Left Balance vs NMI, Righ Balance vs ARI. The circles denote the scores recorded by `FSdbSkm`. 114

List of tables

1 Data matrix $\mathbf{X}$ and associated row and column partitions indicated respectively by the representations $\mathbf{z}$ and $\mathbf{Z}$, $\mathbf{w}$ and $\mathbf{W}$, with $g = 3$. 7

2.1 Comparison of MAE and RMSE achieved by three recommender systems, on a small example (numbers in bold correspond to best performances). . . 28

2.2 Comparison of computational times (in the worst case) in various situations. W^* denotes the number of observed ratings in $\mathbf{U}$. g and m are the number of row and column clusters, k denotes the number of neighbours for memory CF (`IUCF`, `IICF`). d^* is the number of observed ratings for a new user, n^* denotes the number of available ratings for a new item. Finally, B denotes the number of iterations 32

2.3 Computation time comparison in the dynamic scenario (in sec). 34

3.1 Description of Datasets 49

3.2 Comparison of Average recommendation accuracy over different datasets. "Improve" indicates the improvement reached by the proposed social model `Soc-movMF` relative to performance of the traditional `movMF` model. 52

3.3 Comparison of Average recommendation accuracy on cold start users—with 5 or fewer ratings—over different datasets. "Improve" indicates the improvement reached by the proposed social model `Soc-movMF` relative to the performance of `movMF`. 55

5.1 Computational complexity of different algorithms (in the worst case). nz denotes the number of non-zero entries in the matrix $\mathbf{X}$, g is the number of clusters (co-cluster) and *it* denotes the number of iterations. 88

5.2 Comparison of true and estimated parameters by our algorithms on different simulated dataset, (α,κ,μ) denote the true parameters while $(\hat{\alpha},\hat{\kappa},\hat{\mu})$ denote the estimated parameters. 90

5.3 Description of Datasets 91

5.4 Comparison of average NMI and ARI, on CSTR, CLASSIC4 and WEBACE datatsets. 93
5.5 Comparison of average NMI and ARI, on NG20, SPORTS and TDT2 datasets. 93
5.6 Comparison of classification log-likelihood. 93
5.7 SPORTS dataset: confusion matrices crossing the row clusters obtained by both algorithms (rows) and the true row clusters (columns). The column $z_{.h}$ indicates the sizes of clusters. 96
5.8 Word clusters resulting from $\text{SAEM}_{\mathbf{b}}$ on the CLASSIC4 dataset. Each cluster is represented by its top 15 terms, sorted according to their popularity. Clusters C1, C2, C3 and C4 correspond respectively to CACM, CISI, CRANFIELD and MEDLINE. 97
5.9 Clustering of CLASSIC4 into 7 co-clusters by $\text{SAEM}_{\mathbf{b}}$: the obtained word clusters represented by their top 15 terms, sorted according to their popularity. 98
5.10 The top 6 word clusters resulting from $\text{SAEM}_{\mathbf{b}}$ on dataset TDT2. Each cluster is represented by its top 15 terms, sorted according to their popularity. 98

6.1 Description of Datasets. 111
6.2 Comparison of Average NMI, ARI on the NG20, SPORTS, LA12 datasets. 113
6.3 Comparison of Average NMI, ARI on the NG17-19, CLASSIC4 and NG2 datasets. 113
6.4 Comparison BALANCE, RME and SDCS on the NG20, SPORTS, LA12 datasets. 114
6.5 Comparison BALANCE, RME and SDCS on NG17-19, CLASSIC4, NG2. 114
6.6 CLASSIC4 dataset: the term clusters discovered by `FSdbSkm` represented by their top 15 terms sorted according to *cosine* similarity. The clusters C1, C2, C3 and C4 corresponds respectively to CACM, CRANFIELD, CISI and MEDLINE. 115

Introduction

Clustering aims to organize a set of objects into homogeneous groups, such that objects in the same cluster (or group) are more "similar" to each other than to objects in different clusters (Xu et al., 2005; Everitt et al., 2011; Aggarwal and Reddy, 2013). In machine learning, where data is often represented by a matrix where the rows denote objects (instances, individuals), and the columns denote features (attributes, variables), clustering is a powerful unsupervised learning technique that aims to "summarize" data—seeks to find a reduced representation of data by grouping together either similar objects or variables—to ease their exploration and interpretation. More precisely, clustering seeks to discover a hidden structure of the data that can be exploited to represent the data in a compressed fashion, that reflects faithfully the original data. Due to its practical importance, clustering has received a lot of attention in various communities such as machine learning, data mining and information retrieval. This led to the development of a wide variety of clustering methods, regarding the context and types of data, i.e., binary, categorical, continuous or count data (Aggarwal and Reddy, 2013). Despite the extensive efforts of many research communities, clustering approaches are still seriously challenged by the characteristics—high dimensionality and sparsity—of certain data sets arising in some current areas of interest such as text mining and collaborative filtering. In fact, when dealing with such data, most of traditional clustering algorithms suffer from poor performances in terms of both scalability and quality of clustering. In this thesis, we propose a suite of novel algorithms that are based on rigorous probabilistic assumptions, and which are able to handle effectively and efficiently high dimensional sparse data sets. In particular, we focus on the two areas of text mining and collaborative filtering, with the aim to solve the challenging tasks of document clustering and item recommendation, using high dimensional sparse document-term and user-item matrices, respectively.

I.1. Motivation and Contribution

I.1.1. Why von Mises-Fisher based models?

Several data sets arising in some current areas of interest, such as text mining, collaborative filtering and bioinformatics, are high dimensional and extremely sparse; usually theses data sets consist of more than 1000 features and 95% of zero entries. Analysing such data is a major challenge. Indeed, most of existing clustering methods suffer from poor performances in this context. Apart from being high dimensional and sparse, the data sets arising in the above domains are also directional in nature (Mardia and Jupp, 2000; Banerjee et al., 2005b).

In text mining, where data arises in the form of high dimensional sparse document-term matrices, clustering text documents is of great interest for several practical reasons: automatic summarization and organization of documents, efficient browsing and navigation of huge text corpora, dimensionality reduction, visualization, speed up search engines, etc. In this domain, it is well known that normalizing the data vectors so that they lie on a unit-hypersphere—L_2 normalization—allows us to remove the biases induced by the lengths of documents and reach better clustering performances. Furthermore, it has been found that von Mises-Fisher (vMF) based clustering algorithms, including the spherical k-means algorithm (Dhillon and Modha, 2001), are superior to several other clustering methods which are based on popular assumptions such as Gaussian, Multinomial or Bernoulli, see for instance (Zhong and Ghosh, 2005; Gopal and Yang, 2014). Recall that the vMF distribution is a continuous probability distribution, from directional statistics (Mardia and Jupp, 2000), that arises naturally for directional data lying on the surface of a unit-hypersphere. In particular, the vMF distribution focuses on the directions of objects—data vectors—and measures the distance between them using the *cosine* similarity, which turns out to be a more effective measure for clustering text documents (Strehl et al., 2000). From a probabilistic perspective, the success of the L_2 normalization and vMF-based models, for clustering text documents, constitutes a strong empirical evidence that text data has intrinsic directional characteristics that match well with the modeling assumption of the vMF distribution.

Collaborative filtering (CF) is a widely used recommendation approach, it consists in making recommendations for users according to the preferences of other similar users. Over the last decade, CF systems have become central in modern web applications such as Amazon, Netflix, Youtube, Flixster, etc. In fact, CF makes it possible to filter the huge amount of information generated in the above applications so as to guide the users towards items that might interest them. Collaborative Filtering is another domain where it is better to model data as directional. In fact, let us consider the *cosine* similarity and *Pearson Correlation Coefficient (PCC)*, two widely used measures in this domain. The above two, or their

variants, have been found to be more effective and efficient than Euclidean distortions to assess similarities between users or items (Sarwar et al., 2001; Liu et al., 2014). Observe that both measures focus on the directions of data vectors, i.e., the similarity between two objects—users or items—is measured relative to the angle between them. The success of such measures in CF suggests that the direction of a user (resp., item) preference-vector is relevant, not its magnitude.

It is worth noting that both the above measures are exactly the scalar product between objects lying on the surface of a unit-hypersphere, i.e., objects of unit length (L_2 norm). This is straightforward for the *cosine* similarity, now let us look at the *PCC* between two users, represented by vectors $\mathbf{x}$ and $\mathbf{y}$ in $\mathbb{R}^d$, given by:

$$PCC(\mathbf{x},\mathbf{y}) = \frac{(\mathbf{x}-\bar{\mathbf{x}})^\top(\mathbf{y}-\bar{\mathbf{y}})}{\|\mathbf{x}-\bar{\mathbf{x}}\|\|\mathbf{y}-\bar{\mathbf{y}}\|},$$

where $\bar{\mathbf{x}}$ and $\bar{\mathbf{y}}$ are two vectors of the appropriate dimensions, such that $\bar{x}_1 = \cdots = \bar{x}_d = \sum_j x_j/d$ and $\bar{y}_1 = \cdots = \bar{y}_d = \sum_j y_j/d$. Let $\mathbf{x}' = \frac{\mathbf{x}-\bar{\mathbf{x}}}{\|\mathbf{x}-\bar{\mathbf{x}}\|}$ and $\mathbf{y}' = \frac{\mathbf{y}-\bar{\mathbf{y}}}{\|\mathbf{y}-\bar{\mathbf{y}}\|}$, then the PCC between $\mathbf{x}$ and $\mathbf{y}$ is exactly the scalar product (*cosine* similarity) between the two unit-length vectors $\mathbf{x}'$ and $\mathbf{y}'$. Hence, using the *cosine* and *PCC* measures is equivalent to assume that CF data is directional and distributed on the surface of a unit-hypersphere. The success of such measures constitutes empirical evidence that CF data sets possess intrinsic directional properties (Mardia and Jupp, 2000) that should be taken into account when modeling such data.

Despite the substantial empirical evidence that some high dimensional sparse data sets are better modeled as directional[1] data, most existing models in the context of text document clustering and CF are based on popular assumptions such as Gaussian or Multinomial, which are inadequate for directional data destributed on the surface of a unit-hypersphere.

I.1.2. Contributions

This thesis deals with a suite of new models and algorithms tailored for directional data distributed on the surface of a unit-hypersphere. The proposed models are scalable and able to handle effectively certain high dimensional sparse data sets such as text and CF data. The major contributions of this thesis are structured around four main topics:

A directional clustering-based incremental collaborative filtering

Collaborative filtering systems seek to filter huge amount of information so as to provide users with useful items. While some CF approaches, such as memory and Matrix Factorization,

[1] In the rest of this thesis we treat "direction data" and "L_2 normalized data" as synonyms

may offer a good recommendation accuracy, they are computationally prohibitive. Scalable approaches are, therefore, needed if CF techniques are to be usable in realistic scenarios where data evolves rapidly. This thesis presents a scalable incremental variant of the spherical k-means algorithm, that is able to handle effectively the frequent changes in CF data such as submission of new ratings, update of existing ratings, appearance of new users and items.

Directional Clustering with Social-Network information

Recently, several works have demonstrated that incorporating information from social networks, such as friendships and trust relationships, into tradition traditional CF enables to alleviate significantly the sparsity related issues such as the problem of cold start users, i.e., who expressed very few ratings. Social-based CF builds on the assumption that, for making good recommendations, not only the user's expressed preferences are important but also the user's social interactions. This is natural as people often turn to their friends for advice before choosing a movie, a book, a restaurant, etc. By capturing this real-life behaviour, social CF approaches make more realistic recommendations and, thereby, improve the performances of traditional methods noticeably in most cases. This thesis propose a novel social CF model in order to address the sparsity issue. By contrast to existing social CF models, which are based on popular assumptions such as Gaussian, the proposed one builds on the von Mises-Fisher assumption that turns out to be more adequate to model CF data. Extensive experiments, on several real-wold data sets, illustrate the advantages of our modeling assumption and suggest that not only the users' social interactions, but also the intrinsic directional characteristics of CF data should be taken into account so as to improve recommendations.

Co-clustering via a mixture of von Mises-Fisher distributions

In the context of high dimensionality and sparsity, co-clustering, or simultaneous clustering of rows and columns of data matrices, turns out to be more beneficial than traditional one-sided clustering even if we are interested in clustering along one dimension, only. Popular co-clustering assumptions such as Gaussian, Multinomial or Bernoulli are, however, inadequate for L_2 normalized data. In this thesis, we propose novel rigorous probabilistic model for co-clustering, which is suitable for directional data lying on the surface of a unit hypersphere. The proposed model successfully integrates a directional measure, namely the *cosine* similarity, into a co-clustering framework. Based on the proposed model we derived several new co-clustering algorithms that are very scalable and effective. Empirical results, on numerous simulated and real-world data sets, demonstrate the effectiveness of our model and algorithms for handling high dimensional sparse document-term matrices.

Frequency sensitive co-clustering

It is well known that, because of high dimensionality and sparsity, co-clustering approaches, like one-sided clustering methods, tend to generate highly skewed solutions with very unbalanced or empty clusters (Dhillon et al., 2002; Cho et al., 2004; Banerjee et al., 2004; Zhang et al., 1999), especially when the number of required clusters is large. In some situations, however, it may be more beneficial to obtain balanced clusters or at least avoid bad solutions with very unbalanced or empty clusters. In order to tackle the aforementioned issue and avoid extremely skewed solutions, we develop a novel directional co-clustering method which integrates a "conscience" mechanism—inspired from Frequency Sensitive Competitive Learning—that prevents from such bad local solutions.

I.2. Overview

The rest of this thesis is organized as follows:

- **Chapter 1** reviews the major existing approaches to clustering and discusses their main strengthens and weakness.

- **Chapter 2** is devoted to clustering on a unit-hypersphere for item recommendation, it first reviews the spherical k-means algorithm, using the *cosine* similarity instead of *Euclidean* distortions, then it presents a novel efficient incremental variant of the spherical k-means algorithm that is designed for collaborative filtering, and which takes into account the frequent changes in the CF data, namely occurrence of new users, items, ratings and update of existing ratings.

- **Chapter 3** is about social collaborative filtering which consists in leveraging information from social networks so as to alleviate the sparsity problem and enhance recommendations. In this chapter, a new social CF model is described, the latter relies on the vMF mixture model, and it aims to partition the set of users into homogeneous groups as well as bring the distributions over clusters of socially connected users closer to each other. In doing so, the proposed model is able to capture social interactions among users and, thereby, formulate more realistic recommendations.

- **Chapter 4** reviews the main existing approaches to co-clustering or simultaneous clustering of rows and columns of data matrices.

- **Chapters 5** and **6** are devoted to new co-clustering approaches tailored for directional data distributed on the surface of a unit-hypersphere. More precisely, Chapter 5 presents a novel co-clustering model that builds on the von Mises-Fisher distribution and a suite of algorithms arising from this model, including soft, hard, stochastic and simulated annealing variants. While Chapter 6 presents a novel vMF-based co-clustering approach with some balancing constraints on the row and column cluster sizes. A property that could be of great interest in some situations as illustrated in Chapter 6.

I.3. Notations

Through this thesis, we will use the following notations (unless stated otherwise):

- Matrices are denoted with boldface uppercase letters, vectors with boldface lowercase letters and sets by script style uppercase letters. The L_2 norm is denoted by $\|.\|$. The $(d-1)$ dimensional unit sphere embedded in $\mathbb{R}^d$ is denoted by $\mathbb{S}^{d-1}$.

- Data is represented by a matrix $\mathbf{X} = (x_{ij})$ of size $n \times d$, $x_{ij} \in \mathbb{R}$, the i^{th} row of this matrix is represented by a vector $\mathbf{x}_i = (x_{i1}, \ldots, x_{id})^\top$, where $\top$ denotes the transpose.

- The partition of the set of rows $\mathcal{I}$ into g clusters can be represented by a classification matrix $\mathbf{Z}$ of elements z_{ih} in $\{0,1\}$ satisfying $\sum_{h=1}^{g} z_{ih} = 1$. The notation $\mathbf{z} = (z_1, \ldots, z_n)^\top$, where $z_i \in \{1, \ldots, g\}$ represents the cluster label of i, will be also used. These notations are illustrated in Table 1.

- Similarly the notations $\mathbf{W} = (w_{jh})$, $w_{jh} \in \{0,1\}$ satisfying $\sum_{h=1}^{g} w_{jh} = 1$, and $\mathbf{w} = (w_1, \ldots, w_d)^\top$, where $w_j \in \{1, \ldots, g\}$ represents the cluster label of j, will be used to represent the partition of the set of columns $\mathcal{J}$, see Table 1.

- In the same way, the fuzzy classification matrix of $\mathcal{I}$ will be denoted by $\tilde{\mathbf{Z}} = (\tilde{z}_{ih})$ where $\tilde{z}_{ih} \in [0,1]$, satisfying $\sum_{h=1}^{g} \tilde{z}_{ih} = 1$, for all i in $\mathcal{I}$.

- The sums and the products relating to rows, columns, row clusters and column clusters will be subscripted respectively by the letters i, j and h, without indicating the limits of variation which will be implicit. So, the sums $\sum_i$, $\sum_j$ and $\sum_h$ stands respectively for $\sum_{i=1}^{n}$, $\sum_{j=1}^{d}$ and $\sum_{h=1}^{g}$.

Table 1 Data matrix $\mathbf{X}$ and associated row and column partitions indicated respectively by the representations $\mathbf{z}$ and $\mathbf{Z}$, $\mathbf{w}$ and $\mathbf{W}$, with $g = 3$.

		columns ($\mathcal{J}$)	$\mathbf{z}_{(n\times 1)}$	$\mathbf{Z} = (z_{ih})_{(n\times g)}$
		$1 \cdots j \cdots d$		
rows ($\mathcal{I}$)	$\mathbf{x}_1$	$x_{11} \cdots x_{1j} \cdots x_{1d}$	1	1 0 0
	$\vdots$	$\vdots$	$\vdots$	$\vdots$
	$\mathbf{x}_i$	$x_{i1} \cdots x_{ij} \cdots x_{id}$	3	0 0 1
	$\vdots$	$\vdots$	$\vdots$	$\vdots$
	$\mathbf{x}_n$	$x_{n1} \cdots x_{nj} \cdots x_{nd}$	2	0 1 0

$\mathbf{w}_{(1\times d)}$	$3 \cdots 1 \cdots 2$

$\mathbf{W}^T = (w_{hj})$ $(g\times d)$	$0 \cdots 1 \cdots 0$ $0 \cdots 0 \cdots 1$ $1 \cdots 0 \cdots 0$

Chapter 1

Clustering approaches

This chapter provides a brief survey of existing approaches to clustering. The objective here is not to provide a detailed description of all existing clustering methods, but rather to give an outline of the major approaches to clustering.

1.1 Hierarchical clustering

Given a set of objects to be clustered, hierarchical methods consist in constructing a sequence of partitions of the set varying from singletons to the whole set. A hierarchical clustering algorithm aims to transform a measure of dissimilarity between objects into a sequence of nested partitions, or an indexed hierarchy, such as the closest objects are grouped in the clusters with the smallest indexes. The obtained clustering hierarchy can be represented by a dendrogram as illustrated in Figure 1.1. The way in which the clustering hierarchy is built gives rise to two major hierarchical clustering categories.

Agglomerative hierarchical clustering. These approaches , also known as bottom up, first put each object in its own cluster, then start merging closest clusters until there is only a single cluster left. These methods require to choose a metric (e.g. euclidean distance), to compute dissimilarities between objects, and a linkage function (or criteria) to compute the dissimilarity between clusters based on the pairwise distances between objects. Several linkage criteria exist; the most popular are: single-linkage (Sibson, 1973), complete-linkage (Sorensen, 1948), average-linkage (Sokal, 1958) and Ward's criterion (Ward Jr, 1963). For instance, the single-linkage defines the distance between two clusters as the pairwise distance between the two closest objects—one in each cluster. In addition to the above traditional methods several agglomerative clustering approaches have been proposed, they differ in their dissimilarity metric and linkage strategy, for more details the reader can refer to survey papers (Xu et al., 2005; Murtagh and Contreras, 2012) or the book of Everitt et al. (2011).

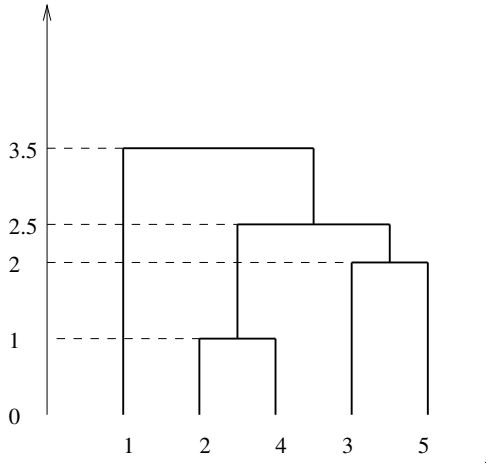


Fig. 1.1 An indexed hierarchy represented as a dendrogram

Divisive hierarchical clustering. As opposed to agglomerative methods, divisive, also denoted as top down, approaches assume that all objects belong to a big single cluster at the beginning, then split it repeatedly into smaller clusters, according to some chosen criterion, until each object form its own cluster or some stopping condition is reached. Example of divisive clustering approaches are MONA and DIANA described in (Kaufman and Rousseeuw, 2009), some other methods can be found in (Everitt et al., 2011). Divisive approaches are less common in practice, as opposed to agglomerative ones, due to their high computational cost.

The main advantages of hierarchical-based clustering approaches is that they do not require the number of clusters as an input, some methods, such as single-linkage, are able to discover clusters with non-convex shapes, and they can offer different clustering levels according to the position where the dendrogram is cut. These approaches exhibit, however, two major weaknesses: the first one is that they do not scale well to large datasets, the computational time complexity of most existing hierarchical methods is usually given respectively in $O(n^3)$ and $O(2^n)$ for agglomerative and divisive approaches, where n is the number of objects. The complexity of some efficient agglomerative approaches is $O(n^2)$. Hierarchical methods are therefore not suitable for large high dimensional datasets, such those encountered in text mining and collaborative filtering domains. The second, is that most existing methods perform the splitting or merging operations irreversibly and never undo what was done.

1.2 Density-based clustering

Density-based clustering methods assume that clusters are dense regions—of objects—in the feature space, and that clusters are separated by low dense regions. Objects in the latter regions are considered as noise. One of the most known density-based clustering approach is DBSCAN proposed in (Ester et al., 1996). In this approach, the density around a given object o_i is defined as the number of objects within a distance ε around object o_i. If the number of such objects if greater than a certain threshold $minPts$, called the density threshold, then o_i is considered a core point, and o_i is considered a border point otherwise. Based on the above notions, DBSCAN starts by randomly selecting an object o_i, if it is a core point then a cluster is formed by connecting all *density-reachable* objects from o_i, otherwise—o_i is a border point—another object is selected. This process is repeated until all objects are processed, for more details the reader can refer to (Ester et al., 1996). The DBSCAN method has the advantage of being able to discover clusters of arbitrary shapes. It is, however, highly sensitive to the settings of the density threshold $minPts$ and radius ε parameters. In order to overcome the latter drawback, the same authors proposed OPTICS (Ankerst et al., 1999) a generalization of DBSCAN that do not require the parameters $minPts$ and ε. In general density-based clustering methods are not very efficient for high dimensional data. Some approaches, such as grid-based methods (Wang et al., 1997; Hinneburg and Keim, 1998), that can handle more efficiently high dimensional data have been proposed. Nevertheless, the latter may be less effective in some situations. For further details and more recent developments in density based-clustering, please refer to (Kriegel et al., 2011).

1.3 Graph-based clustering

This category of clustering methods works on data represented as a graph—from a graph theory point of view—that consists of a set of vertices (or nodes) and a set of edges between them—usually undirected; each edge connects a pair of nodes. Given such a graph structure, the aim of graph-based clustering is to find clusters of nodes, in such way that nodes within a cluster are more connected to each other than to those in other clusters, an illustration is given in figure 1.2. In other words, graph clustering seeks a partition of a graph into k clusters in such way that there will be many edges within each cluster and only few edges between clusters. A large class of graph clustering methods consists in solving graph cut problems; may be the simplest and one of the most popular is the *minimum cut* problem. It seeks to find a partition of the nodes in a graph by minimizing the sum of weights of the edges between clusters. Several variants of *minimum cut* have been proposed, among which

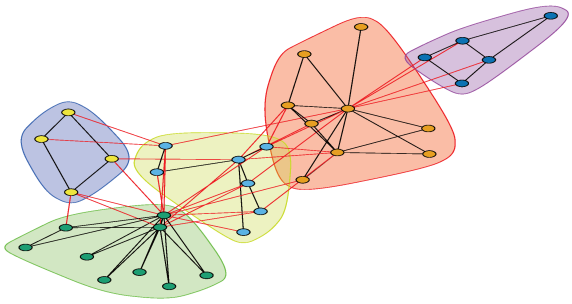


Fig. 1.2 Illustration of graph clustering in social networks. The graph corresponds to the well known Zachary's karate club network; friendships between 34 members of a karate club at a US university in the 1970s (Zachary, 1977).

we can cite the *minimum ratio cut* problem (Leighton and Rao, 1988; Chan et al., 1994) that introduces a division by the size of each clusters, into the *min cut* problem, in order to avoid skewed solutions with isolated nodes in clusters. In the same vein the *minimum normalized cut* problem (Shi and Malik, 1997, 2000) introduces a division by the sum of node degrees within each cluster. The *min-max cut* problem (Ding et al., 2001) aims to maximize the similarities within each cluster and to minimize similarities between clusters, simultaneously. The aforementioned problems are usually solved based on the spectral graph theory, by using proper relaxations. For details on spectral-based approaches for solving graph cut problems, the reader can refer to (Von Luxburg, 2007) or (Nascimento and De Carvalho, 2011). Several other graph-clustering methods exist, examples of such approaches can be found in the survey papers (Xu et al., 2005) and (Schaeffer, 2007).

The main advantage of graph-based clustering, is that many practical problems can be modeled by graphs, it is sufficient to treat objects as nodes and to define relations (e.g. similarities) between them. Thereby, such approaches have a broad scope of application, we can rely on graph clustering as long as we can model our data by a graph. Another notable advantage is the ability of graph-based clustering to discover clusters of arbitrary shapes. Nevertheless, most of graph-based methods are computationally demanding. Furthermore, spectral-based approaches are seriously challenged by high dimensional and sparsity and my suffer from poor performances in such situation.

1.4 Partitional clustering

The aim of partitional clustering is to discover a hidden partition of a set of objects into g disjoint clusters, i.e., with non-hierarchical structure, in such a way that objects within a cluster are more "similar" to each other than to those in other clusters. Partitional approaches are widely used in practice due to their simplicity and scalability.

1.4.1 Centroid-based approaches

Centroid-based clustering represents a large class of partitional clustering methods. In this approach each cluster is represented by a centroid (or prototype) vector—which is not necessarily in the set of objects—and the aim is to find a partition of the objects, into g clusters, that optimizes some criterion, usually the deviation of each object from its cluster centroid. A trivial way to find such an optimal partition is to enumerate all possible partitions. As the latter is infeasible in practice, centroid-based approaches rely on optimization procedures which guarantee only a locally optimal solution. The choice of the cluster representatives—centroids—and the criterion to optimize gives rise to a large number of centroid-based clustering methods. Among which we can mention the well known k-means algorithm.

The k-means algorithm (MacQueen et al., 1967; Bock, 2007) is the most popular partitional clustering method, it seeks to find a partition of the objects into g clusters in such a way that the sum of squared euclidean distances, between each object and its cluster centroid, is minimized. Formally, k-means aims to minimize the following objective function

$$C(\mu, \mathbf{z}) = \sum_{i=1}^{n} \sum_{h=1}^{g} z_{ih} \|\mathbf{x}_i - \mu_h\|^2. \tag{1.1}$$

where $\mathbf{x}_i \in \mathbb{R}^d$ denote the ith object, $\mu_h \in \mathbb{R}^d$ is the centroid of cluster h, $z_{ih} = 1$ if the ith object belongs to cluster h and $z_{ih} = 0$, otherwise. In order to minimize (1.1) the k-means algorithm involves the following steps:

1. **Initialization:** sample at random g initial centroids from the set of objects.
2. **Optimization:** alternate the following two steps until convergence.
   (a) **Assignement:** assign each object to the closest centroid cluster, according to the euclidean distance.
   (b) **Update:** compute the new centroids minimizing (1.1).

Several variants of the k-means algorithm have been developed, we can cite the k-medoids algorithm (Kaufman and Rousseeuw, 1987) that constraints the centroids to be members

of the set of objects. Both k-means and k-medoids are special cases of a more general centroid-based clustering approach known as *méthode des nuées dynamiques* (Diday, 1971). The latter makes it possible to have centroids of various forms, not necessarily vectors in $\mathbb{R}^d$ as in k-means. A more recent generalization of the k-means principle to any *Bregman divergence*—from which the euclidean distance arises as a special case—has been proposed in (Banerjee et al., 2005b).

1.4.2 Model-based approaches

Mixture model is undoubtedly a very powerful approach to clustering. It offers considerable flexibility, it is able to model different types of data—continuous, contingency, binary, etc.—and uncover various specific cluster structures. Furthermore, it allows us to make precise assumptions and understand the theoretical foundation behind some clustering methods. In fact, several partitioning clustering approaches arise as special cases from mixture model-based approaches. For instance, the k-means algorithm presented in the previous section can be derived from a mixture of Gaussian distributions, under some restrictive constraints. Intuitively, model-based clustering assumes that data is generated according to an underlying mixture of g probability distributions, usually from the same family. Objects within a cluster are assumed to be generated from the same distribution, that is each cluster is characterised by a probability distribution—called a component—with some specific parameters. The probability density function of such a mixture takes the following form:

$$f(\mathbf{X}|\Theta) = \prod_{i=1}^{n} \sum_{h=1}^{g} \alpha_h \varphi(\mathbf{x}_i|\theta_h). \tag{1.2}$$

where $\mathbf{X}$ is a $n \times d$ data matrix, its ith row (object) is denoted by $\mathbf{x}_i$. The parameters of the mixture are $\Theta = \{\alpha_1, \ldots, \alpha_g, \theta\}$, parameter α_h denotes the proportion of objects generated from component h such that $\sum_{h=1}^{g} \alpha_h = 1$, and $\varphi(\mathbf{x}_i|\theta_h)$ denotes a density/mass function of component h with some specific parameters θ_h. For instance, in the case of a mixture of Gaussian distributions the set of parameters for each component is $\theta_h = \{\mu_h, \Sigma_h\}$, where $\mu_h \in \mathbb{R}^d$, and $\Sigma \in \mathbb{R}^{d \times d}$ denote respectively the mean and covariance matrix parameters. Figure 1.3 uses the density obtained from a mixture of three Gaussian components in $\mathbb{R}^2$ to illustrate this concept of a probability mixture.

The generative process assumed by the above mixture model is as follows

1. Choose a component $h \sim Multinomial(\alpha_1, \ldots, \alpha_g)$.

2. Choose a data point $\mathbf{x}_i \sim \varphi(\mathbf{x}_i|\theta_h)$.

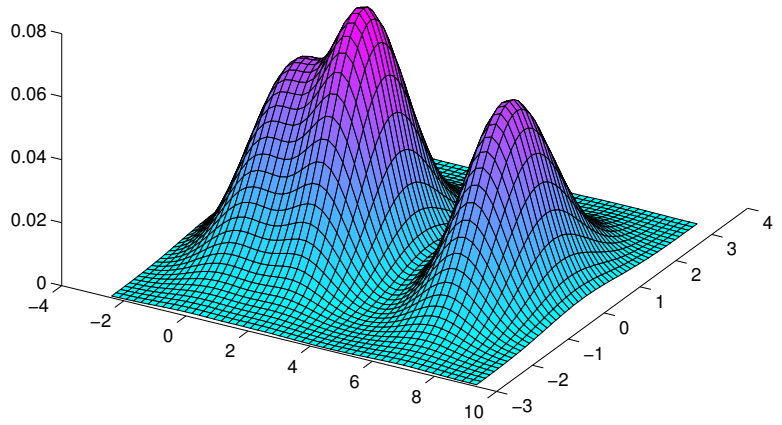


Fig. 1.3 Gaussian Mixture in $\mathbb{R}^2$

The clustering in this context consists in reversing the above generative process and estimating the model parameters Θ, given the observed data $\mathbf{X}$. To perform the latter task two main approaches are widely used in practice, namely the Maximum Likelihood (ML) and the Classification ML (CML) (Scott and Symons, 1971; Symons, 1981) approaches. The principle of the above two is to find the parameters Θ maximizing the likelihood of the observed data $\mathbf{X}$. To this end, both techniques rely on the complete data log-likelihood, because it is difficult to work directly with the likelihood function, given as follows

$$L_c(\Theta;\mathbf{X},\mathbf{Z}) = \sum_i \sum_h z_{ih} \log \alpha_h + \sum_i \sum_h z_{ih} \log \varphi(\mathbf{x}_i|\theta_h). \tag{1.3}$$

As the latent variable $\mathbf{Z}$ is unknown in practice, the Expectation-Maximization (EM) algorithm (Dempster et al., 1977) is usually used to find the ML estimates of the model parameters, while the Classification variant of EM (CEM) (Celeux and Govaert, 1992) is used under the CML approach. Regarding the clustering context, the main difference between the above two, is that the EM algorithm yields a soft clustering, i.e., each object has a probability of being generated from all clusters, and the clustering partition is obtained at the end—when the convergence is reached—by allocating each object to the most likely cluster that might have generated it, while the CEM algorithm simultaneously estimates the model parameters and the clustering partition, in an iterative scheme. Several model-based clustering approaches exist we can mention, for instance, the Gaussian (Banfield and Raftery, 1993; Celeux and Govaert, 1995), Multinomial (Jollois and Nadif, 2002; Govaert

and Nadif, 2007), Bernoulli (Govaert, 1990; Govaert and Nadif, 1996; Nadif and Govaert, 1998; McCallum et al., 1998) and von Mises-Fisher (Banerjee et al., 2005a; Gopal and Yang, 2014) mixture models for clustering respectively continuous, categorical, binary and directional data. For more details about model-based approaches to clustering the reader can refer to the books (McLachlan and Basford, 1988; McLachlan and Peel, 2004; Govaert, 2009).

1.5 Conclusion

In this chapter, we outlined the major approaches to clustering and discussed their main strengths and weaknesses. The above list is not exhaustive and other clustering approaches exist. For instance, some methods consist in combining two or more of aforementioned approaches so as to reap the benefits of each of them. We can also mention non-negative matrix factorization based-approaches that have emerged recently (Xu et al., 2003). In the rest of this thesis we focus on partitional clustering under the centroid- and model-based approaches that are closely related to each other. Our choice for these methods it largely motivated by their scalability, their strong theoretical foundations and their flexibility.

In the next chapter, we propose to tackle the problem of item recommendation by means of partitional clustering. While clustering has received very few attention in this context, as opposed to some other approaches such as matrix factorization and k nearest neighbors, we illustrate how clustering can benefit collaborative filtering systems, in terms of both scalability and effectiveness, when making proper modeling assumptions.

Chapter 2

Incremental directional clustering for a dynamic collaborative filtering system

This chapter describes a novel efficient incremental CF system based on a clustering approach. More precisely, we propose to model the user-item preferences as directional data distributed on the surface of a unit-hypersphere and develop a generalized sequential variant of the spherical k-means algorithm, which arises from a mixture of von Mises-Fihser distributions, so as to handle the frequent changes in CF filtering, i.e., submission of new ratings, update of existing ratings, appearance of new users and items. This gives rise to a very effective and scalable CF system, as illustrated through extensive experiments on several popular benchmark data sets.

2.1 Motivation

A collaborative filtering system (CF) aims at filtering huge amount of information, in order to guide users of web applications towards items that might interest them. Such a system, consists in recommending a set of personalized items for an active user, according to the preferences of other similar users. Existing methods, such as memory and Matrix Factorization (MF) approaches can achieve very good recommendation accuracy, unfortunately they are computationally very expensive. Applying such approaches to real-world applications in which users, items and ratings are frequently updated remains therefore a challenge.

In this work we aim to overcome the above limitation and develop a novel efficient incremental CF system. The proposed approach builds on the spherical k-means algorithm (`Skmeans`) (Dhillon and Modha, 2001), thereby in addition to handling frequent changes in CF data, our approach leverages the benefits of the *cosine* similarity which has been found to

be superior to several other measures, such as Euclidean distortions, in collaborative filtering (Liu et al., 2014). The `Skmeans` algorithm has strong theoretical basis and can be derived from a mixture of von Mises-Fisher distribution (Banerjee et al., 2005b). Furthermore, `Skmeans` has been successfully applied to text and gene-expression data whose characteristics are similar to those of CF.

2.2 Introduction

Nowadays the Web provides a very large amount of information, unfortunately users are unable to manage all this information correctly to reach the relevant information in a short time. The common solution to this problem is the use of recommender systems (RSs) (Bobadilla et al., 2013; Tang et al., 2013) which automatically predict the preferences of users for some given items, in order to provide them with the useful recommendations. Many RSs have been proposed and adopted by real applications such as Amazon (books recommendation) (Linden et al., 2003), YouTube (videos recommendation) (Davidson et al., 2010), Netflix (movies recommendation) (Koren, 2009), Facebook (recommendation of: users, advertising, etc.). These RSs can be divided into three overall groups:

Content-based approaches. They attempt to predict how users will rate a set of items based on their personal information (country, interests, gender, occupation, etc.) and/or the features of the items (author, type, description, field, title, tag, etc.) that they liked in the past (Pazzani and Billsus, 2007). For instance, if some users enjoyed science-fiction movies in the past, the system will suggest them other movies of the same genre. The major shortcoming of content-based approaches is that they suggest only the same kinds of items, and the sort of information mentioned above (user information and item features) is hard to collect.

Collaborative filtering (CF). It is the most often used approach in RSs. It consists in predicting items that an active user will enjoy, based on items that the people who are most similar to this user have enjoyed. Among CF systems two distinct types of approach are to be found:

- *Memory-based CF:* these approaches are based on computing similarities. User-based collaborative filtering (Bobadilla et al., 2013) looks for similarities between the active user $\mathbf{u}_a$ and all other users and tries to predict the preference of $\mathbf{u}_a$ for a set of new items, according to the preferences of the K most similar users to $\mathbf{u}_a$. Item-based collaborative filtering (Sarwar et al., 2001) consists in finding the K nearest neighbors of each item and making recommendations according to the neighborhood of items enjoyed by the

user $\mathbf{u}_a$. The most commonly used similarity measures in these approaches are *Cosine Similarity (COS)*, *Pearson Correlation Coefficient (PCC)* and *Jaccard*. In addition to these traditional measures, a number of other similarity measures have been proposed in this context: *Adjusted COS*, *Adjusted PCC*, *Constrained COS*, *Constrained PCC*, *Jaccard Mean Square Difference*, *PIP similarity* (Ahn, 2008), *NHSM similarity* (Liu et al., 2014), etc.

- *Model based CF:* these approaches begin by suggesting a model that will learn from the user/item rating matrix **U** in order to capture the hidden features of the users and items. Then, they predict the missing ratings according to this model. Many model-based CF techniques have been proposed, the most popular being those based on clustering (Ungar and Foster, 1998), co-clustering (George and Merugu, 2005; Khoshneshin and Street, 2010), matrix factorization (MF) (Sarwar et al., 2000; Koren, 2009; Koren et al., 2009; Mazumder et al., 2010; Hastie et al., 2014), non-negative matrix factorization (Zhang et al., 2006; Sindhwani et al., 2009; Gu et al., 2010), mixtures models (Marlin et al., 2007; Kim and Choi, 2014) and transfer learning approach (Li et al., 2009; Wang and Ke, 2014). There is also an important category of CF models, often referred as social model-based CF (Ma et al., 2011; Tang et al., 2013; Bobadilla et al., 2013; Delporte et al., 2013; Gao et al., 2015), which exploit additional information from social networks (i.e, trusted users, friendship graph, followers, followed, etc) to improve the quality of recommendations. All these methods can provide reasonable prediction time, since the model is learned offline.

Hybrid approaches This category of recommender systems (Burke, 2002) consists in combining several RS methods , in order to reap the benefits of several approaches simultaneously. Most of hybrid RSs combine collaborative filtering with content-based approaches (Barragáns-Martínez et al., 2010).

2.3 Related work and contribution

Collaborative filtering is the most often used approach in real-world recommender systems. Nevertheless, CF systems do have some drawbacks of their own.

Traditional CF approaches, such as matrix factorization (MF) and memory-based methods, can achieve good prediction accuracy, but their computation time rises steeply as the number of users and items increases. Furthermore, these methods need to be performed periodically (offline) in order to take into account new ratings, users and items. However, with this strategy, new information which appear between two offline computations are not considered.

As a result, applying traditional CF techniques to real-world applications such as Netflix in which the sets of users, items and ratings are frequently updated, is a non-trivial task.

To overcome the problem of computation time, incremental CF systems have been proposed. The most popular are incremental CF based on MF approaches (Sarwar et al., 2002; Han et al., 2011), incremental CF based on co-clustering (George and Merugu, 2005; Khoshneshin and Street, 2010), and incremental memory-based CF, including user (Papagelis et al., 2005) and item (Yang et al., 2012) based approaches. All these efforts have demonstrated the effectiveness of developing incremental models to provide scalable collaborative filtering systems. But often, these approaches will significantly reduce the quality of recommendations. Further, most of these approaches (except memory-based CF) do not handle all possible dynamic scenarios (i.e, submission of new ratings, update of existing ratings, appearance of new users and new items). For instance incremental CF based on singular value decomposition (Sarwar et al., 2002), do not treat the two first scenarios (mentioned above).

In this chapter we focus on the problem of computation time in CF systems. In order to overcome this drawback we propose a novel incremental CF approach, which is based on a weighted version of the online spherical k-means algorithm `OSkmeans` (Zhong, 2005). Our method is able to handle in a very short time the frequent changes in CF data, including the submission of new ratings, the update of existing ratings, the occurrence of new users and items. Below, we summarize the key contributions we make in this chapter.

- We derive a novel effective CF system, based on a weighted version of `OSkmeans` which is more suitable for CF data.
- In order to handle frequent changes in CF data, we design incremental updates, which allow to efficiently treat submissions of new ratings, updates of existing ratings, and appearance of new users and items.
- We provide extensive experiments on real-world datasets, under two scenarios: 1) static situation, where available data are kept unchanged, and 2) dynamic situation, where new information are incorporated incrementally.

Numerical experiments validate our approach. The results on several real datasets show that our method outperforms significantly state-of-the-art incremental methods in terms of both scalability and recommendation quality.

2.4 Online spherical k-means

Throughout this chapter matrices are denoted with boldface uppercase letters and vectors with boldface lowercase letters. The user-item preference matrix is denoted by $\mathbf{U} = (u_{ij})$ of size $n \times d$, the ith row (user) of this matrix is represented by a vector $\mathbf{u}_i = (u_{i1}, \ldots, u_{ip})^\top$,

where $\top$ denotes the transpose. The column j corresponds to the jth item. The partition of the set of rows into g clusters can be represented by a classification matrix $\mathbf{Z}$ of elements z_{ik} in $\{0,1\}^g$ satisfying $\sum_{h=1}^{g} z_{ih} = 1$. The notation $\mathbf{z} = (z_1, \ldots, z_n)^\top$, where $z_i \in \{1, \ldots, g\}$ represents the cluster of i, will be also used.

Many clustering algorithms have been proposed depending on the type of data and patterns to be found. In this chapter we focus on the spherical k-means (`Skmeans`) algorithm (Dhillon and Modha, 2001), and in particular on its online spherical version (`OSkmeans`). Our focus on this algorithm is motivated by the effectiveness of the *cosine* similarity in the context of high dimensional sparse data, such as CF data (Liu et al., 2014), document-term matrices and gene-expression data (Banerjee et al., 2005b). Before describing `OSkmeans`, we will first introduce the `Skmeans` algorithm.

The `Skmeans` algorithm proposed by Dhillon and Modha (2001), and available in R software (Hornik et al., 2012) is a k-means algorithm in which the objects (users) $\mathbf{u}_1, \ldots, \mathbf{u}_n$ are assumed to lie on a unit-hypersphere. The `Skmeans` algorithm originally maximizes the sum of the dot product between the elements of the data points and the g means directions characterizing the clusters. This is equivalent to maximizing the sum of the *cosine* similarity of the normalized data, and the aim at each iteration is to measure the similarity between $\mathbf{u}_i$ and center μ_h by $\langle \mathbf{u}_i, \mu_h \rangle = \mathbf{u}_i^\top \mu_h = \|\mathbf{u}_i\| \, \|\mu_h\| \cos(\mathbf{u}_i, \mu_h) = \cos(\mathbf{u}_i, \mu_h)$. Maximizing the sum of the dot products is equivalent to maximizing the sum of the *cosine* similarity on the unit hypersphere. The algorithm then maximizes the following objective function:

$$L = \sum_{i=1}^{n} \sum_{h=1}^{g} z_{ih} \cos(\mathbf{u}_i, \mu_h) = \sum_{i=1}^{n} \sum_{h=1}^{g} z_{ih} \mathbf{u}_i^\top \mu_h, \tag{2.1}$$

where $z_{ih} \in \{0,1\}$; $z_{ih} = 1$ if $\mathbf{u}_i \in h$th cluster, $z_{ih} = 0$ otherwise. `Skmeans` repeats the following two steps until convergence:

- For $i = 1, \ldots, n$, assign $\mathbf{u}_i$ to the hth cluster, where $z_i = \arg\max_h \left(\mathbf{u}_i^\top \mu_h \right), h = 1, \ldots, g$.
- Calculate $\mu_h = \frac{\sum_{i,h} z_{ih} \mathbf{u}_i}{||\sum_{i,h} z_{ih} \mathbf{u}_i||}$.

Note that, the objective function 2.1 is associated to a restricted von Mises-Fisher mixture model (Banerjee et al., 2005b). The latter will be studied in details in the next chapters.

Furthermore, the online `Skmeans` developed by Zhong (2005) uses competitive learning (Winner-Takes-All strategy) to minimize the objective function (2.1), which leads to

$$\mu_h^{new} = \frac{\mu_h + \eta \frac{\partial L_i}{\partial \mu_h}}{||\mu_h + \eta \frac{\partial L_i}{\partial \mu_h}||} = \frac{\mu_h + \eta \mathbf{u}_i}{||\mu_h + \eta \mathbf{u}_i||},$$

Algorithm 1: `OSkmeans`.

Input: n normalized objects $\mathbf{u}_i$ ($\|\mathbf{u}_i\| = 1$) in $\mathbb{R}^p$, g is the number of clusters, B is the number of batch iterations.
Output: g centroids μ_h in $\mathbb{R}^p$, the partition of the objects represented by $\mathbf{z} = (z_1, \ldots, z_n)^\top$ where $z_i \in \{1, \ldots, g\}$.
Steps:
1. Initialization: $t = 0$, random normalized cluster centroids $\{\mu_1, \ldots, \mu_g\}$;
for $b = 1$ **to** B **do**
  for $i = 1$ **to** n **do**
    2. Assignment step: for each object $\mathbf{u}_i$, compute $z_i = \arg\max_h(\mathbf{u}_i^\top \mu_h)$;
    3. Update of Centroids step: compute the winner centroid $\mu_{z_i}^{new} = \frac{\mu_{z_i} + \eta \mathbf{u}_i}{\|\mu_{z_i} + \eta \mathbf{u}_i\|}$;
    $t = t + 1$
  end for
end for

where η is the learning rate, μ_h is the closest centroid to the object $\mathbf{u}_i$, and $L_i = \sum_{h=1}^{g} z_{ih} \mathbf{u}_i^\top \mu_h$. From the `OSkmeans` method (described in Algorithm 1), we can see that each centroid is updated incrementally with a learning rate η. Zhong (2005) proposed an exponentially decreasing learning rate $\eta^\top = \eta_0 (\frac{\eta_f}{\eta_0})^{\frac{t}{n \times B}}$, where $\eta_0 = 1.0$, $\eta_f = 0.01$, B is the number of batch iterations and $t, (0 \leq t \leq n \times B)$ is the current iteration. The author showed that this decreasing learning rate is better than a flat rate ($\eta = 0.05$).

2.5 Efficient Incremental Collaborative Filtering system

In this section, we describe our collaborative filtering system `EICF`, designed to provide a high quality of recommendations with a very low computation cost. This system can be divided into three main steps: training, prediction, and incremental training. The different steps are as follows:

2.5.1 Training step

This step, consists in clustering the users into g groups. Unfortunately the traditional `OSkmeans` algorithm which has been proposed in the context of text document clustering, is not adapted for CF data. Unlike text data, the sparsity in CF is caused by unknown ratings, which requires a different handling than if the sparsity is caused by entries of "zero". To address this problem, we propose a novel variant of `OSkmeans` which is more suitable for CF. It consists in introducing user weights, in order to tackle the sparsity problem; by giving

more importance to users who provided many ratings. The resulting clusters will be highly influenced by the most useful users (i.e. users with high weights). Below we give more details about this weighted version of `OSkmeans`. Let w_i denotes the weight of the ith user, the weighted objective function of the `Skmeans` is given by:

$$L^w = \sum_{i=1}^{n} L_i^w, \text{ where } L_i^w = \sum_{h=1}^{g} w_i z_{ih} \mathbf{u}_i^\top \mu_h, \tag{2.2}$$

Thus, the corresponding update centroid for the weighted `OSkmeans` is given by:

$$\mu_h^{new} = \frac{\mu_h + \eta \frac{\partial L_i^w}{\partial \mu_h}}{||\mu_h + \eta \frac{\partial L_i^w}{\partial \mu_h}||} = \frac{\mu_h + \eta w_i \mathbf{u}_i}{||\mu_h + \eta w_i \mathbf{u}_i||}, \tag{2.3}$$

We now give an intuitive formulation of user weights. Let $\mathbf{M} = (m_{ij})$ be an $(n \times p)$ binary matrix, such that $m_{ij} = 1$ if the rating u_{ij} is available, and $m_{ij} = 0$ otherwise. Its ith row corresponds to a vector $\mathbf{m}_i = (m_{i1}, \ldots, m_{ip})^\top$ indicating which items have been rated by the ith user. Thus, we define the weight of the ith user to be proportional to the number of his available ratings as follows:

$$w_i = (\mathbf{m}_i^T \mathbb{1}) \times \sigma(\mathbf{u}_i) \tag{2.4}$$

where $\mathbb{1}$ is the vector of the appropriate dimension which all its values are 1, and $\sigma(\mathbf{u}_i)$ denotes the standard deviation of ratings provided by $\mathbf{u}_i$. We consider the standard deviation in order to give less importance to users who provide only low ratings or similarly, only high ratings (i.e. users who expressed the same preference for all items they have rated).

In the following, we focus on the initialization of the clustering process which is a crucial step. In fact, initializing the weighted `OSkmeans` by sampling g random centroids is not effective. Most of users will have rated only a few items, so there is a high probability of selecting as an initial centroid a user with only few observed ratings. Furthermore, selecting the initial centroids only from the set of users who have rated a lot of items is not a good solution, because not all structures would be detected. In order to avoid this problem we suggest the following initialization routine:

1. Generate a random partition of the users into g clusters, which can be represented by $\mathbf{z} = (z_1, \ldots, z_n)$, where $z_i \in \{1, \ldots, g\}$ represents the cluster to which the ith user belongs.
2. Estimate initial centroids as follows: let μ_{hj} denotes the jth component of the kth centroid, then we have

$$\mu_{hj} = \frac{\sum_i m_{ij} w_i z_{ih} u_{ij}}{\| \sum_i w_i z_{ih} \mathbf{u}_i \|} \tag{2.5}$$

Algorithm 2: EICF training.

Input: n normalized users $\mathbf{u}_i$ ($\|\mathbf{u}_i\| = 1$) in $\mathbb{R}^p$, K is the number of clusters and B is the number of batch iterations;
Output: K centers μ_k in $\mathbb{R}^p$, and $\mathbf{z} = (z_1, \ldots, z_n)$;
Steps:
1. Compute user weights: $w_i = (\mathbf{m}_i^T \mathbf{1}) \times \sigma(\mathbf{u}_i)$;
2. Initialization: random initialization of the partition $\mathbf{z}$, $t = 1$;
3. Estimation of the initial centroids: $\mu_{kj} = \frac{\sum_i m_{ij} w_i z_{ik} u_{ij}}{\|\sum_i w_i z_{ik} \mathbf{u}_i\|}$
for $b = 1$ **to** B **do**
  for each $\mathbf{u}_i$ **in** U **do**
    4. User assignment: compute $z_i = \arg\max_k (w_i \mathbf{u}_i^\top \mu_k)$;
    5. Centroid update: compute the winner centroid by $\hat{\mu}_{z_i} = \frac{\mu_{z_i} + \eta w_i \mathbf{u}_i}{\|\mu_{z_i} + \eta w_i \mathbf{u}_i\|}$;
    $t = t + 1$
  end for
end for

According to formula (2.5), a component μ_{hj} of an initial centroid is estimated by taking the sum over all observed ratings for the jth item, within the corresponding cluster h. Thereby, items which are rarely evaluated will be automatically penalized. Algorithm 2 describes in more details our training step.

2.5.2 Prediction step

In this step, unknown ratings are predicted according to the clustering results. However, it is difficult to make consistent predictions, even when the best clustering results are achieved, because there are so many unknown ratings in **U**. To overcome this difficulty we propose to estimate unknown ratings by a weighted average of observed ratings, as follows:

$$u_{aj} = \frac{\sum_{i=1}^{n} w_i z_{ih} \mathbf{u}_i^\top \mu_h \times u_{ij}}{\sum_{i=1}^{n} w_i z_{ih} \mathbf{u}_i^\top \mu_h}, \tag{2.6}$$

Let $\mathbf{u}_a$ denotes the active user, $h = z_a$. The key idea behind this strategy is to weight the available ratings u_{ij} according to the similarity between each user $\mathbf{u}_i$ and its corresponding centroid μ_h, and to weight by w_i, in order to give greater importance for users closest to their centroid, and respectively to give more importance for ratings provided by most important users.

The prediction equation (2.6) is attractive because it only depends on the clustering results, which means that it can be performed offline and stored in a $(g \times d)$ matrix **P**, which leads to very short prediction times.

2.5.3 Incremental training step

In the sequel, we design incremental updates, in order to handle the frequent changes in CF data. We can distinguish four main situations: 1) submission of new ratings, 2) update of existing ratings, 3) appearance of new users, 4) appearance of new items. In the following, we give the update formulas for each situation.

Submission of a new rating Let $\mathbf{u}_a$ denotes an active user who submits a new rating for an item j. The equations below, give the different incremental updates to perform in this case.

- Update the norm of $\mathbf{u}_a$: $\|\mathbf{u}_a^+\| = \sqrt{\|\mathbf{u}_a\|^2 + u_{aj}^2}$
- For each h, update the similarity between $\mathbf{u}_a$ and μ_h:

$$cos(\mathbf{u}_a^+, \mu_h) = \frac{1}{\|\mathbf{u}_a^+\|}[\|\mathbf{u}_a\| \times \mathbf{u}_a^\top \mu_h + u_{aj}\mu_{hj}],$$

- Update the weight of the active user:

$$\hat{w}_a = (\frac{w_a}{\sigma(\mathbf{u}_a)} + 1) \times \sigma(\mathbf{u}_a^+)$$

- Update the assignment of $\mathbf{u}_a$: $\hat{z}_a = \arg\max_h cos(\mathbf{u}_a^+, \mu_h)$.
- Update the corresponding centroid $\mu_{\hat{z}_a}$, by using formula (2.3)

where $\sigma(\mathbf{u}_a^+)^2 = \frac{N_a \times (\sigma(\mathbf{u}_a)^2 + \bar{u}_a^2) + u_{aj}^2}{N_a+1} - (\frac{N_a \bar{u}_a + u_{aj}}{N_a+1})^2$, thanks to König–Huygens formula, i.e., $\sigma(\mathbf{u}_a) = \sqrt{\frac{1}{N_a}\sum_j u_{aj}^2 - \bar{u}_a^2}$. The notation $\mathbf{u}_a^+$ denotes the active user $\mathbf{u}_a$ after submitting the new rating u_{ij}, N_a and $\bar{u}_a$ denote respectively, the number of ratings and the average rating of $\mathbf{u}_a$ before evaluating item j. Note that, as the centroids are stable at the end of training, the two latter incremental updates concerning the assignment of $\mathbf{u}_a$, do not need to be performed after each new rating.

Update an existing rating In this case, the active user updates an existing rating for an item j. As for the submission of a new rating, the main updates are summarized below.

- Update the norm of $\mathbf{u}_a$: $\|\mathbf{u}_a^+\| = \sqrt{\|\mathbf{u}_a\|^2 - u_{aj}^2 + \hat{u}_{aj}^2}$

- For each h, update the similarity between $\mathbf{u}_a$ and μ_h:

$$cos(\mathbf{u}_a^+, \mu_h) = \frac{1}{\|\mathbf{u}_a^+\|}[\|\mathbf{u}_a\| \times \mathbf{u}_a^\top \mu_h - u_{aj}\mu_{hj} + \hat{u}_{aj}\mu_{hj}]$$

- Update the weight of the active user: $\hat{w}_a = \frac{w_a}{\sigma(\mathbf{u}_a)} \times \sigma(\mathbf{u}_a^+)$

- Update the assignment of $\mathbf{u}_a$: $\hat{z}_a = \arg\max_h cos(\mathbf{u}_a^+, \mu_h)$.

- Update the corresponding centroid $\hat{z}_a$, by using equation (2.3)

where $\sigma(\mathbf{u}_a^+)^2 = (\sigma(\mathbf{u}_a)^2 + \bar{u}_a^2 + \frac{\hat{u}_{aj}^2 - u_{aj}^2}{N_a}) - (\bar{u}_a + \frac{\hat{u}_{aj} - u_{aj}}{N_a})^2$, $\hat{u}_{aj}$ denotes the new value substituted for the existing rating u_{aj}, and the notation $\mathbf{u}_a^+$ represents the active user after updating the known rating u_{aj}.

Appearance of new user In this situation, a new user is incorporated into the model in real time. Let $\hat{\mathbf{u}}_a$ denotes a new user. The model is incremented as follows:

- Compute the weight of $\hat{\mathbf{u}}_a$, by using equation (2.4).

- Assign $\hat{\mathbf{u}}_a$ to cluster h where $h = \arg\max_{1 \leq h' \leq g}(\frac{\hat{\mathbf{u}}_a^\top \mu_{h'}}{\|\hat{\mathbf{u}}_a\|})$.

- Update the corresponding centroid: $\hat{\mu}_h = \frac{\mu_h + \eta w_a \frac{\hat{\mathbf{u}}_a}{\|\hat{\mathbf{u}}_a\|}}{\|\mu_h + \eta w_a \frac{\hat{\mathbf{u}}_a}{\|\hat{\mathbf{u}}_a\|}\|}$.

Appearance of new item When a new item appears, it has no ratings, so there is nothing to change in the model. When a new item starts receiving ratings, handling new item, reduces to handling the submission of new ratings.

2.6 Experimental results

Hereafter, we compare our collaborative filtering system with other popular incremental CF systems, over different real-world datasets.

2.6.1 Datasets

We use three popular datasets in the field of recommender systems. The first is *MovieLens*[1] *(ML-1M)*, consisting of 1,000,209 ratings provided by 6040 users for 3952 movies (only 4.2% of ratings are observed). The second is *MovieLens (ML-100k)*, containing 100,000

[1] http://grouplens.org/datasets/movielens/

ratings given by 943 users for 1664 movies. The proportion of observed ratings in this dataset is 6.4%. The last dataset is Epinions, with 664,824 ratings from 49,290 users on 139,738 items (movies, musics, electronic products, books, ...), it is collected from the epinions web site[2]. The Epinions dataset is more than 99% sparse.

In all the above datasets the user ratings (u_{ij}) belong to the following set: $\{1,2,3,4,5,\text{NA}\}$, such that $u_{ij} = \text{NA}$, for the non observed ratings.

2.6.2 Methods and environment

We compare our Efficient Incremental CF (`EICF`) with several popular methods, namely: incremental user-based CF `IUCF` (Papagelis et al., 2005), incremental item-based CF `IICF` (Yang et al., 2012), and incremental CF based on co-clustering `COCLUST` (George and Merugu, 2005). The comparisons are made with the R package *recommenderlab* (Hahsler, 2011). All the evaluations are made under the same machine (OS: ubuntu 14.04 LTS 64-bit, Memory: 8 GiB, Processor: Intel® Core™ i7-3770 CPU @ 3.40GHz $\times$ 8). Except for comparisons on Epinions data (OS: ubuntu 14.04 LTS 64-bit, Memory: 32 GiB, Processor: Intel(R) Xeon(R) CPU E5-2620 0 @ 2.00GHz).

2.6.3 Evaluation metrics

Prediction metrics such as *Mean Absolute Error (MAE)* or *Root of Mean Square Error (RMSE)* are widely used in order to evaluate recommender systems. As mentioned by the authors in (Liu et al., 2014), these metrics are not the most relevant in the context of RS, because they do not measure the quality of the set of recommendations. In many cases, low MAE and RMSE do not necessarily equate to best user satisfaction. As illustrated in Table 2.1, whether the ratings are in the set $\{1,2,3,4,5\}$ let $\mathbf{u}_a$ be an active user who rated two items as follows: $u_{a1} = 2$ and $u_{a2} = 4$ and the three recommender systems RS1, RS2 and RS3. As reported in this table,

- RS1 predicts 1 instead of 2 for u_{a1} and 5 instead of 4 for u_{a2},
- RS2 predicts 2 for u_{a1} and 3 for u_{a2},
- and RS3 predicts 3, for both u_{a1} and u_{a2}.

In terms of MAE and RMSE, RS2 appears to be the best. However RS1 will make better recommendations; indeed, it does not recommend the first item since it predicts 1 and unlike RS2, it clearly recommend the second item; it is a good recommendation since the real rating

[2]http://www.epinions.com

Table 2.1 Comparison of MAE and RMSE achieved by three recommender systems, on a small example (numbers in bold correspond to best performances).

		item1	item2		
True ratings	active user ($\mathbf{u}_a$)	$u_{a1}=2$	$u_{a2}=4$	MAE	RMSE
Predicted ratings	RS1	1	5	1.0	1.0
	RS2	2	3	**0.5**	**0.71**
	RS3	3	3	1.0	1.0

is 4. Furthermore, in terms of MAE and RMSE, RS1 and RS3 are equivalent, nevertheless RS1 is clearly better than RS3, in terms of recommendation quality.

Another category of evaluation metrics used in RS are those known in the field of CF as recommendation metrics, and they include: *Receiver Operating Characteristic (ROC)*, *Precisions*, *Recall* and *F-measure*. These metrics measure the quality of the set of recommendations (Bobadilla et al., 2013), which makes them more appropriate for evaluating RS. Below we give a brief description of the aforementioned metrics:

Precision: is the proportion of good recommendations R_g from the total number of recommendations R_T: $precision = \frac{R_g}{R_T}$.

Recall: is the proportion of good recommendations from the number of relevant items in the testing set RI_t: $recall = \frac{R_g}{RI_t}$.

F-measure(F1): combines *precision* and *recall* as follows:

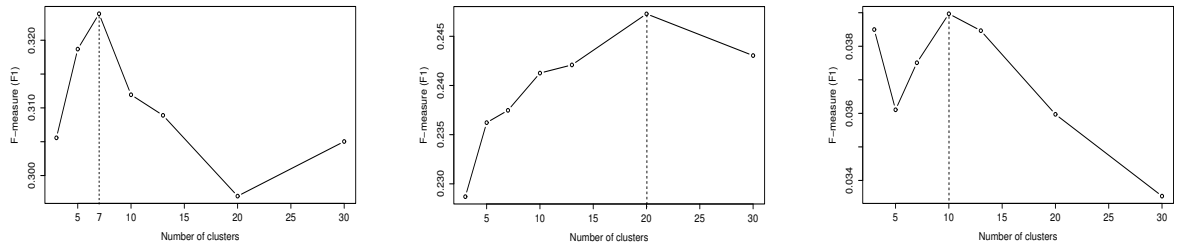
$$F1 = 2 \times \frac{precision \times recall}{precision + recall}.$$

The ROC curve is the plot of recall (*true positive rate TPR*) by the proportion of bad recommendations out of the total number of recommendations (*false positive rate FPR*).

To evaluate our CF system we focus on the quality of the recommendations and the computation time (prediction and training time).

2.6.4 Performance comparison

To measure the quality of recommendations we adopt the following split strategy: 1) We generate ten random training-test (80-20%) sets from each database. 2) In each test set only x ratings per user are given to the recommender systems and the others are retained for the evaluation. This method is known as the *Given x protocol* (*Given x*) (Breese et al., 1998; Hahsler, 2011). In the following evaluations we set $x = 10$ in the static situation and $x = 5$ in the dynamic situation. 3) Averaged results (*ROC-curves*, F-measure F1, MAE, RMSE) are presented for each RS over all the splits. Before starting the comparisons we experimentally define the best parameters for each algorithm on each dataset using the following strategy:



(a) ML-100k (b) ML-1M (c) Epinions

Fig. 2.1 Assessing the number of clusters for EICF on different datasets, using F1 metric.

each RS is performed several times with different parameters on the corresponding dataset, and the parameters that correspond to the best performance are retained. Figure 2.1 illustrates the above process for assessing the number of clusters for EICF on different datasets. At the end we obtain the following parameters.

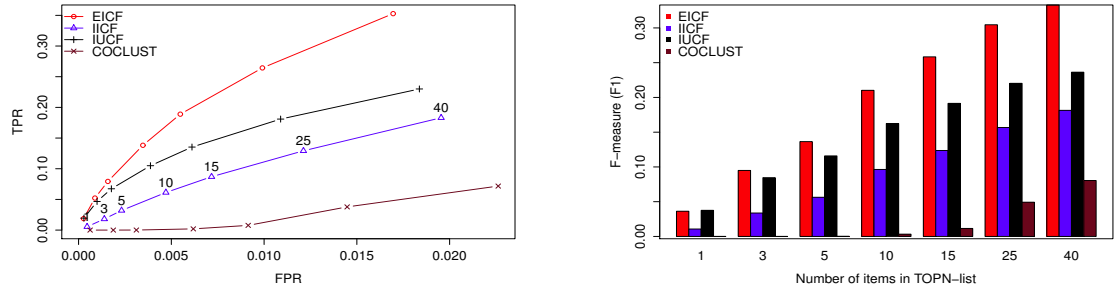
- For the ML-100k: the number of clusters $g = 7$ for our approach EICF; the number of k-nearest neighbours is set to 25 and 40 for IUCF and IICF, respectively. For COCLUST we select the same parameters as those used by *George* and *Merugu* (George and Merugu, 2005), the number of row and column clusters are both set to 3.
- For the ML-1M: the number of clusters $g = 20$ for EICF; the number of k-nearest neighbours is set to 50 for both IUCF and IICF; and for COCLUST the number of row and column clusters are both set to 3.
- For the Epinions dataset: we retain 10 clusters for EICF, the number of k-nearest neighbours is set to 120 and 135 for IUCF and IICF, respectively. The number of row clusters and column clusters for COCLUST are set to 6 and 7 respectively.

Below, we report the performances of each method on each dataset, under two situations: 1) static situation, where available data are kept unchanged, and 2) dynamic situation, where new information are incorporated incrementally.

Static situation

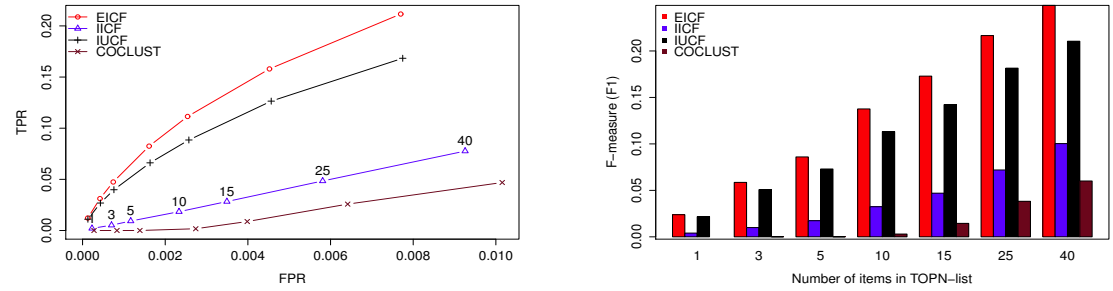
Figure 2.2 shows the *ROC-curves* and F1 comparisons of the different CF methods on the ML-100k dataset. The *ROC-curves* in Figure 2.2a are built by varying the size of the *top-N* recommendation list and representing the *TPR* and *FPR* of each *CF* for the different lists. The longest list contains 40 elements. In Figure 2.2b the *F-measure* of each method is represented over the different lists. All other *ROC-curves* and F1 comparisons are built in the same way.

From figures 2.2a and 2.2b we note that on the ML-100k dataset our method EICF provides a high quality of recommendations, thanks to our approach for alleviating the sparsity



(a) ROC-curves comparison (b) F-measure (F1) comparison

Fig. 2.2 Evaluation of several recommender systems on ML-100k dataset (static situation).



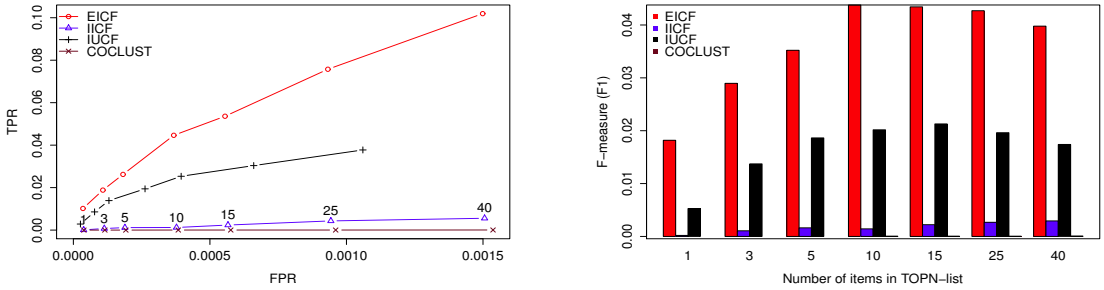
(a) ROC curves comparison (b) F-measure (F1) comparison

Fig. 2.3 Evaluation of several recommender systems on the ML-1M dataset (static situation).

problem; by introducing user weights. Moreover, both figures show that, `EICF` outperforms all other incremental approaches. We also note a low recommendation quality from `COCLUST`, which can be explained by the fact that `COCLUST` ignores the sparsity problem, and operates only on data that have actually been observed. In other words, `COCLUST` combines different summary statistics in order to compute the predictions (user, item averages, row-cluster, column-cluster and co-cluster averages) which are highly biased since they are often estimated from rare observed values. All of this lead to a low quality of recommendations.

Figure 2.3 shows that with the ML-1M dataset our method achieves a high quality of recommendations, and still outperforms all other methods, just like with the ML-100k dataset, which confirms the stability of our approach.

Figure 2.4 shows the evaluation of the different collaborative filtering systems, on the Epinon dataset in the static situation. In Figures 2.4a and 2.4b `EICF` provides the best performances. Also, both figures show an important decline in the performances of the other incremental CF systems, especially `COCLUST` and `IICF`. The low recommendation quality of



(a) ROC-curves comparison

(b) F-measure (F1) comparison

Fig. 2.4 Evaluation of several recommender systems on Epinions dataset (static situation).

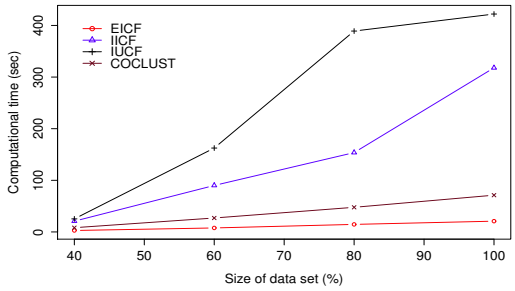


Fig. 2.5 Comparison of training + prediction time on ML-1M subsets.

the two latter approaches is due to the high rate of unknown ratings (the Epinions dataset is more than 99% sparse). For `COCLUST` we provided an explanation above, for `IICF`, the reason is that, most items have received only few ratings, thereby the estimation of item-to-item similarities is not reliable.

Computation time comparison

To compare computation time we generate four subsets from the *MovieLens* (ML-1M) database: Movie40, Movie60, Movie80 and Movie100. Each MovieX is obtained by randomly sampling $X\%$ of users and $X\%$ of items from ML-1M. For each subset and each approach, to estimate the average computation time (training + prediction time), we use the split strategy described above. To make coherent comparisons, we retain for each method the same parameters that were used for the ROC and F1 comparisons.

Figure 2.5 shows that `EICF` is the best in terms of computation time. As reported in Table 2.2, the complexity of `EICF` is just $O(gBW^*)$ for the training step and $O(1)$ for the prediction step, where B is the number of batch iterations, W^* is the total number of observed

Table 2.2 Comparison of computational times (in the worst case) in various situations. W^* denotes the number of observed ratings in $\mathbf{U}$. g and m are the number of row and column clusters, k denotes the number of neighbours for memory CF (IUCF, IICF). d^* is the number of observed ratings for a new user, n^* denotes the number of available ratings for a new item. Finally, B denotes the number of iterations

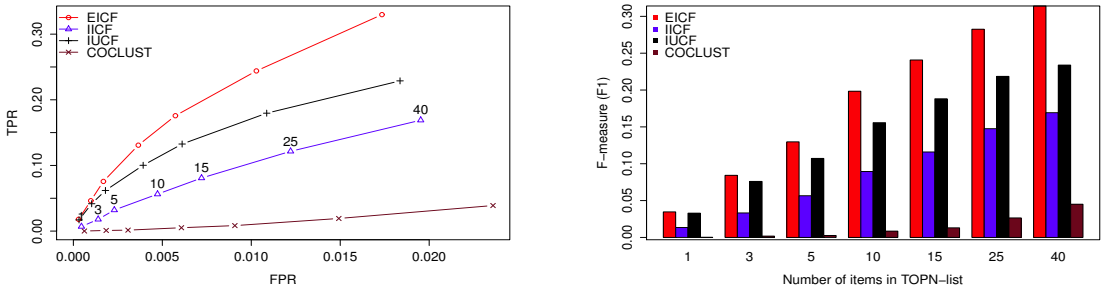
Algorithm	Static training	Prediction	Inc. train
IUCF	$O(nW^*)$	$O(k)$	$O(nd^*)$
EICF	$O(gBW^*)$	$O(1)$	$O(gd^*)$
COCLUST	$O(BW^* + ngmB + dgmB)$	$O(1)$	$O(d^*)$
IICF	$O(dW^*)$	$O(d^*)$	$O(dn^*)$

ratings in $\mathbf{U}$, and g is the number of clusters (i.e, $g \ll n$ and $g \ll d$). Moreover, Figure 2.5 shows that the computation time required by memory-based methods (IUCF and IICF) is high and increases more rapidly. This is due to the intensive training step of these approaches, which consists in computing user-to-user and item-to-item similarities for IUCF and IICF, respectively.

From Figure 2.5, we also note that the computation time of COCLUST is high, even if its complexity (training: $O(W^* + ngm + dgm)$; prediction: $O(1)$) might appear attractive. The reason is that COCLUST fails to converge rapidly, because the estimates used in this method's optimization step are biased. Furthermore, the strategy proposed by the authors in (George and Merugu, 2005), which consists in replacing the average of empty blocks (i.e, which do not contain observed ratings) by the global rating average, disturbs the convergence of the co-clustering.

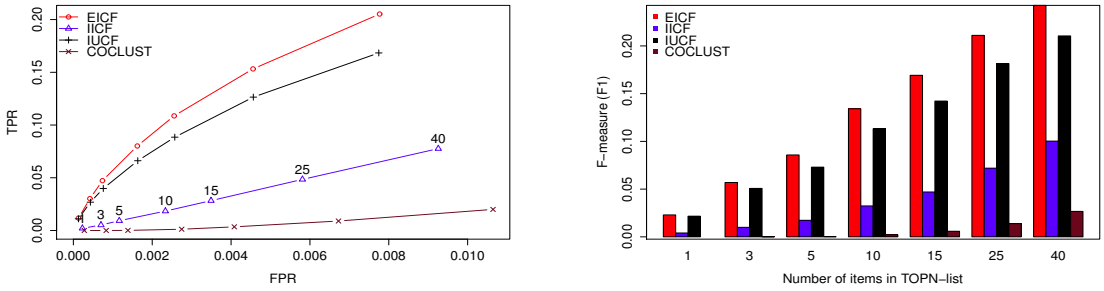
Dynamic situation

In this case new information (i.e, ratings, users and items) are incorporated incrementally, after the training step of each method. From Figures 2.6 and 2.7, we observe that on both MovieLens datasets, EICF continues to exhibit a high quality of recommendation; thanks to its efficient incremental updates for handling new data. As in the static case, EICF outperforms all other incremental methods. We also observe that COCLUST continues to provide poor performances in the dynamic situation, because it uses only partial updates for handling new changes in CF data, thereby new information are not really incorporated into the model in a dynamic way. In fact, for new users the predictions are based on item averages computed from available ratings. Thus, the recommendations for new users, are made without involving the co-clustering model.



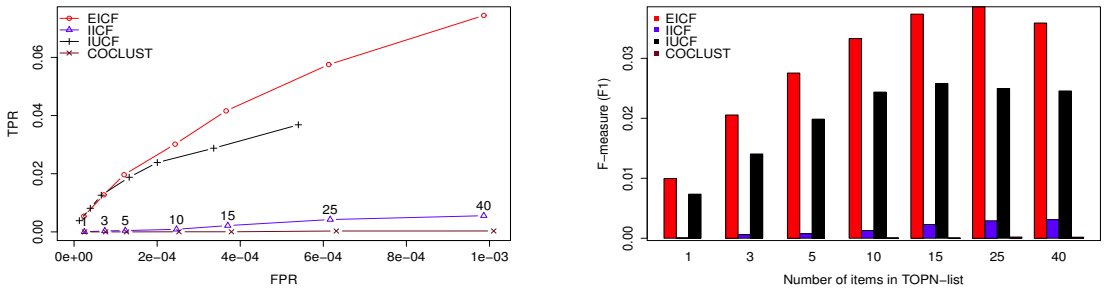
(a) ROC curves comparison (b) F-measure (F1) comparison

Fig. 2.6 Evaluation of several RS on ML-100k dataset (Dynamic situation).



(a) ROC curves comparison (b) F-measure (F1) comparison

Fig. 2.7 Evaluation of several recommender systems on ML-1M dataset (Dynamic situation).



(a) ROC curves comparison (b) F-measure (F1) comparison

Fig. 2.8 Evaluation of several CF systems on the Epinions dataset (Dynamic situation).

Figure 2.8 gives the ROC-curves and F1 comparison on Epinions dataset, in the dynamic situation. From Figures 2.8a and 2.8b we can note that `EICF` outperforms the other methods. We can also note a significant decrease in the performances of `IICF` and `COCLUST`; a poor recommendation for both methods in all the recommendation lists (TPR $\simeq 0$, for all the lists).

Table 2.3 Computation time comparison in the dynamic scenario (in sec).

datasets	CF methods.	Computation time (sec)
ML-100k	EICF	**0.51**
	IUCF	1.78
	IICF	0.88
	COCLUST	0.97
ML-1M	EICF	**2.86**
	IUCF	138.1
	IICF	7.96
	COCLUST	11.85
Epinions	EICF	**149.20**
	IUCF	4041.01
	IICF	927.22
	COCLUST	212.71

This is due to sparsity problem as mentioned above; more than 99% of ratings are unknown in the Epinions dataset.

Computation time in the dynamic situation

In Table 2.2 we reported the complexity of the incremental updates (i.e, incremental training) in the worst case, for each method. Incorporating a new user, represents the most expensive computation for EICF and IUCF, similarly incorporating a new item denotes the worst case for IICF. From Table 2.2, we observed that EICF have the best complexity in the dynamic scenario. In fact, in order for EICF to incorporate a new user $\hat{\mathbf{u}}_a$ with d^* observed ratings, it performs an assignment and update steps. The cost of the assignment is $O(gd^*)$, as g similarities need to be computed and at most d^* dimensions (items) will be involved. The cost of updating the winner centroid is $O(d^*)$, at most d^* items will be considered. Thus, the total cost of EICF for handling a new user is $O(gd^*)$ (i.e, based on $O(gd^*) + O(d^*)$).

In order to measure the computation time in the dynamic situation, 20% of users (i.e, randomly selected) in each dataset are considered as new ones. We then report in Table 2.3, the computation time required by each method, for incorporating and generating recommendations for these users. Note that IICF is favoured in this comparison, unlike the other methods; incorporating new users is not the most expensive computation for this approach.

From Table 2.3 we observe that EICF requires much less time for handling new information, than the other incremental methods, including IICF although it is advantaged. This performance rises significantly as the volume of data increases. In fact, contrary to the other methods, the complexity of EICF does not depend on the number of users and items. Therefore, EICF is more suitable than the other incremental methods, for real world applications involving large databases in which users, items and ratings are frequently updated. Note that, the computation time of COCLUST reported in Table2.3 is high, even if its complexity in the

dynamic situation (i.e, inc. train: $O(d^*)$) might appear attractive. The reason is that, this approach provides only partial updates, and the co-clustering is performed periodically to completely incorporate new information.

2.7 Conclusion and perspectives

In this chapter we presented a novel efficient and effective incremental CF system, which is based on a weighted clustering approach. To achieve high quality of recommendations, we introduced user weights into the clustering process, to lessen the effect of users who provided only few ratings. In order to address the computational time problem, we designed incremental updates, which allows our system to handle in a very short time, the frequent changes in CF data; such as submissions of new ratings, appearance of new users and items. Numerical experiments on real-world datasets demonstrate the efficiency and the effectiveness of our method which provides a better quality of recommendations than existing incremental CF systems, while requiring less computation time. In fact, unlike the other methods, the complexity of our approach does not depend on the number of users and items. Thus, `EICF` is more suitable than existing incremental approaches, for real-world applications involving huge databases, in which available information (i.e, users, items and ratings) frequently changes.

The good results of `EICF` suggest practical possible studies. A possible future work can be to improve the efficiency of `EICF`, by developing a parallel version, that can support distributed computations. On the other hand, more advanced strategies for handling the sparsity problem in CF can be considered. To this goal, two main tracks deserve to be investigated. The first consists in developing a social version of `EICF`, that can exploit information from social networks (Tang et al., 2013). The second consists in using statistical approaches for handling missing data because the sparsity in CF is due to missing ratings (Little and Rubin, 2002; Marlin et al., 2007). As earlier works shown that the `Skmeans` algorithm can be derived from mixture of von Mises-Fisher distributions (Banerjee et al., 2005b), it will be possible to handle the missing ratings statistically (Little and Rubin, 2002). Furthermore, in order to treat both users and items simultaneously, we can embed the `EICF` in a probabilistic co-clustering framework. In this way, we can use the latent block models (Govaert and Nadif, 2003, 2013) and derive a variant of the *Block EM* algorithm (Govaert and Nadif, 2005).

Finally, our approach could imply the extension of NMF approaches (widely used in CF) to the context of dynamic CF. Indeed, it has been shown that under some restrictions NMF can be interpreted as a clustering approach (Xu et al., 2003). Hence, in the case of

two factors NMF, the first factor indicates the degree in which a user belongs to a cluster and the second one contains cluster centroids. Thus, based on the incremental updates of EICF (mainly related to users assignments and centroids updates), one can similarly design incremental updates for the latent factors in the NMF context.

Chapter 3

Directional Clustering with Social-Network information for Recommendation

Collaborative Filtering models are severely challenged by the high dimensionality and sparsity of user-item preference matrices. Recently, several works have shown that incorporating information from social networks, such as friendship and trust relationships, into traditional CF alleviates the sparsity related issues and yields a better recommendation quality in most cases. More interestingly, even with comparable performances, social-based CF is more beneficial than traditional CF; the former makes it possible to provide recommendations for cold start users. This chapter presents a novel CF model that leverages information from social networks in order to improve recommendations. While existing social CF approaches are based on popular modelling assumptions such as Gaussian or Multinomial, our model is based on the von Mises-Fisher assumption that turns out to be more adequate than the aforementioned assumptions, for high dimensional sparse data. Setting the model parameters under the maximum likelihood approach, we derive an effective social-based CF system. Empirical results on several real-world datasets provide strong support for the advantages of the proposed model.

3.1 Introduction

Although memory- and model-based CF approaches introduced in the previous chapter, often denoted as traditional CF methods, constitute an important contribution to CF and can offer a good recommendation accuracy, these techniques are still seriously challenged by the

characteristics of collaborative filtering data, i.e. high dimensionality and sparsity. Over the last few years, with the advent of social networks, social-based collaborative filtering has emerged as a new promising technique to alleviate the sparsity related issues. Such an approach consists in using information from online social networks—usually friendship and/or trust information—to improve recommendations. More intuitively, social-based CF approaches are based on the assumption that, for making a good recommendation, not only the user's expressed preferences are important but also the user's social interactions. This is natural, since in real life people often turn to their friends, to ask for a nice movie to watch, an interesting book to read, a good restaurant, etc. By taking into account this real-life behaviour, social-based CF approaches make more realistic recommendations and are therefore expected to offer better performances than traditional CF methods. More interestingly, even with comparable recommendation accuracy, social-based methods are more beneficial than traditional approaches, in that they can make recommendations for cold start users—with very few expressed preferences or none at all. In fact, Social CF models exploit the social interactions of cold-start users to provide them with useful recommendations.

As mentioned in the beginning of this thesis and the previous chapter, apart from being high dimensional and sparse , CF data are also directional in nature. In fact, it seems better to model CF data as directional data distributed on the surface of a unit-hypersphere (Mardia and Jupp, 2000; Banerjee et al., 2005b), and to use directional measures, such as the *cosine* similarity, instead of euclidean distortions to assess the similarities between users/items.

Existing social CF models, however, are based on popular modelling assumptions, such as Gaussian or Multinomial, which are inadequate for directional data lying on the surface of a unit-hypersphere. Hence, it seems natural to question whether it is possible to leverage both the users' social interactions and the directional properties of CF data, simultaneously. In this chapter, we provide an answer to this question: we develop a novel social CF model which is based on the von Mises-Fisher (vMF) assumption, that arises naturally for directional data lying on the surface of a unit-hypersphere. The proposed model successfully integrates a directional measure, namely the *cosine* similarity, into a social CF model. This makes it possible to achieve a high recommendation accuracy, as illustrated in our experiments. To the best of our knowledge the work we present is the first social-based CF approach that accounts for the directional characteristics of collaborative filtering data.

The remainder of the chapter is organized as follows. Section 2 is devoted to present related work. In Section 3, we review the vMF mixture model, then we propose to generalize the above model to account for social interactions among users and derive a scalable Generalized Expectation-Maximization (GEM) algorithm for inference and parameter estimation (Section 4). Finally, we evaluate our contribution on real data sets.

3.2 Related work

The work we present here is related to two main topics, namely CF approaches accounting for social-network information, and vMF mixture models from directional statistics (Mardia and Jupp, 2000; Banerjee et al., 2005b).

Over the last few years, several social-based collaborative filtering approaches have been proposed, most of which are based on Probabilistic Matrix Factorization (PMF) (Ma et al., 2008, 2009; Jamali and Ester, 2010; Ma et al., 2011). The key idea behind these approaches is to make the latent preference factor of each user close to that of his/her direct neighbors in the user-user social graph, so as to capture the influence between friends. In (Ma et al., 2008) the authors built a MF model that connects the user-item preference matrix with the user-user social graph through a shared user latent factor. The same authors in (Ma et al., 2009) proposed an approach that fuses a MF model on the user-item matrix with a MF on the user-user graph, then predicts unknown preferences by combining the ratings resulting from both models. Based on the above works, in (Jamali and Ester, 2010) the authors proposed `SocialMF` another matrix factorization-based method that accounts for trust propagation. Ref. (Ma et al., 2011) proposed to add a regularization, into the traditional PMF (Salakhutdinov and Mnih, 2008), so as to bring the latent factors of socially connected users closer to each other. In (Yang et al., 2013) the authors proposed three trust MF-based models that consider different aspects of trust information. The first variant reflects that the preference of a user for an item is influenced by the preferences of his/her trustees on that item, the second reflects that the behaviour of a user will influence that of his/her trusters and the third is a combination of the above two during the prediction phase. More recently, in (Guo et al., 2015) the authors proposed `TrustSVD` an extension of the well known SVD++ (Koren, 2008) method that accounts for social trust information. Ref. (Chaney et al., 2015) proposed Social Poisson Factorization. In addition to learn the user and item latent factors, as in traditional MF, this method introduces a third latent factor that reflects how much each user is influenced by his/her direct neighbors in the social network. Then, the preference of a user for an item is explained by combining the three factors above.

All these efforts demonstrated that, considering information from social networks in CF makes it possible to alleviate the sparsity related issues and generate more realistic and accurate recommendations. Nevertheless, existing approaches to social CF are based on popular assumptions such as Gaussian, Multinomial or Poisson, and therefore do not account for the aforementioned directional characteristics of CF data. In this chapter, we aim to address this limitation by building a novel model-based social CF approach, that leverages the social interactions among users as well as the directional properties of CF data. More precisely, the proposed model is based on a mixture of von Mises-Fisher distributions,

and successfully integrates a directional measure, the *cosine* similarity, into a social CF framework.

The vMF distribution is a continuous probability distribution, on a unit-hypersphere, from directional statistics (Mardia and Jupp, 2000). It focuses on the directions of objects and measures the distance between them using the *cosine* similarity. Most of the earlier works using vMF distributions focused on low dimensional data, i.e., 2- or 3- dimensional data (McLachlan and Peel, 2004), due to the difficulties related to the estimation of the concentration parameter kappa, which involves the inversion of a ratio of Bessel functions. A notable contribution is the mixture of vMF distributions *movMFs* (Banerjee et al., 2005b) for clustering high dimensional sparse data. Banerjee et al. (2005b) derived an EM-based solution for inference and parameter estimation, and they proposed an accurate approximation to estimate the concentration parameter κ for a high dimensional vMF distribution. Since this contribution, different vMF-based models for high dimensional sparse data have been proposed. For instance (Reisinger et al., 2010) proposed a topic model based on a mixture of vMF distributions. More recently, for text data clustering, (Gopal and Yang, 2014) proposed a full Bayesian formulation of *movMFs* and developed two novel variants of *movMFs*, namely hierarchical and temporal. Le and Lauw (2014) proposed a vMF-based model for the semantic visualization of high dimensional sparse text data.

3.3 Preliminaries

In this section, we shall describe the mixture of von Mises-Fisher distributions proposed by Banerjee et al. (2005b), but first we review the von Mises-Fisher (vMF) distribution, which is well known in directional statistics (Mardia and Jupp, 2000).

3.3.1 The Multivariate von Mises-Fisher (vMF) Distribution

A d dimensional vMF (d-vMF) distribution, i.e, $d \geq 2$ is a continuous probability distribution on a unit hypersphere $\mathbb{S}^{d-1}$. Thus, if a d dimensional data point $\mathbf{x}_i$ on $\mathbb{S}^{d-1}$, i.e, $\mathbf{x}_i \in \mathbb{R}^d$ and $\|\mathbf{x}_i\| = 1$ follows a d-vMF distribution, its probability density function is given by:

$$f(\mathbf{x}_i|\mu,\kappa) = c_d(\kappa)\exp^{\kappa\mu^T\mathbf{x}_i}, \tag{3.1}$$

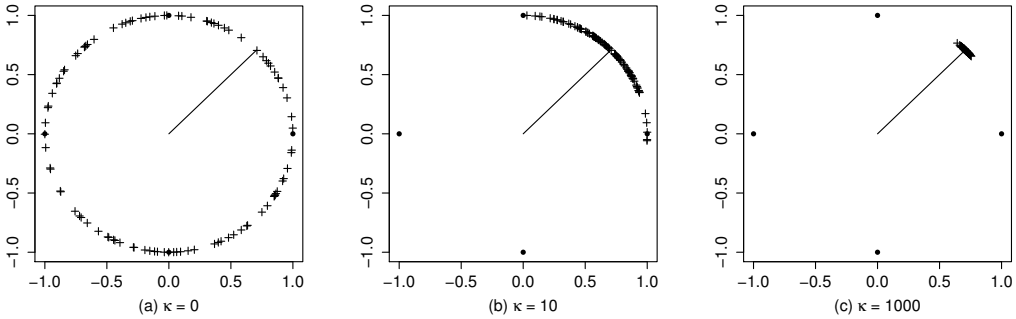


Fig. 3.1 Three sample of 100 points according to three 2-vMF distributions, with different concentration parameters, and the same mean direction $\mu = (1/\sqrt{2}, 1/\sqrt{2})$. The segment starting from the circle centre denotes the mean direction of each vMF distribution.

where μ is the mean direction (centroid) parameter and κ denotes the concentration parameter, such that $\|\mu\| = 1$ and $\kappa \geq 0$. The normalization term $c_d(\kappa)$ is equal to:

$$c_d(\kappa) = \frac{\kappa^{\frac{d}{2}-1}}{(2\pi)^{\frac{d}{2}} I_{\frac{d}{2}-1}(\kappa)} \tag{3.2}$$

where $I_r(\kappa)$ represents the modified Bessel function of the first kind and order r. In the vMF distribution the parameter κ controls the concentration of data points $\mathbf{x}_i \in \mathbb{S}^{d-1}$ following (3.1), around the mean direction μ. Thus, $f(\mathbf{x}_i|\mu,\kappa)$ reduces to the uniform density on $\mathbb{S}^{d-1}$ for $\kappa = 0$, and it is uni-modal if $\kappa > 0$. In particular, when $\kappa \to \infty$, $f(\mathbf{x}_i|\mu,\kappa)$ tends to a point density. Figure 3.1 illustrates the impact of κ, by presenting three sample of 100 points on the unit circle (i.e $d = 2$) according to three 2-vMF distributions with the same centroid μ, but with different values of κ. For more details on the vMF distribution, the reader can refer to (Mardia and Jupp, 2000; Dhillon and Sra, 2003).

3.3.2 The mixture of von Mises-Fisher Distributions (*movMFs*)

Now, we review the mixture model of von Mises-Fisher distributions for clustering directional data distributed on a unit hypersphere (Banerjee et al., 2005b). In this model, the data points $\mathbf{x}_1, \ldots, \mathbf{x}_n$ are assumed to be generated from a mixture of g vMF distributions with a set of unknown parameters Θ. The density function of this mixture takes the following form:

$$f(\mathbf{x}_i|\Theta) = \sum_h \alpha_h f_h(\mathbf{x}_i|\mu_h,\kappa_h), \tag{3.3}$$

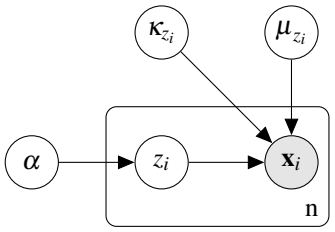


Fig. 3.2 *movMFs* as a graphical model.

where $\Theta = \{\mu_1, \ldots, \mu_g, \alpha_1, \ldots, \alpha_g, \kappa_1, \ldots, \kappa_g\}$, μ_h and κ_h represent the centroid and the concentration parameters of the h^{th} component, respectively. Each parameter α_h denotes the proportion of points $\mathbf{x}_i$ generated from the h^{th} component, such that $\sum_h \alpha_h = 1$ and $\alpha_h > 0$, $\forall h \in \{1, \ldots, g\}$. The generative process of this model is summarized below:

1. Choose a component $h \sim Multinomial(\alpha_1, \ldots, \alpha_g)$.
2. Choose a data point $\mathbf{x}_i$ on $\mathbb{S}^{d-1} \sim f_h(\mathbf{x}_i|\mu_h, \kappa_h)$.

The graphical model of the vMF mixture model (*movMFs*) is depicted in figure 3.2, and the corresponding log-likelihood function takes the following form

$$L(\Theta; \mathbf{X}) = \sum_i \log \left(\sum_h \alpha_h f_h(\mathbf{x}_i|\mu_h, \kappa_h) \right), \tag{3.4}$$

As the optimization of the above function is intractable, we rely on the "complete" data likelihood given by

$$\mathcal{L}(\mu, \alpha, \kappa; \mathbf{X}, \mathbf{z}) = \prod_i \alpha_{z_i} (c_d(\kappa_{z_i}) \exp^{\kappa_{z_i} \mu_{z_i}^\top \mathbf{x}_i}), \tag{3.5}$$

where $\mathbf{z}$ is the latent variable which is assumed to be known, i.e, $z_i = h$ if $\mathbf{x}_i$ is generated from the h^{th} component. Using the classification matrix $\mathbf{Z}$, the corresponding complete data log-likelihood takes the following form:

$$L_c(\Theta; \mathbf{X}, \mathbf{Z}) \quad = \quad \sum_h z_{.h} \log \alpha_h + \sum_h z_{.h} \log c_d(\kappa_h) + \sum_{i,h} z_{ih} \kappa_h \mu_h^\top \mathbf{x}_i \tag{3.6}$$

where $z_{.h}$ denotes the cardinality of cluster h.

As the latent variable $\mathbf{z}$ is unknown in practice, the authors in (Banerjee et al., 2005b) proposed to use the EM algorithm to obtain the maximum likelihood estimates for the parameters Θ. The E-step finds the conditional expectation of the missing variable $\mathbf{z}$ given the current estimated parameters $\Theta^{(t)}$ and the observed data, i.e. $\tilde{z}_{ih} = \mathbb{E}(z_{ih} = 1|\mathbf{x}_i, \Theta^{(t)})$.

The M-step finds the new parameters $\Theta^{(t+1)}$ maximizing the expectation of the complete data log-likelihood (3.6) subject to the constraints $\sum_h \alpha_h = 1$, $\|\mu_h\| = 1$ and $\kappa_h > 0$. This procedure leads to the soft vMF clustering algorithm (Banerjee et al., 2005b), denoted in this chapter as `movMF`.

Note that if we impose the following constraints on the parameters: equality of proportions $\alpha_1 = \ldots = \alpha_h$ and concentration $\kappa_1 = \ldots = \kappa_h$ parameters, the maximization of $L_c(\Theta; \mathbf{X}, \mathbf{Z})$ reduces to the maximization of the spherical k-means criterion (Dhillon and Modha, 2001; Banerjee et al., 2005b)

$$\sum_{i,h} z_{ih} \kappa_h \mu_h^\top \mathbf{x}_i = \sum_{i,h} z_{ih} < \mu_h, \mathbf{x}_i >= \sum_{i,h} z_{ih} \cos(\delta_{ih})$$

where $<,>$ denotes the scalar product and δ_{ih} is the angle between both vectors $\mathbf{x}_i$ and μ_h. So when relying on a vMF mixture model, the *cosine* similarity is underlying. In fact, the well known spherical k-means algorithm, using *cosine* similarity instead of euclidean distortions, arises as special case from the `movMF` algorithm considered in this chapter, when we enforce some restrictive constraints (Banerjee et al., 2005b).

3.4 Social von Mises-Fisher Mixture Model

We now propose *Social-movMFs* a novel model that leverages simultaneously the benefits of the vMF-based modeling and social information. Specifically, we propose to generalize the mixture of vMF distribution *movMFs*, presented above, to account for the social interactions among users. The intuition behind our model is to bring the distributions over clusters of socially connected users closer to each other. To this end, inspired from previous works on learning using manifold regularization (Zhu and Lafferty, 2005; Belkin et al., 2006; Cai et al., 2008; Mei et al., 2008; He et al., 2011), we propose to smooth the posterior probabilities $\tilde{z}_{ih}$ based on the user-user graph. Posterior smoothness can be achieved by using any adequate function, here we adopt the quadratic energy function—also denoted as the graph harmonic function—(Zhu and Lafferty, 2005) defined as follows in our case

$$R(\mathcal{T}) = \frac{1}{2} \sum_h \sum_i \sum_j \tau_{ij} (\tilde{z}_{ih} - \tilde{z}_{jh})^2 \tag{3.7}$$

where $\tau_{ij} = 1$ if users i and j are socially connected and $\tau_{ij} = 0$, otherwise, $\mathcal{T} = (\tau_{ij})$ is the adjacency matrix of the user-user social graph. The above function is minimized if all socially connected users exhibit similar posterior distributions over clusters. Recall that our purpose is to force socially connected users to exhibit similar distributions over clusters. This

objective can be achieved by regularizing the log-likelihood (3.4) by function (3.7). By doing so in an adequate manner, we obtain the following regularized log-likelihood:

$$L_r(\Theta;\mathbf{X},\mathcal{T}) = L(\Theta;\mathbf{X}) - \lambda R(\mathcal{T}) \tag{3.8}$$

where $R(\mathcal{T})$ plays the role of the regularization term that enforces smoothness of the posterior probabilities on the social network, and λ is the regularization parameter that controls the degree of smoothness. Observe that the complete data log-likelihood (3.6) of *movMFs* arises from (3.8) as a special case when $\lambda = 0$. The corresponding regularized complete data log-likelihood can be obtained by substituting $L_c(\Theta,\mathbf{X},\mathbf{Z})$ for $L(\Theta,\mathbf{X})$ in (3.8).

It is worth noting that the regularized log-likelihood (3.8) is penalized by users who are socially connected and who exhibit substantially different distributions over clusters. Thus, as opposed to the *movMFs* model, *Social-movMFs* accounts for interactions among users.

3.4.1 Maximum Likelihood estimates

In order to obtain the maximum likelihood estimates of the model parameters we rely on the *Generalized* EM algorithm (Dempster et al., 1977; McLachlan and Krishnan, 2007). The E-step is to estimate the posterior probabilities $\tilde{z}_{ih}$, given the current estimated parameters $\Theta^{(t)}$ and the observed data $\mathbf{X}$, as follows (Neal and Hinton, 1998):

$$\tilde{z}_{ih} = \mathbb{E}(z_{ih} = 1|\mathbf{x}_i,\Theta^{(t)}) \propto \alpha_h^{(t)} f_h(\mathbf{x}_i|\mu_h^{(t)},\kappa_h^{(t)}). \tag{3.9}$$

The M-step finds the new parameters $\Theta^{(t+1)}$ maximizing or increasing the expectation of the complete data log-likelihood which is given by

$$\begin{aligned} Q(\Theta,\Theta^{(t)}) &= \mathbb{E}\left(L_c(\Theta;\mathbf{X},\mathbf{Z})|\mathbf{X},\Theta^{(t)}\right) \\ &= \sum_{i,h}\tilde{z}_{.h}\log\alpha_h + \sum_h \tilde{z}_{.h}\log c_d(\kappa_h) + \sum_{i,h}\tilde{z}_{ih}\kappa_h\mu_h^\top\mathbf{x}_i \end{aligned} \tag{3.10}$$

where $\tilde{z}_{.h} = \sum_i \tilde{z}_{ih}$. The above optimization scheme yields the `movMF` algorithm providing parameter estimation for the classical *movMFs* model (Banerjee et al., 2005b). In our case, the purpose is to take into account social interactions among users. Thus, instead of maximizing expression (3.10), we maximize the following regularized expected complete data log-likelihood (or equivalently the expectation of the regularized complete data log-

likelihood):

$$\begin{aligned} Q_r(\Theta,\Theta^{(t)}) &= Q(\Theta,\Theta^{(t)}) - \lambda R(\mathcal{T}) \\ &= \sum_{i,h} \tilde{z}_{.h} \log \alpha_h + \sum_h \tilde{z}_{.h} \log c_d(\kappa_h) + \sum_{i,h} \tilde{z}_{ih} \kappa_h \mu_h^\top \mathbf{x}_i \\ &- \frac{\lambda}{2} \sum_h \sum_i \sum_j \tau_{ij} (\tilde{z}_{ih} - \tilde{z}_{jh})^2 \end{aligned} \tag{3.11}$$

Note that the direct maximization of expression (3.11) is intractable due to the introduction of the regularization term—the M-step of EM does not have a closed-form solution. Fortunately, in the GEM algorithm it is sufficient to find a better Θ at each iteration, i.e. we choose $\Theta^{(t+1)}$ so that $Q_r(\Theta^{(t+1)},\Theta^{(t)}) \geq Q_r(\Theta^{(t)},\Theta^{(t)})$. Hence, following the strategy described in (Cai et al., 2008; He et al., 2011), which is closely related to the optimization scheme proposed in (Zhu and Lafferty, 2005) in the context of semi-supervised learning, we derive an efficient M-step that is guaranteed to increase (3.11) at each iteration. The key idea is to optimize the different parts of (3.11) separately so as to increase $Q_r(\Theta^{(t)},\Theta^{(t)})$. More precisely, we first minimize the regularization term, as it depends only on the posterior probabilities $\tilde{z}_{ih}$. This step yields the smoothed posteriors on the user-user graph. It is obvious that the smoothed posterior $\tilde{z}_{ih}$ minimizing $R(\mathcal{T})$ is given by $\tilde{z}_{ih} = \frac{\sum_j \tau_{ij} \tilde{z}_{jh}}{\sum_j \tau_{ij}}$. This minimization scheme, however, can lead to a strong smoothing, where the new posteriors are substantially far from the original ones. Hence, for a better control of the smoothing process one should decrease $R(\mathcal{T})$ gradually, instead of its direct minimization. This can be done by the Newton-Raphson method as follows:

$$\tilde{z}_{ih} := \tilde{z}_{ih} - \gamma \frac{R'(\mathcal{T})}{R''(\mathcal{T})} = (1-\gamma)\tilde{z}_{ih} + \gamma \frac{\sum_j \tau_{ij} \tilde{z}_{jh}}{\sum_j \tau_{ij}} \tag{3.12}$$

where R', R'' denote the first and second derivative of the regularization term relative to $\tilde{z}_{ih}$, and $\gamma \in [0,1]$ is the Newton-Raphson's step parameter. In our context, we can think of γ being the level of smoothing. If $\gamma = 0$ then no smoothing is performed, and if $\gamma = 1$ then the smoothed posterior distribution of each user is completely specified by the posterior distributions of his neighbors in the social graph.

Once the smoothing step is done, the next step consists in maximizing the expectation of the complete data log-likelihood Q relative to the parameters Θ, which yields the following

update formulas (Banerjee et al., 2005b):

$$\hat{\alpha}_h = \frac{\sum_i \tilde{z}_{ih}}{n}, \tag{3.13a}$$

$$\hat{\mu}_h = \frac{\mathbf{r}_h}{\|\mathbf{r}_h\|} \quad \text{where} \quad \mathbf{r}_h = \sum_i \tilde{z}_{ih}\mathbf{x}_i, \tag{3.13b}$$

$$\hat{\kappa}_h \approx \frac{\bar{r}_h d - \bar{r}_h^3}{1-\bar{r}_h^2} \quad \text{where} \quad \bar{r}_h = \frac{I_{d/2}(\hat{\kappa}_h)}{I_{d/2-1}(\hat{\kappa}_h)} = \frac{\|\mathbf{r}_h\|}{\sum_i \tilde{z}_{ih}}. \tag{3.13c}$$

Notice that, an exact estimation of the concentration parameter $\hat{\kappa}_h$ implies to inverse a ratio of Bessel functions, which has not a close form expression. To overcome this difficulty Banerjee et al. (2005b) proposed the efficient approximation (3.13c), which is suitable for high dimensional datasets. Furthermore, Tanabe et al. (2007) showed theoretically that the above approximation lies in the interval in which the exact ML estimates of κ_h exists. More accurate approximation of κ_h can be reached by using iterative methods (Tanabe et al., 2007; Sra, 2012), the latter, however, are less suitable in high dimensions, since they involve an extensive computation of a ratio of Bessel functions.

It is worth nothing that the M-step just described does not necessarily increase the regularized log-likelihood function (3.8), due to the smoothing step. In order to address this issue, we adopt the same strategy as in (Cai et al., 2008; He et al., 2011). After each M-step we check if the regularized log-likelihood function has been decreased, then we decrease the smoothing parameter γ and perform again the M-step. Alternating the above E-step and optimization scheme (M-step) constitutes our soft social movMF algorithm—denoted as `Soc-movMF` in the rest of the chapter—, which is described in more details by Algorithm 3.

It can be shown that the computational complexity of `Soc-movMF` is $O(g \cdot nr + g \cdot ns)$ per iteration, which scales linearly with the number of observed relations ns in the social network and observed ratings nr in the user-item matrix. In practice, we have $nr \ll n \times d$ and $ns \ll n \times n$, thereby `Soc-movMF` is very efficient and suitable for large datasets.

Fitting the parameters of the proposed model *Social-movMFs*, to the user-item matrix, using `Soc-movMF` constitutes the training component of our CF system. Once this step is done, the missing ratings of the ith user can be easily predicted as follows

$$\hat{\mathbf{x}}_i = \frac{\sum_h \tilde{z}_{ih}\mu_h}{\|\sum_h \tilde{z}_{ih}\mu_h\|}. \tag{3.14}$$

Algorithm 3: `Soc-movMF`.

Input: $\mathbf{X}$ ($\mathbf{x}_i \in \mathbb{S}^{d-1}$) the rating matrix of size ($n \times d$),
$\mathcal{T}$ the adjacency matrix of the social network, g the number of clusters, λ the regularization parameter.
Output: $\tilde{\mathbf{Z}}$, Θ;
1. Random initialization: $\Theta \leftarrow \Theta^{(0)}$; $t \leftarrow 0$;
repeat
  2. Expectation step of GEM:
  for $i = 1$ **to** n **do**
    for $h = 1$ **to** g **do**
      $\tilde{z}_{ih}^{(t)} \leftarrow \frac{\alpha_h f_h(\mathbf{x}_i|\mu_h,\kappa_h)}{\sum_\ell \alpha_\ell f_\ell(\mathbf{x}_i|\mu_\ell,\kappa_\ell)}$
    end for
  end for
  3. Maximization step of GEM:
  smooth $\leftarrow$ TRUE; $\gamma \leftarrow 0.9$;
  while smooth **do**
    3.1 Smooth the posterior probabilities
    $\tilde{z}_{ih}^{(t+1)} \leftarrow \tilde{z}_{ih}^{(t)}$; $\forall i,h$
    $\tilde{z}_{ih}^{(t+1)} \leftarrow (1-\gamma)\tilde{z}_{ih}^{(t+1)} + \gamma \frac{\sum_j \tau_{ij}\tilde{z}_{jh}^{(t+1)}}{\sum_j \tau_{ij}}$; $\forall i,h$
    3.2. Compute the new parameters $\Theta^{(t+1)}$
    for $h = 1$ **to** g **do**
      $\hat{\alpha}_h \leftarrow \frac{\sum_i \tilde{z}_{ih}^{(t+1)}}{n}$
      $\hat{\mu}_h \leftarrow \frac{\mathbf{r}_h}{\|\mathbf{r}_h\|}$ with $\mathbf{r}_h = \sum_i \tilde{z}_{ih}^{(t+1)}\mathbf{x}_i$
      $\hat{\kappa}_h \leftarrow \frac{\bar{r}_h d - \bar{r}_h^3}{1-\bar{r}_h^2}$ with $\bar{r}_h = \frac{I_{d/2}(\hat{\kappa}_h)}{I_{d/2-1}(\hat{\kappa}_h)} = \frac{\|\mathbf{r}_h\|}{\sum_i \tilde{z}_{ih}^{(t+1)}}$.
    end for
    3.3. Compute the regularized log-likelihood
    $L_r(\Theta^{(t+1)};X) \leftarrow L(\Theta^{(t+1)};X) - \lambda R(\mathcal{T})$
    if $L_r(\Theta^{(t+1)};X) < L_r(\Theta^{(t)};X)$ **then**
      3.4 Decrease the smoothing parameter γ
      $\gamma \leftarrow \gamma \times \gamma$
    else
      smooth $\leftarrow$ FALSE; $t \leftarrow t+1$;
    end if
  end while
until convergence

Observation It is worth noting that the proposed model *Social-movMFs* allows the propagation of social information, between users, through the smoothing process (see step 3.1 in Algorithm 3). A property which is desirable since it may help to alleviate the sparsity related issues in both the rating matrix and social network, as it has already been emphasized in previous works (Ma et al., 2008, 2009; Jamali and Ester, 2010).

3.5 Experimental study

To show the benefits of our approach, we conduct extensive experiments on several real-world datasets, in which we benchmark `Soc-movMF` against several strong competing social-CF methods, namely `SoRec` (Ma et al., 2008), `RSTE` (Ma et al., 2009), `SocialMF` (Jamali and Ester, 2010), `SoReg` (Ma et al., 2011), `TrustMF` (Yang et al., 2013) and `TrustSVD` (Guo et al., 2015). In order to illustrate the advantage of leveraging information from social networks, we also compare `Soc-movMF` with the traditional *movMFs*-based clustering algorithm `movMF` (Banerjee et al., 2005b). Notice that `movMF` arises as a special case from `Soc-movMF` when $\lambda = 0$. For all competing methods we used LibRec[1], except for `movMF` we used our implementation.

Note that previous works, in the context of social CF, established empirically that the aforementioned competing methods perform better than several traditional CF approaches, such as `SVD++` (Koren, 2008), Matrix Factorization- and probabilistic MF-based methods, therefore we do not consider these approaches in our experiments.

3.5.1 Datasets

We selected five popular benchmark real-world datasets, including both the user-item preferences and social relationships between users, namely FilmTrust, CiaoDVD, Ciao-280k, Flixster and Epinions.

- **Flixster**[2]: is a social rating dataset crawled by Jamali and Ester (Jamali and Ester, 2010) from the flixster.com website. The latter is a social movie website, where users can watch, buy, share, rate and review movies. Each user can create his/her own social network by adding users into his/her friend list.

[1] http://www.librec.net/
[2] http://www.cs.ubc.ca/ jamalim/datasets/

- **FilmTrust**[3]: is a dataset crawled by Guo et al (Guo et al., 2013) from the FilmTrust website. FilmTrust, as Flixster, is a movie rating and sharing community. Unlike the Flixster datasets, the social interactions between users are directed in FilmTrust.
- **Ciao-280k**[4]: this dataset is crawled by Tang et al (Tang et al., 2012) from the product review website Ciao (ciao.co.uk). In the above site users can rate and review products as well as add users to their trust network.
- **CiaoDVD**[3]: is crawled by Guo et al (Guo et al., 2014) from the Ciao site from the category of DVDs.
- **Epinions**[4]: is crawled from the Epinions website (epinions.com). As Ciao, Epinions is a consumer review site where users can rate and review products, and add members to their trust list.

The characteristics of the above datasets are reported in Table 3.1. The trust links, in the social network, are directed, while friendship are undirected.

Table 3.1 Description of Datasets

Characteristics	Datasets				
	FilmTrust	CiaoDVD	Ciao-280k	Flixster	Epinions
#Users	1,508	17,615	7,375	147,612	40,163
#Items	2,071	16,121	106,797	48,794	139,738
#Ratings	35,497	72,665	284,052	8,196,077	664,824
Density	1.14%	0.026%	0.04	0.114%	0.01%
ratings-scale	[0.5,4]	[1,5]	[1,5]	[0.5,5]	[1,5]
#links	1,853	22,484	111,781	2,538,746	442,979
links-type	trust	trust	trust	friendship	trust
Network-density	0.08%	0.01%	0.2%	0.011%	0.029%

3.5.2 Evaluation Metrics

Evaluating CF approaches still remains a challenging task. In our experiments we adopt a commonly used approach to evaluate such systems, that consists in assessing the recommendation accuracy on a set of held-out items—the test set. To this end, we retain four widely used measures, from information retrieval, namely the Normalized Discount Cumulative Gain (nDCG), Mean Reciprocal Rank (MRR), Precision@k (Prec@k) and Recall@k (Rec@k), where k is the number of items in the recommendation list.

- **MRR**: The Reciprocal Rank (RR) for a recommendation list is the multiplicative inverse of the rank of the first "good" item. The mean reciprocal rank is the average of the RR's of

[3] http://www.librec.net/datasets.html
[4] http://www.jiliang.xyz/trust.html

all the recommendation lists.

$$MRR = \frac{1}{n}\sum_{i}\frac{1}{rank_i}$$

where n is the number of users who receive recommendations, i.e. the number of recommendation lists, and $rank_i$ is the rank of the first correct item in the recommendation list of user i. Intuitively, the RR measures how far a user should go in the recommendation list to find a good item.

- **nDCG**[5]: the DCG is used to measure the gain of each item relative to its position in a ranked list of items. Formally the DCG for a user i is given by

$$DCG_i = \sum_{j \in D_i} \frac{1}{\log(rank_j + 1)}$$

where D_i denotes the set of held-out items for user i, and $rank_j$ is the rank of item j. The normalized DCG_i is given by

$$nDCG_i = \frac{DCG_i}{idealDCG_i}$$

where the $idealDCG_i$ is the best achievable DCG_i, i.e. the value of the measure if the ranking was perfect. The nDCG is high if the most relevant items appear early in the ranked list. To evaluate an entire model we compute the average nDCG over all users: $nDCG = \frac{1}{n}\sum_i nDCG_i$.

- **Precision@k**: for each user the Precision@k denotes the proportion of good items in his/her top-k recommendation list. To evaluate an entire CF system we compute the average Precision@k over all users.
- **Recall@k** the Recall@k for a user is the proportion of good items, in the user's top-k recommendation list, from the number of relevant held-out items for that user. As for the above measures, we can compute the average Recall@k over all users, to evaluate an entire model.

The nDCG measures the raking quality of a model, while Precision@k and Recall@k assess the quality of a user's top-k recommendation list. All the above measures vary from 0.0 to 1.0, the higher these measures, the better is the recommendation quality.

Notice that we do not consider prediction metrics, such as MAE and RMSE, in our experiments. As it has been already established by previous works (Cremonesi et al., 2010; Amatriain et al., 2012; Loiacono et al., 2014; Chaney et al., 2015), low MAE and RMSE do not necessarily equate to best user satisfaction. As emphasized in (Cremonesi et al., 2010;

[5]Several variants of nDCG exist, here we adopt the same as in LibRec for fairness purpose.

Amatriain et al., 2012), the purpose of a CF system is to provide users with a set of relevant items, as a ranked list. In most commercial systems, users do not receive the predicted rating values, but rather lists of few selected items and ordered according to these values. Thus, the task of item recommendation is by nature a ranking problem. It is therefore more adequate to evaluate CF systems according to the quality of lists of items they recommend. Furthermore, in our case each approach makes predictions in its own range. For instance our method normalizes data so that it lies on a unit hypersphere, some retained competing methods map the original ratings to the interval [0,1]. So, it is not consistent to compare theses approaches in terms of prediction accuracy, by using measures such as MAE and RMSE which are strongly sensitive to the range in which the predicted ratings lie.

3.5.3 Experimental Settings

In our experiments we adopt the 5-fold cross validation strategy. Each dataset was randomly split into five folds. At each run four folds—80% of data—are used for training and the remaining fold is used for testing. On each dataset we perform five runs in order to test all folds. The average performance over the five runs is reported as the final result.

In order for comparisons to be fair and assess the impact of the social network information, we use the same random parameters Θ^0 to initialize both `Soc-movMF` and `movMF`, in all our experiments.

Our approach `Soc-movMF` and the traditional `movMF`, require as an input the number of clusters g which is analogous the number of latent factors k in matrix factorization-based approaches. The number of latent factors k, in the retained baselines, is usually set to 5 or 10 in previous works. In our experiments we found that the different baselines achieve better performances with $k = 5$ in all most all situations, we choose therefore $k = 5$ for all MF-based approaches. Concerning the number of clusters, we empirically found that both our approach and `movMF` provide high performances with a small number of clusters—usually $g \leq 10$. Thus, for fairness purpose, and due to the analogy between g and k, we set $g = 5$ in all our experiments. Nevertheless, we illustrates the impact of g in our experiments.

Another input required by all approaches including the proposed one and competing methods, is the regularization parameter λ. In our case we set this parameter to 1 so as to give equal importance to both the preference and social information. The impact of λ is, however, illustrated in our experiments. The MF-based methods considered here often require several regularization parameters that are usually set to $\lambda = 0.001$. The other settings for the regularization parameters, determined either by our experiments or by previous works (Guo et al., 2015), are as follows: `TrustMF` and `socialMF` $\lambda_T = 1.0$, `SoRec` $\lambda_c = 0.001$ for Flixster and 1.0 for the others, `SoReg` $\beta = 1.0$ for Flixster and 0.1 for the others, `RSTE`

$\alpha = 0.4$, TrustSVD $\lambda_t = 0.9,\ 1.0,\ 1.0,\ 0.5,\ 0.5$ and $\lambda = 1.2,\ 0.5,\ 0.5,\ 0.9,\ 0.8$ for FilmTrust, CiaoDVD, Ciao-280k, Epinions and Flixster, respectively.

3.5.4 Empirical results and discussion

The average performances of each approach over the different datasets are reported in Table 3.2. In order to ease interpretation, Figures 3.3 to 3.6 provide another representation of the results reported in Table 3.2. As these results show clearly, the proposed Soc-movMF performs substantially better than all competing methods, over all datasets. Note that even a small improvement in nDCG may result in an important improvement in terms of the other recommendation quality measures, this is due to the log factor in nDCG.

Beyond the fact that the proposed approach outperforms competing methods, several questions still deserve to be asked.

Is it beneficial to model CF data as directional data distributed on the surface of a unit-hypersphere?

We observe that even if the traditional vMF-based model movMF does not exploit the social interactions among users, it is superior to the other competing methods in almost all situations, except on Flixster where SoReg offer the best performance among the competing methods. This provides strong empirical support for the advantage of leveraging the intrinsic directional properties of CF data.

Table 3.2 Comparison of Average recommendation accuracy over different datasets. "Improve" indicates the improvement reached by the proposed social model Soc-movMF relative to performance of the traditional movMF model.

Dataset	Measure	TrustSVD	TrustMF	SocialMF	RSTE	SoRec	SoReg	movMF	Soc-movMF	Improve
FilmTrust	nDCG	0.1764	0.1679	0.2040	0.4612	0.1833	0.4658	0.6291	**0.6626**	5.05%
	MRR	0.0142	0.0142	0.0799	0.3926	0.0829	0.3664	0.5843	**0.6311**	7.41%
	Prec@10	0.0029	0.0090	0.0117	0.1807	0.0087	0.2069	0.3218	**0.3385**	4.93%
	Rec@10	0.0048	0.0219	0.0242	0.3215	0.0194	0.3391	0.5785	**0.6215**	6.92%
CiaoDVD	nDCG	0.1085	0.1023	0.1149	0.1236	0.1149	0.1283	0.1451	**0.1527**	4.98%
	MRR	0.0062	0.0062	0.0124	0.0214	0.0122	0.0139	0.0252	**0.0335**	24.8%
	Prec@10	0.0012	0.0014	0.0024	0.0059	0.0025	0.0024	0.0059	**0.0077**	23.4%
	Rec@10	0.0059	0.0090	0.0141	0.0384	0.0143	0.0172	0.0351	**0.0455**	22.9%
Ciao-280k	nDCG	0.1206	0.1117	0.1158	0.1094	0.1172	0.1157	0.1690	**0.1820**	7.14%
	MRR	0.0079	0.0009	0.0053	0.0005	0.0068	0.0065	0.0653	**0.0907**	28.0%
	Prec@10	0.0012	0.0001	0.0009	0.0001	0.0012	0.0006	0.0199	**0.0256**	22.3%
	Rec@10	0.0016	0.0002	0.0013	0.0002	0.0019	0.0016	0.0243	**0.0380**	36.1%
Epinions	nDCG	0.1071	0.1004	0.1021	0.0953	0.0944	0.1113	0.1438	**0.1508**	4.64%
	MRR	0.0037	0.0094	0.0045	0.0016	0.0021	0.0047	0.0263	**0.0409**	35.7%
	Prec@10	0.0006	0.0021	0.0009	0.0002	0.0004	0.0005	0.0069	**0.0097**	28.9%
	Rec@10	0.0013	0.0053	0.0022	0.0006	0.0010	0.0017	0.0169	**0.0229**	26.2%
Flixster	nDCG	0.1800	0.1277	0.1399	0.1289	0.1397	0.2768	0.2072	**0.3547**	41.6%
	MRR	0.0376	0.0005	0.0157	0.0004	0.0157	0.1237	0.0497	**0.2382**	79.1%
	Prec@10	0.0137	0.0002	0.0016	0.0001	0.0016	0.0313	0.0113	**0.0948**	88.1%
	Rec@10	0.0116	0.0000	0.0104	0.0000	0.0105	0.0924	0.0045	**0.1635**	97.2%

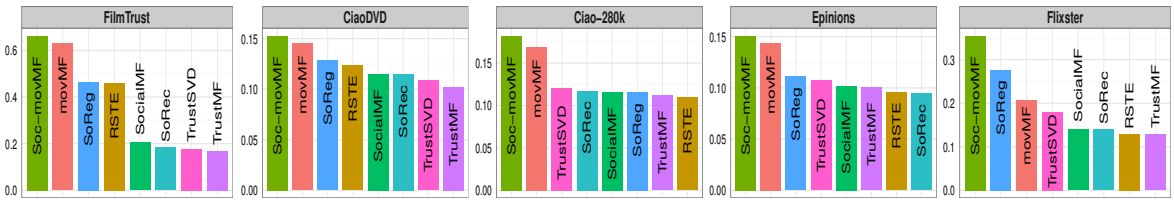


Fig. 3.3 Comparison of average nDCG over different datasets.

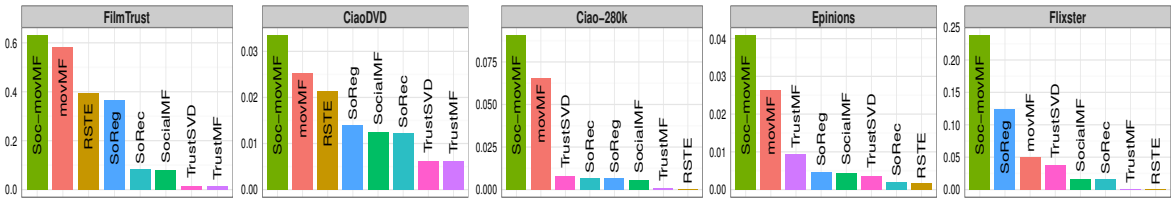


Fig. 3.4 Comparison of average MRR over different datasets.

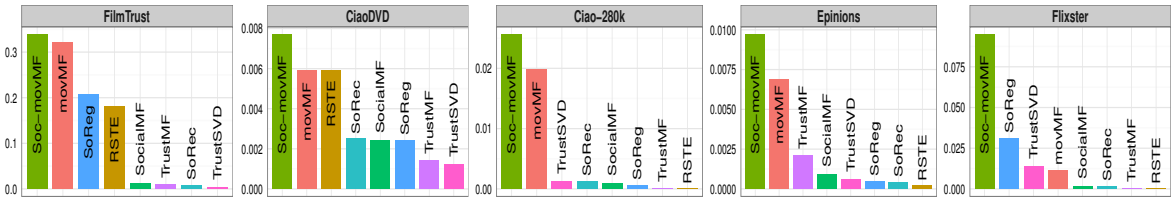


Fig. 3.5 Comparison of average Precision@10 over different datasets.

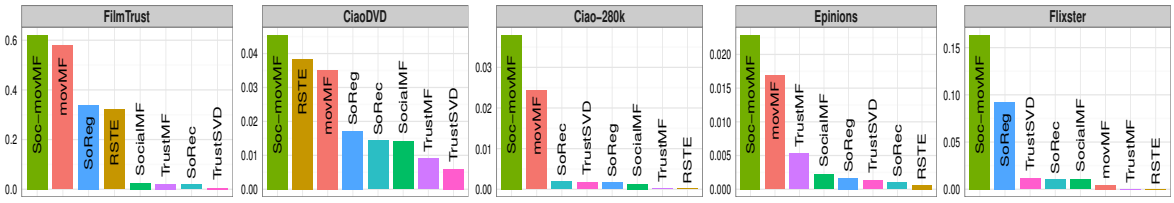


Fig. 3.6 Comparison of average Recall@10 over different datasets.

What is the impact of the social component of `Soc-movMF`*?*

Recall that `movMF` arises as a special case from `Soc-movMF` when $\lambda = 0$, i.e. `movMF` is exactly the `Soc-movMF` without the social component. Column "Improve", in Table 3.2, shows that the proposed `Soc-movMF` noticeably improves the performances of `movMF`. This constitutes empirical evidence that accounting for social interactions among users helps to improve the recommendation quality.

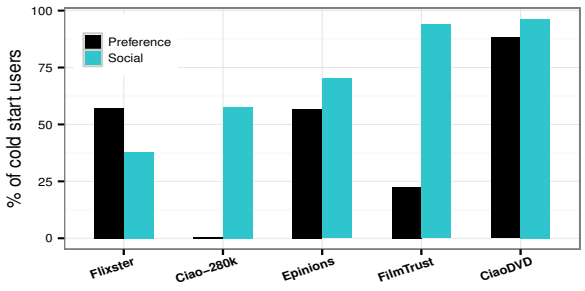


Fig. 3.7 % of cold start users in the user-item matrix (Preference) and social-network (Social).

Why does the improvement rate, in the recommendation quality, differ substantially from one dataset to another?

The improvement rate on Flixster is significantly more important than on the other datasets, while the improvement on FilmTrust is quite low. We believe that this behaviour is due to the characteristics of the different datasets in terms of the number of observed ratings and social relations per user. In Figure 3.7 we report the proportion of cold start users[6], who have very few ratings (resp., social-relations), five or fewer, in the user-item matrix (resp., social network). From Figure 3.7 we note that Flixster exhibits different characteristics in comparison to the other datasets. In fact, in Flixster, while most users—more than 50%—are cold start users in the user-item matrix, only few are cold start in the social network as opposed to the other datasets. This suggests that the social interactions in Flixster play a key role in handling the cold start users in the preference matrix, which allowed the `Soc-movMF` to improve the performances of `movMF` by a noticeable amount. In FilmTrust, we note only few cold start users in the preference matrix, against a lot of cold start users— more than 90%— in the social network. Hence, most information in FilmTrust is contained in the user-item matrix, which may explain the low improvement in the performances of `Soc-movMF` relative to `movMF`. In the Epinions and CiaoDVD datasets we observe a high rate of cold start users in both the user-item matrix and the social network. Although most users expressed only few social interactions in the above two, the social information seems to play an important role, in that it allowed `Soc-movMF` to reach better performances than `movMF`. We believe that this behaviour may be due to the propagation of social interactions in `Soc-movMF`. Finally, the high performances of `Soc-movMF`, relative to its traditional variant `movMF`, on Ciao-280k suggest that even when only few users are cold start users, in the preference matrix, the social information is still of great interest to improve the recommendations.

[6]Cold start users are those with only few observed rating/social-interactions. Following previous works (Jamali and Ester, 2010; Guo et al., 2015) we assume that a user is cold start in the user-item matrix if he/she expressed five or fewer ratings. Similarly a user is cold start in the social network if he/she has five or fewer social interactions.

To sum up, the results from Table 3.2, Figures 3.3 to 3.6 and Figure 3.7 suggest that not only the social information but also the directional properties of CF data should be taken into account to improve the recommendations. It seems better to model collaborative filtering data as directional data. Moreover, accounting for social interactions among users seems to be of particular interest when most users expressed very few ratings, i.e. the cold start users. To investigate the latter result further, in the next section we conduct experiments in which we benchmark the social `Soc-movMF` against the traditional `movMF` on cold start users.

3.5.5 movMFs vs Social-movMFs on cold start users

Cold start is a major challenge in CF, because in many real-world applications most users express very few ratings. In order to complete the results from the previous section, we shall investigate in greater depth the impact of the social information on cold start users, who expressed very few ratings. We conduct, therefore, another series of experiments in which we benchmark our social model `Soc-movMF` against the traditional `movMF` model, on cold start users. From Table 3.3 we can clearly observe that `Soc-movMF` still provides a high recommendation accuracy and substantially improves the performances of `movMF`. More interestingly, on the FilmTrust, Ciao-280*k* and Flixster data sets we observe a greater improvement, in the performance of `Soc-movMF` relative to `movMF`, compared with Table 3.2 (see column "Improve"). In order to understand why the improvement rate may differ substantially from one dataset to another, we report in Figure 3.8 the distribution of out degree

Table 3.3 Comparison of Average recommendation accuracy on cold start users—with 5 or fewer ratings—over different datasets. "Improve" indicates the improvement reached by the proposed social model `Soc-movMF` relative to the performance of `movMF`.

Datasets	Measures	`movMF`	`Soc-movMF`	Improve
FilmTrust	nDCG	0.4534	**0.4663**	2.75%
	MRR	0.3457	**0.3572**	3.21%
	Prec@10	0.0731	**0.0843**	13.3%
	Rec@10	0.4760	**0.5471**	13.0%
CiaoDVD	nDCG	0.1409	**0.1459**	3.40%
	MRR	0.0271	**0.0317**	14.5%
	Prec@10	0.0058	**0.0062**	6.62%
	Rec@10	0.0487	**0.0519**	6.09%
Ciao-280k	nDCG	0.1203	**0.1344**	10.5%
	MRR	0.0201	**0.0431**	53.3%
	Prec@10	0.0041	**0.0073**	42.9%
	Rec@10	0.0185	**0.0375**	50.6%
Epinions	nDCG	0.1102	**0.1127**	2.17%
	MRR	0.0125	**0.0161**	22.0%
	Prec@10	0.0022	**0.0032**	31.1%
	Rec@10	0.0152	**0.0218**	30.1%
Flixster	nDCG	0.1142	**0.3106**	63.2%
	MRR	0.0047	**0.1848**	97.5%
	Prec@10	0.0003	**0.0394**	99.2%
	Rec@10	0.0023	**0.3209**	99.3%

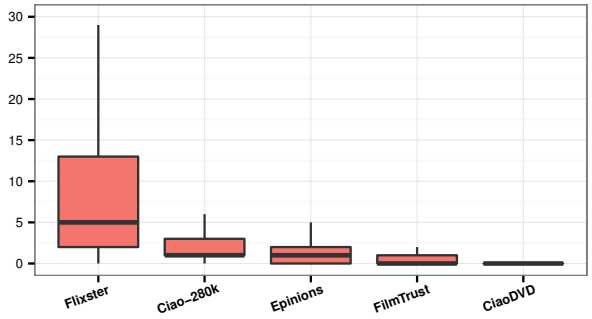


Fig. 3.8 Cold start users : distribution of out degree, i.e., the number of social interactions per cold start user, over the different datasets

per cold start user, i.e. the number of social relations expressed by a cold start user, over the different datasets. From Table 3.3 and Figure 3.8, we observe that the improvement rate in the performances of `Soc-movMF`, relative to `movMF`, goes from high to low as the distribution of the number of social interactions, per cold start user, decreases. For instance on Flixster, we observe that most cold start users expressed more than five social relationships, which may explain the strong superiority of social `Soc-movMF` in comparison to `movMF`. These results, constitute empirical evidence, that accounting for social interactions among users is of great interest and helps to alleviate the cold start issue.

3.5.6 Impact of the number of clusters and the regularization

In the sequel we investigate the impact of the two parameters g and λ on the performances of `Soc-movMF`. We illustrate their behavior on the FilmTrust, CiaoDVD, Ciao-280k and Epinions datasets in terms of nDCG. In Figure 3.9 the values of nDCG are depicted as a function of the number of clusters g, over the different datasets. We observe that a small number of clusters (< 10) seems to be enough, in order for `Soc-movMF` to reach high recommendation performances.

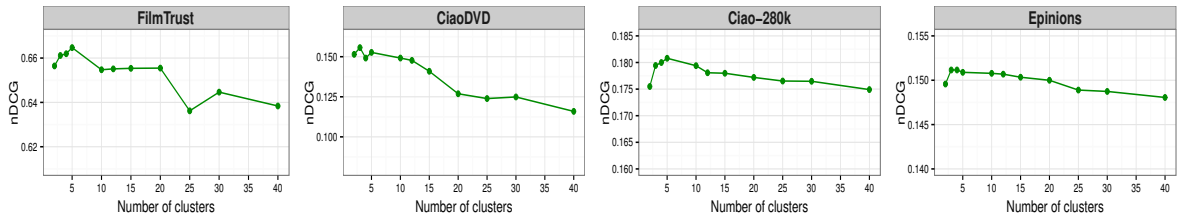


Fig. 3.9 Impact of the number of clusters g.

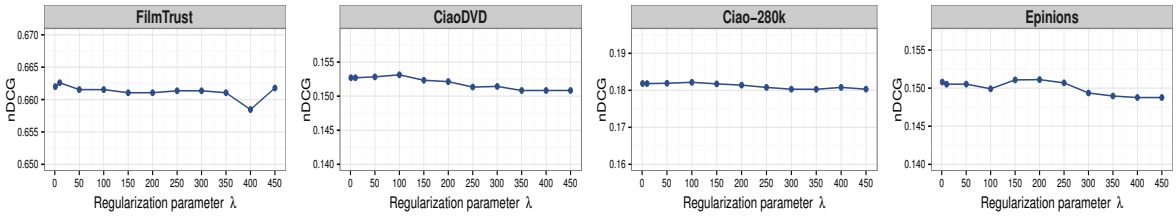


Fig. 3.10 Impact of the regularization parameter λ.

Figure 3.10 illustrates the impact of the regularization parameter λ. As it is clear from this figure, `Soc-movMF` is highly stable relative to the variations of the regularization parameter λ, and seems to provide slightly better performances with a small value of λ, which facilitates the setting of this parameter.

3.6 Conclusion and perspectives

We proposed *Social-movMFs*, a novel model that accounts for social network information to improve item recommendations. *Social-movMFs* simultaneously seeks groups of users who tend to express similar preferences and brings the distributions, over clusters, of socially connected users closer to each other so as to capture the influences between friends. While existing approaches to social CF are based on popular modelling assumptions, such as Gaussian, our approach builds on the vMF distribution which arises naturally for directional data distributed on the surface of a unit-hypersphere. From our experiments, it seems that CF datasets possess intrinsic directional characteristics that are consistent with the vMF modeling assumption. Moreover, incorporating social information into a vMF mixture model turns out to be very beneficial and makes it possible to alleviate the sparsity related issues, such as the cold start problem.

In terms of performance the proposed model improves noticeably the recommendation accuracy of several strong competing methods, including the traditional `movMF` and several social CF models, as illustrated in our experiments. Our empirical results suggest that, for making good recommendations, not only the social interactions among users should be taken into account, but also the intrinsic "directional" properties of CF data.

The good performances of *Social-movMFs* motivates future investigations, that may include incorporating time into *Social-movMFs*, and building online variants so as to handle the frequent changes in social CF: new ratings, social relations, items and users.

Another possible future work is to extend *Social-movMFs* to the context of co-clustering, by relying on the block vMF mixture model, described in Chapter 5, so as to partition the sets of users and items simultaneously. Such an extension would allow us to alleviate the

sparsity problem even better since the co-clustering has proven to be very effective in the context of high dimensional sparse data.

Chapter 4

Co-clustering

This chapter provides a brief overview of existing approaches to co-clustering. The objective here is to give some intuitions about the concept of co-clustering, its advantages over traditional one-sided clustering and an outline of the major approaches to co-clustering.

4.1 Introduction

In general terms co-clustering (Govaert and Nadif, 2013)—also denoted as block clustering (Govaert and Nadif, 2003), bi-clustering (Madeira and Oliveira, 2004), direct clustering (Hartigan, 1972), two-way clustering (Bock, 2003), two-mode clustering (Van Mechelen et al., 2004) or simultaneous clustering (Govaert, 1995)—is an important extension of traditional one-sided clustering, that addresses the problem of simultaneous clustering of both dimensions of a data matrix. More precisely, co-clustering seeks "homogeneous" sub-matrices, namely co-clusters (or blocks)—see Definition 1—where rows and columns follow some consistent patterns.

Definition 1 *Let* $\mathbf{X}$ *be a data matrix of size* $n \times d$ *where* $\mathcal{I}$ *is the set of* n *rows, and* $\mathcal{J}$ *the set of* d *columns. A co-cluster (or a block)* $h\ell$ *is a couple* $(\mathcal{I}_h, \mathcal{J}_\ell)$ *(*$\mathcal{I}_h \subseteq \mathcal{I}, \mathcal{J}_\ell \subseteq \mathcal{J}$*), i.e., each co-cluster corresponds to a sub-matrix* $\mathcal{I}_h \times \mathcal{J}_\ell$*.*

Co-clustering exhibits several practical advantages making it possible to meet the growing needs in several current areas of interest, in terms of effectiveness, scalability and visualization. Below, we summarize some key properties of co-clustering:

- By intertwining row clustering and column clustering at each stage, co-clustering performs an implicitly adaptive dimensionality reduction, which is imperative to deal with high dimensional sparse data. This makes it possible (i) to develop efficient

algorithms with a dramatically smaller number of parameters (ii) to reduce the original data matrix into a much simpler and condensed data matrix with the same structure.

- Co-clustering exploits the inherent duality between rows and columns of data matrices making it possible to enhance the clustering along both dimensions, by exploiting the latent column structure during row assignments and vice versa.

- Far from adding complexity, co-clustering is more informative than one-sided clustering, and produces meaningful clusters. In the case of document-term matrices, for example, co-clustering annotates sets of documents automatically by clusters of words.

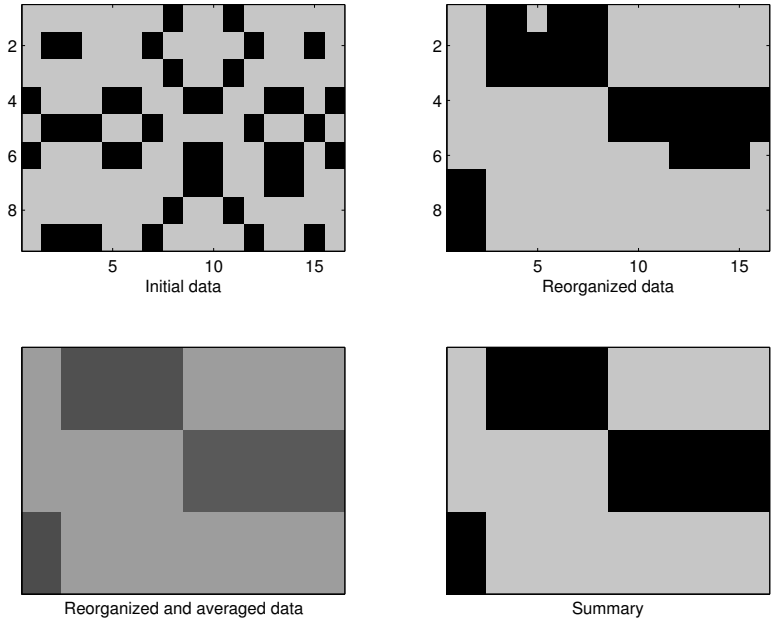


Fig. 4.1 Binary table reorganized according to the row and column partitions

One of the earliest explicit formulation of co-clustering dates back to Hartigan (1972, 1975) under the name of "direct clustering". In the former work, various cluster models and structures were introduced, and a stepwise divisive strategy was proposed that simultaneously seeks hierarchical row and column clustering. The proposed algorithm, initially treats the whole matrix as a block, then at each step a selected block is splited into two sub-blocks, i.e, either by rows or columns, so that the reduction of the within co-cluster variance is maximized. This divisive process is repeated until no significant reduction of the within co-cluster variance is observed. Since this seminal work, co-clustering has received extensive attention in various application domains, such as text mining (Hofmann et al., 1999; Dhillon, 2001; Dhillon et al., 2003; Long et al., 2005; Ding et al., 2006; Ailem et al., 2015) to group

words and documents simultaneously, bioinformatics (Cheng and Church, 2000; Madeira and Oliveira, 2004; Cho et al., 2004; Gupta and Aggarwal, 2010; Hanczar and Nadif, 2012) to cluster genes and experimental conditions simultaneously, web mining (Xu et al., 2010; Charrad et al., 2009), collaborative filtering (Hofmann and Puzicha, 1999; George and Merugu, 2005; Deodhar and Ghosh, 2010; Khoshneshin and Street, 2010) to group users and items simultaneously, affiliation networks (Zanghi et al., 2008, 2010) or evolving graphs (Guigourès et al., 2015), giving rise to a large variety of co-clustering methods.

4.2 Metric-based co-clustering

Metric approaches to co-clustering are based on the intuition that there is an underlying block structure in the original data, that can be exploited to summarize it by a smaller data matrix with the same structure. To this end, metric co-clustering methods consist in optimizing a criterion that measures the difference between the original data matrix and its compressed representation—summary—due to co-clustering. One of the most popular co-clustering criterion is the least-squares, that is suitable for continuous data (Govaert, 1995), given as follows:

$$W(\mathbf{C},\mathbf{z},\mathbf{w}) = \sum_{i,j,h,\ell} z_{ih} w_{j\ell} (x_{ij} - c_{h\ell})^2 \tag{4.1}$$

where $\mathbf{z}$ and $\mathbf{w}$ denote respectively the row and column partitions, $\mathbf{C} = (c_{h\ell})$ is the compressed representation of the original data matrix $\mathbf{X}$ due to co-clustering, i.e, $\mathbf{C}$ is a matrix of size $g \times m$ and $c_{h\ell} \in \mathbb{R}$ is the representative of each block or co-cluster. The co-clustering in this context is to find $\mathbf{z}$, $\mathbf{w}$ and $\mathbf{C}$ minimizing criterion (4.1). The above optimization problem is intractable, however, a locally optimal solution can be obtained, for instance, by using the double k-means procedure—so called because of its relation to the k-means algorithm—described below. Starting from a random initial position, the double k-means algorithm alternates the following optimization steps until convergence:

1. **Row assignment.** Compute $\mathbf{z}$ given $\mathbf{w}$ and $\mathbf{C}$: $z_i = \arg\min_{h'} \sum_{j\ell} w_{j\ell}(x_{ij} - x_{h\ell})^2$, $\forall i$.
2. **Column assignment.** Compute $\mathbf{w}$ given $\mathbf{z}$ and $\mathbf{C}$: $w_j = \arg\min_{\ell'} \sum_{ih} z_{ih}(x_{ij} - x_{h\ell})^2$, $\forall j$.
3. **Update co-clusters.** Compute $\mathbf{C}$ given $\mathbf{z}$ and $\mathbf{w}$: $c_{h\ell} = \frac{\sum_{ij} z_{ih} w_{j\ell} x_{ij}}{z_{.h} w_{.\ell}}$, $\forall h, \ell$.

where $z_{.h}$ and $w_{.\ell}$ denote respectively the cardinality of row cluster h and column cluster ℓ.

A large class of co-clustering algorithms are based on the above least-squares criterion, or its variants, and use the principle of double k-means to solve the corresponding optimization

problem. Among such algorithms we can cite, for instance, the CROEUC algorithm (Govaert, 1995), the minimum sum-squared residue co-clustering algorithms, tailored for gene expression data, proposed in (Cho et al., 2004) and the co-clustering algorithms, based on non-negative matrix tri-factorization, proposed by Labiod and Nadif (2011a).

Another important class of co-clustering methods are those based on information measures such the chi-squared statistic and mutual information. This category of co-clustering algorithms is, in particular, appropriate for dyadic data—contingency tables—such as document-term matrices. In this context we can mention two notable approaches, namely the CROKI2 (Govaert, 1995) and the information theoretic co-clustering (ITCC) (Dhillon et al., 2003) algorithms, based respectively on the chi-squared statistic and mutual information. Both the above approaches aim to minimize the loss in information, relative to the corresponding information measure, due to co-clustering. For more details and theoretical connections between the above two, please refer to the book of Govaert and Nadif (2013).

Motivated by metric approaches to co-clustering, Banerjee et al. (2007) posed the co-clustering as a matrix approximation problem, and they developed a general framework that allows the use of any Bregman divergence to measure the approximation error and, thereby, the quality of co-clustering. For instance, under some constraints and when the the Bregman divergence is the Kullback-Leibler divergence this corresponds to the information theoretic co-clustering (Dhillon et al., 2003), similarly, when the Bregman divergence is the squared euclidean distance this corresponds to the double k-means co-clustering algorithm presented above.

4.3 Graph-based co-clustering

In the graph-based approach data are modeled by a bipartite graph, from graph theory, with two sets of vertices $\mathcal{I}$ and $\mathcal{J}$ denoting respectively rows and columns. Each row/column is represented by a vertex, the two sets of vertices are connected by a set of undirected edges, as illustrated in Figure 4.2.

Let $\mathbf{X}$ denote the original data matrix, the adjacency matrix of the corresponding bipartite graph takes the following form:

$$\mathbf{A} = \begin{pmatrix} 0 & \mathbf{X} \\ \mathbf{X}^T & 0 \end{pmatrix} \tag{4.2}$$

Based on such a graph modelling, the co-clustering is to seek g clusters of densely connected nodes, by minimizing the sum of the weights of the edges between clusters and maximizing the sum of the weights of the edges within clusters.

	B1	B2	B3	B4	B5	B6	B7
A1	1	1	1	0	0	1	1
A2	1	1	0	0	0	1	1
A3	0	0	0	1	1	1	1
A4	0	0	0	1	0	1	1

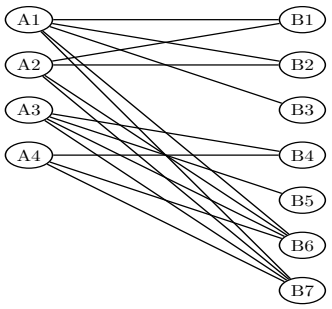


Fig. 4.2 Binary data and its associated bipartite graph

In this context Dhillon (2001) proposed the spectral co-clustering algorithm, he presented a bipartite graph modelling for a document-word collection, then he proposed to find an optimal partitioning of the document-word bipartite graph, into g row and column clusters, by using a spectral relaxation for the minimum cut graph partitioning problem. Labiod and Nadif (2011b) proposed a normalized generalized version of the modularity measure (Newman and Girvan, 2004) which is popular in network analysis. Then, they developed a spectral co-clustering algorithm maximizing the proposed criterion, for simultaneous clustering of binary and categorical data. The authors, shown that the proposed algorithm performs well in the context of text document clustering, and they studied the problem of assessing the number of co-clusters by using the proposed criterion. More recently, Ailem et al. (2015) proposed a novel co-clustering algorithm maximizing the graph modularity. More precisely, they proposed an efficient k-means like alternating optimization scheme as a direct way to maximize the graph modularity, without eigenvector computation—a step which is computationally prohibitive. As in (Labiod and Nadif, 2011b), Ailem et al. (2015) illustrated how the modularity measure can be exploited to assess the number of clusters.

4.4 Model-based co-clustering

The generative mixture model approach (Govaert and Nadif, 2013) is a very powerful technique to design co-clustering algorithms. It offers more flexibility, can incorporate different domain knowledge, allow us to model different types of data and uncover various specific cluster structures. Furthermore, the generative model-based approach provides theoretical justifications behind a large class of metric-based co-clustering algorithms.

In this context, most of co-clustering methods are based on the Latent Block Model (LBM) (Govaert and Nadif, 2003). Intuitively, the LBM assumes that a data matrix is composed of co-clusters, each of which is characterized by an underlying probability distribution called a component. Each co-cluster is a set of real values $x_{ij} \in \mathbb{R}$ drawn from some

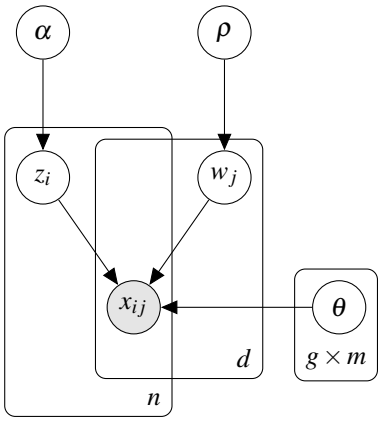


Fig. 4.3 LBM's graphical model

specific *univariate* probability distribution. Putting the co-clustering problem under the LBM framework involves the choice of (i) the number of row and column clusters, (ii) an adequate probability distribution according to the type of data, (iii) a method to fit the model parameters. Formally, the probability density function of the LBM takes the following form:

$$f(\mathbf{X}|\Theta) = \sum_{\mathbf{z},\mathbf{w}} \left(\prod_{i=1}^{n} \alpha_{z_i} \right) \left(\prod_{j=1}^{d} \rho_{w_j} \right) \left(\prod_{i,j} \varphi(x_{ij}|\theta_{z_i w_j}) \right). \tag{4.3}$$

where the notation $\sum_{\mathbf{z},\mathbf{w}}$ stands for the sum over all possible row and column partitions, $\mathbf{X} = (x_{ij})$ denotes a data matrix of size $n \times d$, $z_i \in \{1,\ldots,g\}$ and $w_j \in \{1,\ldots,m\}$ denote the row and column cluster labels, $\Theta = \{\alpha, \rho, \theta_{11}, \ldots, \theta_{gm}\}$ is the set of parameters, such that $\alpha = \{\alpha_1, \ldots, \alpha_g\}$ and $\rho = \{\rho_1, \ldots, \rho_m\}$ represent respectively the row and column mixing proportions, $\theta_{h\ell}$ is the parameter of the probability distribution φ characterizing co-cluster $h\ell$. The LBM assumes that (i) the univariate random variable x_{ij} are independent given $\mathbf{z}$ and $\mathbf{w}$, (ii) the row and column partitions—treated as latent variables—are independent, i.e. $p(\mathbf{z},\mathbf{w}) = p(\mathbf{z})p(\mathbf{w})$. Its graphical model is depicted in Figure 4.3, and the corresponding generative process is as follows:

1. For each row i: choose a cluster label $z_i = h \sim Multinomial(\alpha_1, \ldots, \alpha_g)$.
2. For each column j: choose a cluster label $w_j = \ell \sim Multinomial(\rho_1, \ldots, \rho_m)$.
3. For each entry (i, j) Choose a value $x_{ij} \sim \varphi(x_{ij}|\theta_{z_i w_j})$.

Co-clustering under the LBM is to reverse the above generative process and fit the model parameters Θ to the observed data $\mathbf{X}$. To this end, two main approaches can be considered, namely the Maximum Likelihood (ML) approach—involving the Expectation-Maximization

(EM) algorithm (Dempster et al., 1977)—and the Classification ML (CML) approach (Scott and Symons, 1971; Symons, 1981)—using the Classification EM (CEM) algorithm (Celeux and Govaert, 1992)—, yielding respectively to soft and hard co-clustering. Regarding the type of data several LBM-based co-clustering algorithms have been proposed. We can cite the co-clustering of binary data based on the Bernoulli LBM (Govaert and Nadif, 2003, 2008), the Poisson LBM-based co-clustering (Govaert and Nadif, 2010) dealing with contingency tables. In the above works, the authors considered both the ML and CML approaches to estimate the model parameters. Under the ML approach, the authors emphasized the difficulty due to the coupling of the missing variables $\mathbf{z}$ and $\mathbf{w}$, and derived an efficient variational EM algorithm for inference and parameter estimation.

Under a Bayesian formulation, the latent block model, or its variants, have been also investigated to address specific problems in graph clustering (Latouche et al., 2009; Daudin et al., 2008; Corneli et al., 2015) or to raised challenges such as the model selection or empty classes (van Dijk et al., 2009; Lomet et al., 2014; Keribin et al., 2015). Recently, Wyse and Friel (2012) considered a collapsed Bayesian extension of the Bernoulli LBM (Govaert and Nadif, 2008), which is able to learn the number of row and column clusters. We can also mention the Bayesian co-clustering model (Shan and Banerjee, 2008), which can be viewed as an extension of the latent block model where each row/column may belong to multiple cluster. For more details on the LBM and generative model-based co-clustering approaches the reader can refer to the book of Govaert and Nadif (2013).

One notable advantage of model-based co-clustering approaches is that they provide the theoretical justifications behind a large class of metric-based co-clustering algorithms. For instance, Govaert and Nadif (2013) demonstrated that under some restrictive constraints and when the contingency table is converted to a joint probability distribution, the hard Poisson LBM co-clustering algorithm is equivalent to the well known information theoretic co-clustering (`ITCC`) (Dhillon et al., 2003). In the same spirit, the double k-means co-clustering can be derived as a special case from the Gaussian block model (Nadif and Govaert, 2010; Govaert and Nadif, 2013).

4.5 Non-Negative Matrix Factorization-based approaches

An other variety of co-clustering algorithms are those based on non-negative matrix factorization (NMF). Even if co-clustering is not the purpose of NMF, this approach has recently received a lot of interest for co-clustering positive matrices, such as document-term data arising in text mining. Given a positive data matrix $\mathbf{X}$, NMF-based co-clustering algorithms

seek a three factor decomposition $\mathbf{ZSW}^T$ of $\mathbf{X}$, by optimizing the following criterion

$$\min_{\mathbf{Z}>0,\,\mathbf{S}>0,\,\mathbf{W}>0} \|\mathbf{X} - \mathbf{ZSW}^T\|^2 \tag{4.4}$$

where $\|.\|$ is the Frobenius norm, $\mathbf{Z}$ and $\mathbf{W}$ are two positive matrices containing row and column cluster memberships, respectively, and $\mathbf{S}$ is a positive reduced form—a summary—of $\mathbf{X}$ due to co-clustering. Among NMF based co-clustering methods we can cite the non-negative block value decomposition NBVD by Long et al. (2005), who derived an alternating algorithm based on a set of multiplicative update rules to optimize criterion (4.4). In the same spirit Ding et al. (2006); Yoo and Choi (2010) addressed the problem of simultaneous clustering, by developing two variant of orthogonal three factors NMF, namely ONM3F (Ding et al., 2006) and ONMTF (Yoo and Choi, 2010). The difference with (Long et al., 2005) is that in (Ding et al., 2006; Yoo and Choi, 2010) the authors imposed orthogonality constraints on row and column cluster indicator factors, i.e. $\mathbf{Z}$ and $\mathbf{W}$, and they emphasized the importance of such constraints in the clustering context. We can also cite the work of Labiod and Nadif (2011a) in which two NMF-based co-clustering algorithms have been proposed. Contrary to the previous works, Labiod and Nadif (2011a) placed the co-clustering aim under the NMF formulation at the beginning.

4.6 Conclusion

In this chapter, we introduced the concept of co-clustering, outlined the main approaches to co-clustering and presented some advantages of co-clustering over traditional one-sided clustering when dealing with high dimensional sparse data. Although co-clustering approaches have been proven to be very effective in the context of high dimensionality and sparsity, they still exhibit some limitations. For instance, existing co-clustering methods are based on popular assumptions such as Gaussian, Multinomial, Poisson or Bernoulli, which are inadequate for directional data distributed on the surface of a unit hypersphere (or equivalently L_2 normalized data). However, as it has been emphasized in this thesis and some previous works, some high dimensional sparse data sets, such as text, posses intrinsic directional properties and are, therefore, better modeled as directional data. So far, we highlighted the benefits of both the co-clustering and L_2 normalization in the context of high dimensional sparse data, so it seems natural to question whether it is possible to reap the advantages of both co-clustering and L_2 normalization, simultaneously. In the next chapters, we provide an answer for the above question; we design a general co-clustering model that is well suited for directional data lying on a unit-hypersphere. Our choice for the model-based approach is motivated by

its flexibility and strong theoretical foundations. The proposed model is parsimonious and gives rise to several scalable and effective co-clustering algorithms.

Chapter 5

Von Mises-Fisher based Co-clustering

In the context of high dimensional sparse data, co-clustering turns out to be more beneficial than one-sided clustering even if one is interested in clustering along one dimension, only. As highlighted in the previous chapters and the beginning of this thesis, some datasets, such as document-term matrices, exhibit directional characteristics and the L_2 normalization of such data so that it lies on the surface of a unit hypersphere is necessary.

Popular co-clustering assumptions such as Gaussian or Multinomial are not adequate for directional data. This chapter presents a novel co-clustering model based on a mixture of von Mises-Fisher distributions, which is well-suited for modelling directional data lying on a unit hypersphere. The proposed model successfully integrates a directional measure—cosine similarity—into a co-clustering framework. It is parsimonious and able to reveal a block diagonal structure as well as a good partitioning of rows and columns. By setting the estimate of the model parameters under the maximum likelihood (ML) and the classification ML (CML) approaches, we derive six novel co-clustering algorithms, namely soft, hard, stochastic and simulated annealing variants. Extensive experiments on several simulated and real-world datasets confirm the advantage of our approach and demonstrate the effectiveness of our algorithms.

5.1 Introduction

The mixture of von Mises-Fisher (vMF) distributions (Banerjee et al., 2005b), introduced in chapter 3, turn out to be a wise choice when dealing with high dimensional sparse data. In fact, this model, noted *movMFs*, is one of the most appropriate model for clustering high dimensional sparse data, such as document-term matrices arising in text mining. In this domain, it has been empirically demonstrated that vMF-based clustering methods perform better than several existing approaches, see for instance (Zhong and Ghosh, 2005; Gopal

and Yang, 2014). From a statistical point of view this means that document-term matrices possess directional characteristics (Mardia and Jupp, 2000). Existing vMF-based clustering models, however, focus only on clustering along one dimension, i.e, either row or column clustering. Hence, they do not exploit the inherent duality between rows and columns of data matrices. In the clustering context, it turns out that the exploitation of this duality during the clustering process, presents a real advantage to improve the quality of clustering and alleviate the aforementioned difficulties related to high dimensionality and sparsity. This, can be achieved by using a new form of clustering that simultaneously partitions rows and columns of a data matrix, namely the co-clustering introduced in the previous chapter.

In this chapter, we concentrate on the practical problem of co-clustering document-term matrices arising in text mining. Despite the importance of modelling text data as directional data, existing co-clustering approaches are based on popular modelling assumptions such as Gaussian or Multinomial, which are inadequate to model L_2 normalized data distributed on the surface of a unit hypersphere. Thus, it seems natural to question whether it is possible to get the best of both directional modelling and co-clustering, within the same model. In this chapter, we provide an answer for the above question. We present a general co-clustering framework based on a mixture of von Mises-Fisher distributions. Our model inherits the advantages of both the co-clustering and the modelling assumption of the vMF distribution. Contrary to existing co-clustering methods, it exploits the directional characteristics exhibited by some datasets, such as document-term matrices.

To our knowledge, the work that we present, is the first that addresses the problem of co-clustering with a mixture of vMF distributions. The key contributions of this chapter are the following:

- We present a novel model for co-clustering high dimensional sparse matrices. This model is based the vMF distribution and successfully integrates a directional measure—cosine similarity—into a co-clustering framework. It is therefore more suitable, than existing co-clustering models, for data exhibiting directional characteristics such as document-term matrices.

- We provide theoretical connections between our model and existing ones, namely the mixture of von Mises-Fisher distributions (Banerjee et al., 2005b), the Gaussian (Banfield and Raftery, 1993) and the block Gaussian mixture models (Nadif and Govaert, 2010).

- Setting the estimate of the model parameters under the maximum likelihood (ML) approach, we formulate various co-clustering algorithms a soft, hard, stochastic and two simulated annealing variants. Furthermore, by enforcing some constraints on

the concentration parameters and cluster proportions we derive a novel efficient non-parametric co-clustering algorithm. The latter can be viewed as an extension of the well known spherical k-means algorithm (Dhillon and Modha, 2001) to the context of co-clustering.

- In order to show the benefits of our approach for the analysis of high dimensional sparse data, we conducted extensive experiments on numerous synthetic and real-world datasets, on which we benchmark our algorithms against several strong baselines.

- The dimensionality reduction property of the proposed model alleviates the problem of high concentration parameters κ involved in Bessel functions, which induces over and under flows, a well known difficulty in classical vMF based models (Banerjee et al., 2005b). We validate this important result (i) theoretically by a theorem guaranteeing that our model leads to a concentration parameter that is less or equal to that of *movMFs*, for each cluster, (ii) empirically by demonstrating that our algorithms yield substantially lower concentration parameters than *movMFs*-based algorithms, on real-world datasets.

5.2 Related work

Most of the earlier works using the vMF distribution focused on low dimensional data, i.e, using 2- or 3-dimensional vMF distributions (McLachlan and Peel, 2004), due to difficulties related to the estimation of the concentration parameter κ, that involves the inversion of ratios of Bessel functions. In the context of clustering and high dimensionality, Banerjee et al. (2005b) proposed algorithms derived from a mixture of vMF distributions *movMFs*. They used an EM-based solution to estimate the parameters of their model and proposed an accurate approximation to estimate the concentration parameter κ for a high dimensional vMF distribution. Since this contribution, different vMF based models for clustering high dimensional sparse data have been proposed. For instance Reisinger et al. (2010) proposed a spherical topic model based on a mixture of vMF distributions, which is highly inspired from the Latent Dirichlet Allocation (LDA) (Blei et al., 2003). More recently, for text data clustering, Gopal and Yang (2014) proposed a full Bayesian formulation of *movMFs* and developed two novel variants of *movMFs*, namely hierarchical and temporal. Le and Lauw (2014) proposed a vMF-based model for the semantic visualization of spherical data. In the previous chapter we proposed a vMF model that accounts for the social interactions between users to alleviate the sparsity related issues in CF and improve recommendations. All these works, however, focused only on one-sided clustering.

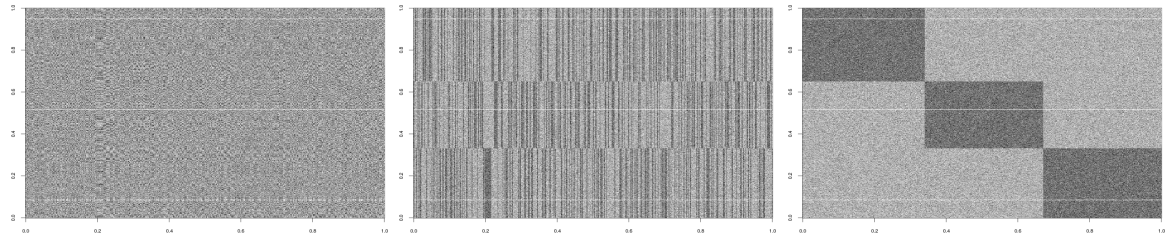


Fig. 5.1 A co-clustering obtained by using $\text{EM}_{\mathbf{b}}$ (soft-dbmovMF): (left) original data, (middle) data reorganized according to $\mathbf{z}$, (right) data reorganized data according $(\mathbf{z}, \mathbf{w})$

Hence, unlike existing vMF-based models, the model proposed in this chapter acts simultaneously on both dimensions of data matrices, and thereby exhibits the advantages of co-clustering, such as the exploitation of the inherent duality between the rows and columns of data matrices, that makes it possible to improve clustering performances. Specifically, our model seeks a diagonal structure by co-clustering, meaning that rows and columns have the same number of clusters, and after a proper reorganisation of rows and columns we obtain a block diagonal structure, as illustrated in Figure 5.1.

In text mining, the domain on which we concentrate our experiments, a diagonal structure arises naturally in document-term matrices, as it has been demonstrated by several earlier works. For example, Dhillon and Modha (2001) proposed the efficient spherical k-means algorithm which can be derived as a special case from a mixture of vMF distributions (Banerjee et al., 2005b), and they empirically demonstrated that documents are grouped together because they use similar words yielding a block diagonal structure. In (Dhillon et al., 2003) the well known information theoretic co-clustering algorithm was proposed, that seeks a more general block structure, i.e, not necessarily diagonal. However during the experiments on real world document-term matrices, the authors observed that some word clusters are highly indicative of individual document clusters inducing a block diagonal sub-structure, that captures the most useful information.

Among popular diagonal co-clustering, we can mention spectral approaches (Dhillon, 2001; Labiod and Nadif, 2011b) and the block diagonal model for co-clustering binary data (Li, 2005) or count data (Ailem et al., 2015).

The diagonal assumption may seem restrictive, however, when dealing with high dimensional sparse data, such as document-term matrices, this assumption exhibits several advantages as opposed to a non-binding model:

- It makes it possible to develop more parsimonious models thereby more efficient algorithms, since we only focus on the most important co-clusters (diagonal ones).

- Unlike a general partionning by co-clustering, that treats similarly useful (relevant) and noisy (irrelevant) co-clusters, a diagonal co-clustering algorithm concentrates on the most relevant co-clusters therefore it is expected to achieve better results, since it implicitly ignores noisy co-clusters.

- Due to sparsity, a general co-clustering algorithm may result in a poor locally optimal solution. More precisely, a general co-clustering algorithm seeks "homogeneous" co-clusters and as all co-clusters are treated equally, the quality of co-clustering may be biased by co-clusters containing a majority of zero entries, that are homogeneous but not useful (or irrelevant). This difficulty increases with the number of co-clusters.

- In the context of document-term matrices, diagonal co-clustering has the advantage of producing directly interpretable document clusters. Let us assume a co-clustering of a document-term matrix into 20 document- and 20 word- clusters. A naive partitioning by co-clustering will produce 400 co-clusters, and it is left to the user to identify the most useful co-clusters so as to determine which document clusters should go with which term clusters, while a diagonal co-clustering will produce only 20 co-clusters that capture the most useful information.

5.3 A Mixture of von Mises-Fisher Distributions for Co-clustering

Recall that the mixture of vMF distributions (*movMFs*) (Banerjee et al., 2005b), presented in chapter 3 (section 3.3.2), focuses solely on clustering along one dimension of a data matrix. Now, we propose a novel mixture of vMF distribution for co-clustering or simultaneous clustering of both dimensions of a data matrix. Following the results of (Dhillon and Modha, 2001), which state that the unit centroids produced by the spherical k-means algorithm—a restricted version of the soft- and hard-*movMF* clustering algorithms (Banerjee et al., 2005b)—are localized in the feature space and tend towards orthonormality, we propose to capture and exploit this structure during the clustering process. More precisely, we assume some natural assumptions on the structure of centroids, i.e, orthonormality and homogeneity, at the beginning. To this end, we introduce a new parameter $\mathbf{w}$ (see Figure 5.2) that simultaneously guarantees the above assumptions and plays the role of a column partition. From a co-clustering point of view, this is equivalent to assume that rows and columns have the same number of clusters and that each column cluster is associated or describes a single row cluster. Thus, inducing a block diagonal structure, see Figure 5.1. The generative process of this model is as follows:

1. Choose a component $h \sim Multinomial(\alpha_1, \ldots, \alpha_g)$.
2. Given μ_h, κ_h and $\mathbf{w}$, generate data point $\mathbf{x}_i$ on $\mathbb{S}^{d-1} \sim f_h(\mathbf{x}_i|\mu_h, \kappa_h, \mathbf{w})$.

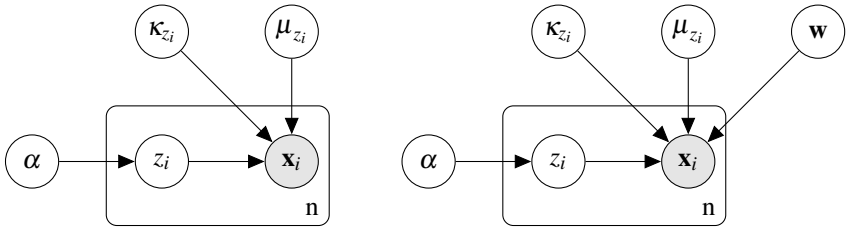


Fig. 5.2 Graphical models: left *movMFs* (Banerjee et al., 2005b), right *dbmovMFs* (proposed model).

Our model called *dbmovMFs* (diagonal block mixture of vMF distributions) seeks to partition simultaneously the set of rows $\mathcal{I}$ and columns $\mathcal{J}$. Thereby it has the advantage of exploiting the duality between rows and columns of data matrices. The density function of *dbmovMFs* is given by:

$$f(\mathbf{x}_i|\Theta) = \sum_h \alpha_h f_h(\mathbf{x}_i|\mu_h^{\mathbf{w}}, \kappa_h^{\mathbf{w}}, \mathbf{w}), \tag{5.1}$$

where the set of parameters Θ is now formed by $\mu_1^{\mathbf{w}}$,..., $\mu_g^{\mathbf{w}}$, $\alpha_1, \ldots, \alpha_g$, $\kappa_1^{\mathbf{w}}, \ldots, \kappa_g^{\mathbf{w}}$ and the column partition $\mathbf{w}$, i.e, $w_j = h$ if the j^{th} column belongs to the h^{th} column cluster, that is associated with the h^{th} row cluster. Notice that, the centroid and the concentration parameters $\mu_h^{\mathbf{w}}$, $\kappa_h^{\mathbf{w}}$, respectively, depend on the column partition $\mathbf{w}$. It follows from the orthonormality and homogeneity assumptions, respectively, that for each centroid $\mu_h^{\mathbf{w}}$: $\mu_{hj} = 0$ if $w_{jh} = 0$, and $\mu_{hj} = \mu_{hh}$ for all j such as $w_{jh} = 1$. For instance, assuming a mixture of 3 vMF distributions, i.e, $g = 3$ and $h, k = 1, 2, 3$, each centroid $\mu_h^{\mathbf{w}} \in \mathbb{S}^{d-1}$ takes this form:

$$\mu_h^{\mathbf{w}} = (\mu_{h1}, \ldots, \mu_{h1}, \mu_{h2}, \ldots, \mu_{h2}, \mu_{h3}, \ldots, \mu_{h3})^\top \tag{5.2}$$

where μ_{hk} is repeated $w_{.h}$ times; $w_{.h}$ denotes the cardinality of the h^{th} column cluster. In addition, we have $\mu_{hk} = 0 \; \forall k \neq h$, leading implicitly to the orthonormality of centroids.

Let $\mathcal{X}$ denotes a set of n randomly sampled data points $\mathbf{x}_i$ on $\mathbb{S}^{d-1}$ according to (5.1). Using the row and column classification matrices $\mathbf{Z}$ and $\mathbf{W}$, respectively, the complete data likelihood of $\mathcal{X}$ takes the following form:

$$\mathcal{L}(\mathbf{W}, \mu, \alpha, \kappa|\mathcal{X}, \mathbf{Z}) = \prod_i \prod_h \left(\alpha_h c_d(\kappa_h) \times \prod_j (\exp^{\kappa_h \mu_{hh} x_{ij}})^{w_{jh}} \right)^{z_{ih}}.$$

The corresponding complete data log-likelihood of $\mathcal{X}$ is given by:

$$\begin{aligned} L_c(\Theta|\mathcal{X},\mathbf{Z}) &= \sum_{i,h} z_{ih}\log\alpha_h + \sum_{i,h} z_{ih}\log(c_d(\kappa_h)) + \sum_{i,h,j} z_{ih}w_{jh}\kappa_h\mu_{hh}x_{ij} \\ &= \sum_{h} z_{.h}\log\alpha_h + \sum_{h} z_{.h}\log(c_d(\kappa_h)) + \sum_{i,h} z_{ih}\kappa_h\mu_{hh}\sum_{j} w_{jh}x_{ij} \\ &= \sum_{h} z_{.h}\log\alpha_h + \sum_{h} z_{.h}\log(c_d(\kappa_h)) + \sum_{i,h} z_{ih}\kappa_h\mu_{hh}u_{ih} \end{aligned} \tag{5.3}$$

where $u_{ih} = \sum_j w_{jh}x_{ij}$. This leads to

$$L_c(\Theta|\mathcal{X},\mathbf{Z}) = \sum_{h} z_{.h}\log\alpha_h + \sum_{h} z_{.h}\log(c_d(\kappa_h)) + \sum_{i,h} z_{ih}y_{ih} \tag{5.4}$$

where $y_{ih} = \kappa_h\mu_{hh}u_{ih}$, and in the same manner, we can give another expression of $L_c(\Theta|\mathcal{X},\mathbf{Z})$ in terms of column assignments as follows

$$\sum_{h} z_{.h}\log\alpha_h + \sum_{h} z_{.h}\log(c_d(\kappa_h)) + \sum_{j,h} w_{jh}t_{jh} \tag{5.5}$$

where $t_{jh} = \kappa_h\mu_{hh}v_{hj}$, with $v_{hj} = \sum_i z_{ih}x_{ij}$.

5.3.1 Connection to existing models

Assuming that the column partition $\mathbf{w}$ is fixed, the *dbmovMFs* model can be viewed as a classical *movMFs* (Banerjee et al., 2005b) where the mean directions vectors μ_h, $h = 1,\ldots,g$, take the form (5.2) described above.

Moreover, we can establish connections with the Gaussian mixture model (Banfield and Raftery, 1993) and the block Gaussian mixture model (Govaert and Nadif, 2013). More precisely, using the equivalence between the vMF and the Gaussian distribution (Mardia and Jupp, 2000) and assuming that $\mathbf{w}$ is fixed, it can be shown that *dbmovMFs* is equivalent to a mixture of Gaussian distributions of spherical form, i.e, the variance of the h^{th} cluster is given by $\sigma_h^2 = \|\mathbf{m}_h^{\mathbf{w}}\|/\kappa_h$, $\mathbf{m}_h^{\mathbf{w}}$ is the centroid of the corresponding Gaussian component and $\mu_h^{\mathbf{w}} = \mathbf{m}_h^{\mathbf{w}}/\|\mathbf{m}_h^{\mathbf{w}}\|$. The latter model, is also equivalent to a diagonal version of the block Gaussian mixture model (Govaert and Nadif, 2013), i.e, each Gaussian component is parameterized by the variance σ_h^2 and mean vector $\mathbf{m}_h^{\mathbf{w}}$, described above.

5.3.2 Maximum Likelihood estimates and the EM$_b$ algorithm

To obtain the maximum likelihood estimates for the parameters Θ, we use the *Generalized* EM algorithm (Dempster et al., 1977; McLachlan and Krishnan, 2007). The E-step is reduced to compute the posterior probabilities defined by $\tilde{z}_{ih} \propto \alpha_h^{(t)} f_h(\mathbf{x}_i|\mu_h^{(t)}, \kappa_h^{(t)})$. The M-step consists in estimating all parameters maximizing or increasing the expectation of the complete data log-likelihood (5.3), subject to the constraints $\sum_h \alpha_h = 1$, $\|\mu_h^{\mathbf{w}}\|^2 = \sum_j w_{jh}\mu_{hh}^2 = 1$ and $\kappa_h > 0$. We obtain the following update formulas (see Appendix A.1).

$$\hat{w}_{jh} \leftarrow \begin{cases} 1, & \text{if } h = \arg\max_{h'} \tilde{t}_{jh'} \\ \\ 0, & \text{otherwise.} \end{cases} \tag{5.6a}$$

$$\hat{\alpha}_h = \frac{\sum_i \tilde{z}_{ih}}{n}, \tag{5.6b}$$

$$\hat{\mu}_{hh} = \frac{r_h^{\mathbf{w}}}{\|\mathbf{r}_h^{\mathbf{w}}\|} = \pm\frac{1}{\sqrt{\hat{w}_{.h}}} \quad \text{where} \quad r_h^{\mathbf{w}} = \sum_{i,j} \tilde{z}_{ih}\hat{w}_{jh}x_{ij} \text{ and } \hat{w}_{.h} = \sum_j \hat{w}_{jh} \tag{5.6c}$$

$$\hat{\kappa}_h \approx \frac{\bar{r}_h^{\mathbf{w}} d - \left(\bar{r}_h^{\mathbf{w}}\right)^3}{1 - \left(\bar{r}_h^{\mathbf{w}}\right)^2} \quad \text{where} \quad \bar{r}_h^{\mathbf{w}} = \frac{I_{d/2}(\hat{\kappa}_h)}{I_{d/2-1}(\hat{\kappa}_h)} = \frac{\|\mathbf{r}_h^{\mathbf{w}}\|}{\tilde{z}_{.h}\hat{w}_{.h}} \tag{5.6d}$$

where $\tilde{t}_{jh} = \kappa_h\mu_{hh}\tilde{v}_{hj}$, with $\tilde{v}_{hj} = \sum_i \tilde{z}_{ih}x_{ij}$, $\mathbf{r}_h^{\mathbf{w}}$ is a d dimensional vector such that $r_{hj}^{\mathbf{w}} = r_h^{\mathbf{w}}$ if $w_{jh} = 1$ and $r_{hj}^{\mathbf{w}} = 0$, otherwise. Alternating the above E and M steps leads to our soft-dbmovMF algorithm described in Algorithm 4.

Observe that, unlike the classical *movMFs* where it is easy to verify that $\bar{r}_h \leq 1$ (see equation 3.13c) given the definition of $\mathbf{r}$, it is not straightforward to verify that $\bar{r}_h^{\mathbf{w}} \leq 1$, without careful analysis. Such a result is imperative, to guarantee that the concentration parameters are positive, i.e, $\kappa_h > 0$, $\forall h$, especially when using the approximation of equation (5.6d). Proposition 1 provides theoretical guarantee about the fact that $0 \leq \bar{r}_h^{\mathbf{w}} \leq 1$, thereby it ensures that concentration parameter κ_h estimated from equation (5.6d) is always positive.

Proposition 1 *Let* $\mathbf{r}$ *be a non-zero vector in* $\mathbb{R}^d$ *(i.e.,* $\mathbf{r} = (r_1, \ldots, r_d)^T$*, such as* $d \geq 1$*) which results from a weighted sum of* n *d-dimensional unit vector, i.e,* $\mathbf{r} = \sum_i p_i\mathbf{x}_i$*,* $\mathbf{x}_i \in \mathbb{R}^d$ *and* $\|\mathbf{x}_i\| = 1$, $\forall i \in \{1, \ldots, n\}$, $n \geq 2$*, the weights* $p_i \geq 0$*,* $\forall i$*. Let* $\mathbf{r}^d$ *be a vector in* $\mathbb{R}^d$*, such as all its components are equal to the sum of elements of* $\mathbf{r}$ *(i.e,* $\mathbf{r}_1^d = \cdots = \mathbf{r}_d^d = \sum_{j=1}^d r_j$*). Then* $0 \leq \|\mathbf{r}^d\| \leq d \times \sum_i p_i$ *with equality only if all unit vectors* $\mathbf{x}_i$ *are equal/collinear.*

The proof is available in Appendix (A.2). By replacing $\mathbf{r}^d$, d and p_i in Proposition 1 with $\mathbf{r}_h^{\mathbf{w}}$, $\hat{w}_{.h}$ and $\tilde{z}_{ih}$ respectively, it is easy to verify $0 \leq \bar{r}_h^{\mathbf{w}} \leq 1$.

Algorithm 4: soft-dbmovMF ($\text{EM}_{\mathbf{b}}$).

Input: $\mathcal{X}$ ($\mathbf{x}_i \in \mathbb{S}^{d-1}$), g the number of co-clusters.
Output: $\tilde{\mathbf{Z}}$ and $\mathbf{W}$,
Steps:
Initialization: $\Theta \leftarrow \Theta^{(0)}$;
repeat
1. Expectation step of EM:
for $i = 1$ **to** n **do**
for $h = 1$ **to** g **do**
$\tilde{z}_{ih} \leftarrow \frac{\alpha_h f_h(\mathbf{x}_i|\mu_h,\kappa_h)}{\sum_l \alpha_l f_l(\mathbf{x}_i|\mu_l,\kappa_l)}$
end for
end for
2. Maximization step of EM:
for $j = 1$ **to** d **do**
for $h = 1$ **to** g **do**
$\tilde{v}_{hj} \leftarrow \sum_i \tilde{z}_{ih} x_{ij}$; $\tilde{t}_{jh} \leftarrow \kappa_h \mu_{hh} \tilde{v}_{hj}$
end for
for $h = 1$ **to** g **do**
$\hat{w}_{jh} \leftarrow \begin{cases} 1, & \text{if } h = \arg\max_{h'} \tilde{t}_{jh'} \\ 0, & \text{otherwise.} \end{cases}$
end for
end for
for $h = 1$ **to** g **do**
$\hat{\alpha}_h \leftarrow \frac{\sum_i \tilde{z}_{ih}}{n}$
$\hat{\mu}_{hh} \leftarrow \pm\frac{1}{\sqrt{\hat{w}_{.h}}}$; $r_h^{\mathbf{w}} \leftarrow \sum_{i,j} \tilde{z}_{ih} \hat{w}_{jh} x_{ij}$
$\hat{\kappa}_h \leftarrow \frac{\bar{r}_h^{\mathbf{w}} d - (\bar{r}_h^{\mathbf{w}})^3}{1-(\bar{r}_h^{\mathbf{w}})^2}$; $\bar{r}_h^{\mathbf{w}} \leftarrow \frac{\|\mathbf{r}_h^{\mathbf{w}}\|}{\tilde{z}_{.h}\hat{w}_{.h}}$
end for
until convergence

More interestingly, notice that the concentration parameters depend on the column partition $\mathbf{w}$, hence, the dimensionality reduction of *dbmovMFs* alleviates the problem of high concentration parameters. The following theorem guarantee that *dbmovMFs* leads to a concentration parameter that is less or equal to that of *movMFs*, for each cluster. The proof is provided in Appendix (A.2). In practice and when dealing with high dimensional datasets, we empirically observe that *dbmovMFs*-based co-clustering algorithms yield substantially lower concentration parameters than *movMFs*-based algorithms, see section (5.7.2).

Theorem 1 *Let* $\mathbf{X}$ *be a* $n \times d$ *matrix, its* i^{th} *row (object)* $\mathbf{x}_i$ *is a* d*-dimensional unit vector in* $\mathbb{S}^{d-1}$ *(i.e,* $\mathbf{x}_i \in \mathbb{R}^d$ *and* $\|\mathbf{x}_i\| = 1$, $\forall i \in \{1,\ldots,n\}$, $n \geq 2$, $d \geq 3$*). Let* $\mathbf{z} = (z_1,\ldots,z_n)$ *denote a partition of the set of objects of* $\mathbf{X}$ *into* g *disjoint clusters. Then, whatever the partition* $\mathbf{w}$ *of attributes of* $\mathbf{X}$ *into* g *disjoints clusters, the concentration parameter of each dbvMF component estimated via approximation (5.6d) is always less or equal to the concentration parameter of the corresponding vMF component estimated via approximation (3.13c). That is,*

$$\hat{\kappa}_h^{\mathbf{w}} \approx \frac{\bar{r}_h^{\mathbf{w}} d - \left(\bar{r}_h^{\mathbf{w}}\right)^3}{1 - \left(\bar{r}_h^{\mathbf{w}}\right)^2} \;\leq\; \hat{\kappa}_h \approx \frac{\bar{r}_h d - \left(\bar{r}_h\right)^3}{1 - \left(\bar{r}_h\right)^2}$$

with equality only if $\bar{r}_h^{\mathbf{w}} = \bar{r}_h$.

5.3.3 Classification Maximum Likelihood estimates and the CEM$_b$ algorithm

Setting our model *dbmovMFs* under the CML approach, that consists in maximizing the classification likelihood instead of its expectation (Scott and Symons, 1971; Symons, 1981; Celeux and Govaert, 1992), we derive a hard version of *dbmovMFs* called CEM$_{\mathbf{b}}$. This is done by incorporating a classification step (C-step) between the E and M steps as follows,

$$z_{ih} \leftarrow \begin{cases} 1, & \text{if } h = \arg\max_{h'} \tilde{z}_{ih'} \\ \\ 0, & \text{otherwise.} \end{cases}$$

The C-step of CEM$_{\mathbf{b}}$, generates a completed sample ($\mathbf{x}_i$,z_i) by allocating each object $\mathbf{x}_i$ to cluster z_i with the highest posterior probability $\tilde{z}_{ih}$, $\forall h$. Then, unlike the EM$_{\mathbf{b}}$ algorithm where the M-step is based on the expectation of the complete data likelihood, the M-step of CEM$_{\mathbf{b}}$ consists in maximizing the complete data likelihood instead of its expectation, thereby the update of the parameters Θ^t are based on the completed sample ($\mathbf{x}_i$,z_i). The corresponding M-step can be deduced from the M-Step of EM$_{\mathbf{b}}$ by replacing $\tilde{z}_{ih}$ by z_{ih} and thereby $\tilde{t}_{jh}$ by t_{jh}. Algorithm 5 gives a full description of CEM$_{\mathbf{b}}$.

Regarding the clustering context, the main difference between ML and CML approaches is that, under the ML approach, the partition $\mathbf{z}$ of the set of objects into g clusters is deduced at convergence of EM$_{\mathbf{b}}$, by assigning each object $\mathbf{x}_i$ to the cluster that maximizes the *a posteriori* probability $\tilde{z}_{ih}$, while under the CML approach the clustering process is taken into account during parameters estimation. In this way, CEM$_{\mathbf{b}}$ simultaneously estimates the parameters and the partition $\mathbf{z}$. It is well known that the ML approach yields more consistent estimate of the parameters thus often provides better clustering than the classification approach,

Algorithm 5: hard-dbmovMF ($\text{CEM}_{\mathbf{b}}$).

Input: $\mathcal{X}$ ($\mathbf{x}_i \in \mathbb{S}^{d-1}$), g the number of co-clusters.
Output: $\mathbf{Z}$ and $\mathbf{W}$,
Steps:
Initialization: $\Theta \leftarrow \Theta^{(0)}$;
repeat
  2. Expectation step of CEM:
  for $i = 1$ **to** n **do**
    for $h = 1$ **to** g **do**
      $f_h(\mathbf{x}_i|\Theta^{(t)}) \leftarrow c_d(\kappa_h)\prod_j (e^{\kappa_h \mu_{hh} x_{ij}})^{w_{jh}}$
    end for
    3. Classification step of CEM:
    for $h = 1$ **to** g **do**
      $z_{ih} \leftarrow \begin{cases} 1, & \text{if } h = \arg\max_{h'} \alpha_{h'} f_{h'}(\mathbf{x}_i|\Theta^{(t)}) \\ 0, & \text{otherwise.} \end{cases}$
    end for
  end for
  4. Maximization step of CEM:
  for $j = 1$ **to** d **do**
    for $h = 1$ **to** g **do**
      $v_{hj} \leftarrow \sum_i z_{ih} x_{ij}$; $t_{jh} \leftarrow \kappa_h \mu_{hh} v_{hj}$
    end for
    for $h = 1$ **to** g **do**
      $\hat{w}_{jh} \leftarrow \begin{cases} 1, & \text{if } h = \arg\max_{h'} t_{jh'} \\ 0, & \text{otherwise.} \end{cases}$
    end for
  end for
  for $h = 1$ **to** g **do**
    $\alpha_h \leftarrow \frac{z_{.h}}{n}$
    $\hat{\mu}_{hh} \leftarrow \pm\frac{1}{\sqrt{\hat{w}_{.h}}}$; $r_h^{\mathbf{w}} \leftarrow \sum_{i,j} z_{ih}\hat{w}_{jh} x_{ij}$
    $\hat{\kappa}_h \leftarrow \frac{\bar{r}_h^{\mathbf{w}} d - (\bar{r}_h^{\mathbf{w}})^3}{1-(\bar{r}_h^{\mathbf{w}})^2}$; $\bar{r}_h^{\mathbf{w}} \leftarrow \frac{\|\mathbf{r}_h^{\mathbf{w}}\|}{z_{.h}\hat{w}_{.h}}$
  end for
until convergence

especially when the clusters are not well separated. However, the CML approach exhibits some nice properties generated by $\text{CEM}_{\mathbf{b}}$,

- $\text{CEM}_{\mathbf{b}}$ is considerably more faster and scalable than $\text{EM}_{\mathbf{b}}$, for instance, consider the update of the parameter $\hat{w}_{jh}$, under the ML approach (see, equation 5.6a) we iterate through all objects $\mathbf{x}_i$ to compute $\tilde{t}_{jh}$, while with $\text{CEM}_{\mathbf{b}}$ we only iterate through objects within the h^{th} cluster to compute t_{jh}.

- $\text{CEM}_{\mathbf{b}}$ allows us to avoid numerical difficulties, i.e, over and under flows, related to the computation of the conditional probabilities $\tilde{z}_{ih}$, especially in the case of vMF distribution where the normalization terms $c_d(\kappa_h)$ involve Bessel functions and the concentration parameters act as multipliers in the exponent. More precisely, with $\text{CEM}_{\mathbf{b}}$ there is no need for computing the conditional probabilities $\tilde{z}_{ih}$, the C-step can be done equivalently by assigning each object $\mathbf{x}_i$ to the cluster maximising $\log \alpha_h + \log f_h(\mathbf{x}_i|\Theta^{(t)})$. Furthermore, this contributes to the efficiency, scalability and memory-space saving of $\text{CEM}_{\mathbf{b}}$.

- It eases the derivation of a large number of standard clustering algorithms as special cases from a mixture framework, which allows to give them a probabilistic interpretation. For example, in our case, starting from $\text{CEM}_{\mathbf{b}}$ we derive a novel co-clustering algorithm, which can be viewed as an extension of the well known spherical kmeans algorithm (Dhillon and Modha, 2001) to the context of co-clustering.

5.4 Diagonal Block Spherical Kmeans (dbSkmeans)

In this section, we derive a novel diagonal co-clustering algorithm from hard-dbmovMF, by enforcing some restrictive constrains on the concentration parameters κ_h and the mixing proportions α_h. We name this novel algorithm diagonal block spherical kmeans (dbSKmeans), due to its theoretical connection to the spherical k-means algorithm (Dhillon and Modha, 2001).

5.4.1 Definition

Let's assume that all the mixing proportions are equal and all the concentration parameters are equal to a finite constant, i.e, $\alpha_h = \frac{1}{g}$, $\kappa_h = \kappa$, for all $h \in \{1,\ldots,g\}$. Under these restrictions, the complete data log-likelihood (A.2) becomes:

$$\begin{aligned} L_c(\Theta|\mathcal{X},\mathbf{Z}) &= \sum_h z_h \log(1/g) + \sum_h z_h \log(c_d(\kappa)) + \kappa \sum_{i,h,j} z_{ih} w_{jh} \mu_{hh} x_{ij} \\ &= \underbrace{n[\log(1/g) + \log(c_d(\kappa))]}_{constant} + \kappa \sum_{i,h} z_{ih} (\mu_h^{\mathbf{w}})^T \mathbf{x}_i. \end{aligned} \tag{5.7}$$

Hence, maximizing the complete data log-likelihood (5.7) is equivalent to maximizing the following criterion:

$$\sum_{i,h} z_{ih} (\mu_h^{\mathbf{w}})^T \mathbf{x}_i. \tag{5.8}$$

Observe that, criterion (5.8) is given in terms of row clustering, we can similarly express it in terms of column clustering as follows

$$\sum_{j,h} w_{jh} t_{jh} \quad \text{where} \quad t_{jh} = \kappa \mu_{hh} v_{hj} \quad \text{with} \quad v_{hj} = \sum_{i} z_{ih} x_{ij}. \tag{5.9}$$

Intertwining the maximization of criterion (5.8) and (5.9) for fixed row and column clustering, respectively, constitutes our block spherical kmeans (dbSkmeans) algorithm, for co-clustering directional data. The centroid μ_h is estimated in the same way as in the M-step of hard-dbmovMF (i.e, $\hat{\mu}_{hh} = \pm\frac{1}{\sqrt{\hat{w}_{.h}}}$, for all $h \in \{1,\ldots,g\}$). The dbSkmeans algorithm is a restricted version of hard-dbmovMF, hence at each iteration it never decreases criterion (5.8) and it is guaranteed to terminate in a finite number of iterations at a local maximum of the restricted likelihood function (5.8). Algorithm 6 describes dbSkmeans in detail.

Algorithm 6: dbSkmeans.

Input: $\mathcal{X}$ ($\mathbf{x}_i \in \mathbb{S}^{d-1}$), g the number of co-clusters.
Output: **Z** and **W**,
Steps:
Initialization: $\Theta \leftarrow \Theta^{(0)}$;
repeat
1. Rows assignment:
for $i = 1$ **to** n **do**
$z_i \leftarrow h$ where $h = \arg\max_{h'} cos(\mu^{\mathbf{w}}_{h'}, \mathbf{x}_i)$
end for
2. Columns assignment:
for $j = 1$ **to** d **do**
$w_j \leftarrow h$ where $h = \arg\max_{h'} \frac{\sqrt{z_{.h'}}}{\sqrt{w_{.h'}}} cos(\mu^{\mathbf{z}}_{h'}, \mathbf{x}^j)$
end for
3. Centroids update:
for $h = 1$ **to** g **do**
$\mu_{hh} \leftarrow \pm\frac{1}{\sqrt{w_{.h}}}$
end for
until convergence

5.4.2 Connection to other algorithms

Note that, maximizing criterion (5.8) is equivalent to maximizing the sum of *cosine similarity* between each object $\mathbf{x}_i$ and its corresponding centroid, thereby if we further assume that the column partition $\mathbf{w}$ is fixed, criterion (5.8) is exactly the criterion maximized by the spherical k-means algorithm (Dhillon and Modha, 2001), where the centroids take the form (5.2) introduced in the previous section.

Moreover, connections with the two-sided (or double) k-means algorithm, i.e, using euclidean distance, can be established. It is well known that the double kmeans algorithm minimizes the following objective function

$$\begin{aligned} W(\mathbf{z},\mathbf{w}) &= \sum_{i,j,h,\ell} z_{ih} w_{j\ell} (x_{ij} - c_{h\ell})^2 \\ &= \sum_{i,j,h,\ell} z_{ih} w_{j\ell} x_{ij}^2 + \sum_{i,j,h,\ell} z_{ih} w_{j\ell} c_{h\ell}^2 - 2 \sum_{i,j,h,\ell} z_{ih} w_{j\ell} c_{h\ell} x_{ij} \\ &= \sum_{i,h} z_{ih} \|\mathbf{x}_i\|^2 + \sum_{i,h} z_{ih} \|\mathbf{c}_h^{\mathbf{w}}\|^2 - 2 \sum_{i,h} z_{ih} (\mathbf{c}_h^{\mathbf{w}})^\top \mathbf{x}_i \end{aligned} \tag{5.10}$$

where $\mathbf{c}_h^{\mathbf{w}}$ denotes the h^{th} row centroid, i.e, $\mathbf{c}_h^{\mathbf{w}} = (c_{h1}, \ldots, c_{h1}, c_{h2}, \ldots, c_{h2}, \ldots, c_{hm}, \ldots, c_{hm})^\top$ assuming that the data points $\mathbf{x}_i$ and the row centroids $\mathbf{c}_h^{\mathbf{w}}$ are normalized in order to lie on the surface of a unit hypersphere, i.e, $\mathbf{x}_i, \mathbf{c}_h^{\mathbf{w}} \in \mathbb{R}^d$ and $\|\mathbf{x}_i\| = \|\mathbf{c}_h^{\mathbf{w}}\| = 1$, then optimizing criterion $W(\mathbf{z},\mathbf{w})$ (5.10) is equivalent to maximizing the following criterion

$$W(\mathbf{z},\mathbf{w}) = \sum_{i,h} z_{ih} (\mathbf{c}_h^{\mathbf{w}})^\top \mathbf{x}_i. \tag{5.11}$$

Furthermore, if we constrain the number of row clusters to be equal to that of column clusters and the centroids $\mathbf{c}_h^{\mathbf{w}}$ to take the form (5.2) introduced earlier, then criterion (5.11) becomes exactly the same as criterion (5.8) optimized by our dbSkmeans algorithm. Hence, under the aforementioned constrains dbSkmeans is equivalent to a large number of co-clustering algorithms using the principle of double k-means, among which we can cite the well known Minimum Sum-Squared Residue Co-Clustering of Gene Expression Data by Cho et al. (2004), the CROEUC algorithm due to Govaert (1995) and the co-clustering based on non-negative matrix tri-factorization proposed by Labiod and Nadif (2011a).

5.4.3 Analysis of column assignment

Hereafter, we shall give some intuitions and interpretations about the column affectation step of dbSkmeans.

$$\begin{aligned} w_j &= \arg\max_h t_{jh} = \arg\max_h \kappa\mu_{hh} v_{hj} \\ &= \arg\max_h \kappa\mu_{hh} \sum_i z_{ih} x_{ij} = \arg\max_h \kappa(\mu_h^{\mathbf{z}})^T \mathbf{x}^j \\ &= \arg\max_h \kappa \|\mu_h^{\mathbf{z}}\| \|\mathbf{x}^j\| cos(\mu_h^{\mathbf{z}}, \mathbf{x}^j) \equiv \arg\max_h \|\mu_h^{\mathbf{z}}\| cos(\mu_h^{\mathbf{z}}, \mathbf{x}^j) \\ &= \arg\max_h \frac{\sqrt{z_{.h}}}{\sqrt{w_{.h}}} cos(\mu_h^{\mathbf{z}}, \mathbf{x}^j) \end{aligned} \tag{5.12}$$

where $\mathbf{x}^j$ denotes the j^{th} column, $\mu_h^{\mathbf{z}}$ denotes the h^{th} "column centroid", i.e, $\mu_{hi} = 1/\sqrt{w_{.h}}$ if $z_{ih} = 1$ and $\mu_{hi} = 0$, otherwise. The superscript $\mathbf{z}$ is used to denote the fact that the column centroid $\mu_h^{\mathbf{z}}$ depends on the row clustering.

Hence, the column affectation step of dbSkmeans is equivalent to maximizing a "weighted" *cosine similarity* between each column $\mathbf{x}^j$ and the corresponding column centroid $\mu_h^{\mathbf{z}}$. The weight $\frac{\sqrt{z_{.h}}}{\sqrt{w_{.h}}}$ has a nice semantic interpretation, for instance let $\mathbf{x}^j$ be a column such that $cos(\mu_1^{\mathbf{z}}, \mathbf{x}^j) = cos(\mu_2^{\mathbf{z}}, \mathbf{x}^j)$, $w_{.1} = w_{.2}$ and $z_{.1} > z_{.2}$, then the column $\mathbf{x}^j$ will be assigned to the first column cluster. The role of the weight in this situation is to assign more elements to column clusters describing row clusters containing more objects than the others. This is natural, since row clusters with a lot of objects are expected to be described by more features than those with few objects. Furthermore, as the above weight is inversely proportional to the root square of the size of the corresponding column cluster, it has an interesting regularizing effect that makes it possible to avoid the formation of empty and/or very large column clusters. Thereby, this implicitly prevents dbSkmeans from generating empty (very large) row clusters induced by empty (very large) column clusters.

In summary, the weight involved in the column affectation step of dbSkmeans, has a nice regularizing effect, that prevents dbSkmeans from generating highly skewed solution with empty and/or very large co-clusters. Moreover, it makes it possible to find a trade-off between row and column clusters sizes, by trying to deal with the natural intuition that row clusters with more objects are expected to be described by column clusters with more features, than the other row clusters.

5.5 Stochastic Variants

It is well known that the EM algorithm is strongly dependent on its starting position. The stochastic variant of EM (SEM) (Celeux and Diebolt, 1985) however makes it possible to overcome this limitation. It consists in incorporating a stochastic step (S-step) between the E and M steps, which generates a completed sample $(\mathbf{x}_i,z_i)$ by drawing a cluster $z_i \in \{1,\ldots,g\}$ for each data point $\mathbf{x}_i$ according to the multinomial distribution $\mathcal{M}(\tilde{z}_{i1},\ldots,\tilde{z}_{ig})$.

Hence, as with the CEM algorithm (Celeux and Govaert, 1992) the update of the parameters Θ^t are based on the completed sample $(\mathbf{x}_i,z_i)$ i.e, complete data likelihood instead of its expectation. It follows from the above description that SEM does not share the convergence properties of EM and CEM. In fact, SEM can allow an update estimate Θ^{t+1} even if $L(\Theta^{t+1}) < L(\Theta^t)$. Thereby, SEM does not necessarily converge to the first encountered stationary point of the log-likelihood, which allows SEM to ignore saddle points and insignificant local optima. This is the main reason why SEM can be expected to achieve better solutions than EM. It has been shown that the sequence of parameters $\{\Theta^t\}$ generated by SEM is an ergodic Markov chain, thereby, it converges to the unique stationary distribution of this Markov chain. For further details about SEM, the reader can refer to (Celeux and Diebolt, 1985).

5.5.1 Stochastic *dbmovMFs* and the SEM$_\mathbf{b}$ algorithm

Based on SEM we formulate a stochastic version of soft-dbmovMF called SEM$_\mathbf{b}$. In our case, we further propose a stochastic column assignment, by converting $\tilde{t}_{jh}$ to probabilities $\tilde{w}_{jh}$, i.e, $\tilde{w}_{jh} \propto \tilde{t}_{jh}$ and drawing a cluster w_j for each column j according to multinomial distribution $\mathcal{M}(\tilde{w}_{j1},\ldots,\tilde{w}_{jg})$. This makes it possible to avoid a quick convergence to a bad local optimum of the likelihood function, due to hard column assignments. More intuitively, in the case of document-term matrices, during the first iterations document clusters are mixed, hence each word is involved to describe several document clusters thereby imposing hard word assignments at the beginning, by using equation (5.6a), can lead to a poor local optimum of the likelihood function, especially when the clusters are poorly separated. Algorithm 7 describes in more details SEM$_\mathbf{b}$.

5.5.2 Simulated annealing *dbmovMFs*-based algorithms

One of the main drawback of the SEM algorithm is that it does not share the convergence properties of EM; SEM converges in distribution and does not converge point-wise. Furthermore, when the sample size of the observed data is small, SEM may converge to a stationary

Algorithm 7: Stochastic-dbmovMF (SEM$_\mathbf{b}$).

Input: $\mathcal{X}$ ($\mathbf{x}_i \in \mathbb{S}^{d-1}$), g the number of co-clusters.
Output: $\mathbf{Z}$, $\mathbf{W}$ and Θ.
Steps:
Initialization: $\Theta \leftarrow \Theta^{(0)}$;
repeat
1. Expectation step:
for $i = 1$ **to** n **do**
for $h = 1$ **to** g **do**
$\tilde{z}_{ih} \leftarrow \frac{\alpha_h f_h(\mathbf{x}_i|\mu_h,\kappa_h)}{\sum_l \alpha_l f_l(\mathbf{x}_i|\mu_l,\kappa_l)}$
end for
2. Stochastic step:
Select $z_i \in \{1,\ldots,g\} \sim \mathcal{M}(\tilde{z}_{i1},\ldots,\tilde{z}_{ig})$
end for
3. Maximization step:
for $h = 1$ **to** g **do**
$\alpha_h \leftarrow \frac{\sum_i z_{ih}}{n}$
for $j = 1$ **to** d **do**
$v_{hj} \leftarrow \sum_i z_{ih} x_{ij}$; $t_{jh} \leftarrow \kappa_h \mu_{hh} v_{hj}$
end for
end for
for $j = 1$ **to** d **do**
Select $w_j \in \{1,\ldots,g\} \sim \mathcal{M}(\tilde{w}_{j1},\ldots,\tilde{w}_{jg})$
end for
for $h = 1$ **to** g **do**
$\hat{\mu}_{hh} \leftarrow \frac{1}{\sqrt{w_{.h}}}$; $r_h^{\mathbf{w}} \leftarrow \sum_{i,j} z_{ih} w_{jh} x_{ij}$
$\hat{\kappa}_h \leftarrow \frac{\bar{r}_h^{\mathbf{w}} d - (\bar{r}_h^{\mathbf{w}})^3}{1-(\bar{r}_h^{\mathbf{w}})^2}$; $\bar{r}_h^{\mathbf{w}} \leftarrow \frac{\|\mathbf{r}_h^{\mathbf{w}}\|}{z_{.h} w_{.h}}$
end for
until convergence

distribution with a high variance, which leads to poor estimation of the parameters. In order to overcome the aforementioned limitations, the Simulated Annealing version of EM (SAEM) has been proposed (Celeux and Diebolt, 1992), that reaps the benefit of both EM and SEM simultaneously. To do so, let γ_t be a sequence of positive real numbers starting from 1 and

decreasing to zero, at each iteration the parameters are updated as follows

$$\Theta_{SAEM}^{t+1} = (1-\gamma_{t+1})\Theta_{EM}^{t+1} + \gamma_{t+1}\Theta_{SEM}^{t+1}$$

where Θ_{EM}^{t+1} and Θ_{SEM}^{t+1} denote the parameters estimated using EM and SEM, respectively. Thus, SAEM goes from pure SEM at the beginning towards pure EM at the end. If we further impose the following constraint $\lim_{t\to\infty} \frac{\gamma_t}{\gamma_{t+1}} = 1$, we can ensure the asymptotic convergence of SAEM to a local maximum of the log-likelihood function.

Based on SAEM, we derive the SAEM$_{\mathbf{b}}$ algorithm, which is a simulated annealing version of EM$_{\mathbf{b}}$. In our case, however, it is difficult to use the above update formula of SAEM due to parameter $\mathbf{w}$. In order to overcome this difficulty, we propose to use the following simplified form to update the parameters, which turns out to be effective and less costly.

$$\Theta_{SAEM_{\mathbf{b}}}^{t+1} \leftarrow \begin{cases} \Theta_{SEM_{\mathbf{b}}}^{t+1}, & \text{if } \gamma_{t+1} \geq (1-\gamma_{t+1}) \\ \\ \Theta_{EM_{\mathbf{b}}}^{t+1}, & \text{otherwise.} \end{cases}$$

We propose to use the following exponentially decreasing form for $\gamma_t = 1 - \exp^{\frac{it^t - it^{max}}{\beta}}$ where it^t, it^{max} denote respectively, the current iteration and the maximum number of iterations, β is a positive real parameter that controls the number of SEM$_{\mathbf{b}}$ and EM$_{\mathbf{b}}$ iterations to be performed. More precisely when $\beta \to \infty$, SAEM$_{\mathbf{b}}$ tends to pure EM$_{\mathbf{b}}$, similarly when $\beta \to 0$, SAEM$_{\mathbf{b}}$ tend to pure SEM$_{\mathbf{b}}$. In our experiments, we set $\beta = 20$ and $it^{max} = 100$, in such a way that most iterations at the beginning are done using SEM$_{\mathbf{b}}$, in order to reach a steady state and avoid poor local optima, and the last few iterations are performed using EM$_{\mathbf{b}}$, to ensure the convergence of SAEM$_{\mathbf{b}}$ to a local optimum of the log-likelihood function.

This simplified version consists in initializing EM$_{\mathbf{b}}$ with SEM$_{\mathbf{b}}$. Note that, seeing the initialization of EM$_{\mathbf{b}}$ via SEM$_{\mathbf{b}}$ as a kind of a Simulated Annealing variant, justifies why EM$_{\mathbf{b}}$ is expected to provide better performances with such an initialization.

In the same manner, we derived a hard simulated annealing variant, denoted in this chapter as CAEM$_{\mathbf{b}}$. The difference with the SAEM$_{\mathbf{b}}$ algorithm is that the iterations at the end are performed using CEM$_{\mathbf{b}}$ instead of EM$_{\mathbf{b}}$. In Figure 5.3 we report the different algorithms discussed in this chapter.

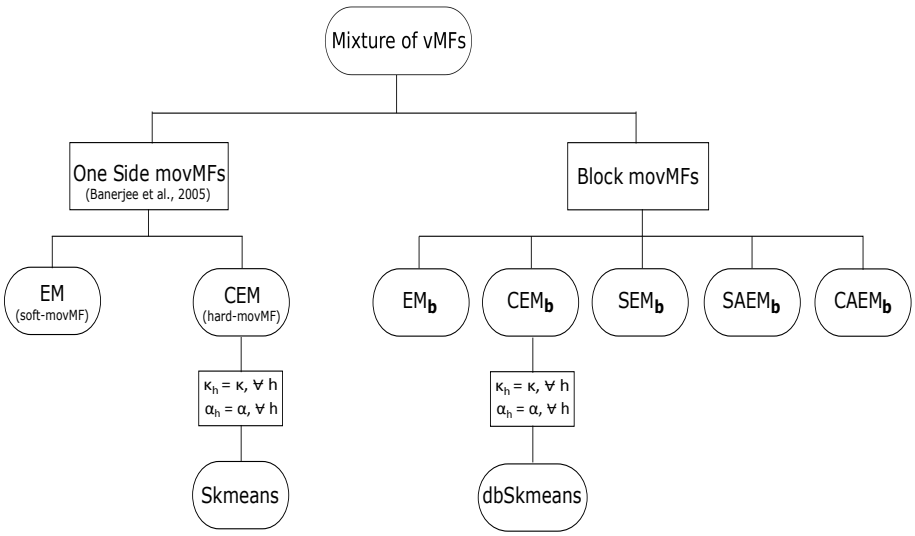


Fig. 5.3 Von Mises-Fisher mixture models-based clustering (left) and co-clustering (right) algorithms.

5.6 Computational Complexity in the Worst Case

In this section, we shall analyse the computational complexity of dbSkmeans, hard-dbmovMF and soft-dbmovMF.

Proposition 2 *Let* $\mathbf{X}$ *be a* $n \times d$ *matrix, let nz denote the number of non-zeros entries in* $\mathbf{X}$*, "it" is the number of iterations and g is the number of co-clusters to be found. Then (i) the computational complexity of dbSkmeans described in Algorithm 6 is given in* $O(it \cdot nz)$*, (ii) the computational complexity of hard-dbmovMF described in Algorithm 5 is given in* $O(it \cdot nz)$ *(iii) the computational complexity of soft-dbmovMF described in Algorithm 4 is given in* $O(it \cdot g \cdot nz)$*.*

Proof (i). The computational bottleneck for *dbSkmeans* is with the row and column assignments. We prove that the complexity of both row and column assignments is $O(nz)$. Let $\mathbf{x}_i$ denote the ith row of $\mathbf{X}$ (i.e, $\mathbf{x}_i$ is a d dimensional vector in $\mathbb{S}^{d-1}$). Assume that we look for a co-clustering of $\mathbf{X}$ into g co-clusters, and let $\mu_h^{\mathbf{w}}$ be the hth centroid, characterising the hth co-cluster. The computational cost of the scalar product $(\mu_h^{\mathbf{w}})^T \mathbf{x}_i$ is $O(x_i^h)$, where x_i^h is number of non-zeros entries of $\mathbf{x}_i$ within the hth column cluster, this complexity holds thanks to the form of $\mu_h^{\mathbf{w}}$, see equation (5.2). Thereby, the complexity of the assignment of $\mathbf{x}_i$ is given in $O(x_i^*)$ (based on $O(x_i^1 + \cdots + x_i^g)$), where x_i^* denotes the number of non-zeros entries of $\mathbf{x}_i$. Therefore, the total cost of one row assignments step is $O(nz)$. Similarly, we can show that the cost of one column assignments step is $O(nz)$. Thus the computational complexity of *dbSkmeans* is given in $O(it \cdot nz)$, based on $O(it \cdot (nz + nz))$.

Table 5.1 Computational complexity of different algorithms (in the worst case). *nz* denotes the number of non-zero entries in the matrix $\mathbf{X}$, g is the number of clusters (co-cluster) and *it* denotes the number of iterations.

movMFs (Banerjee et al., 2005b)			***dbmovMFs*** **(this chapter)**		
soft-movMF	hard-movMF	Skmeans	soft-dbmovMF	hard-dbmovMF	dbSkmeans
$O(it \cdot g \cdot nz)$	$O(it \cdot g \cdot nz)$	$O(it \cdot g \cdot nz)$	$O(it \cdot g \cdot nz)$	$O(it \cdot nz)$	$O(it \cdot nz)$

In the same manner we can prove (ii) and (iii), see Appendix (A.3) for further details. Proposition 2, states that theoretically, the computational time of our algorithms is linear with respect to the number of non-zero entries in $\mathbf{X}$. Hence, the proposed algorithms are very efficient (in particular the hard variants) and therefore suitable for large sparse datasets. Table 5.1 summarizes the computational complexity of the different algorithms.

5.7 Experimental results

In this section, we shall provide extensive empirical results to validate and illustrate the benefits of our model *dbmovMFs* and the corresponding co-clustering algorithms. We first propose to validate the correctness of our model on simulated datasets. Then, in order to show the benefits of our approach, we conduct extensive experiments on numerous real world text datasets, in which we benchmark our algorithms, i.e, $\text{EM}_{\mathbf{b}}$, $\text{CEM}_{\mathbf{b}}$, $\text{SEM}_{\mathbf{b}}$, $\text{SAEM}_{\mathbf{b}}$, $\text{CAEM}_{\mathbf{b}}$ and dbSkmeans against several strong baselines denoted in this chapter as follows

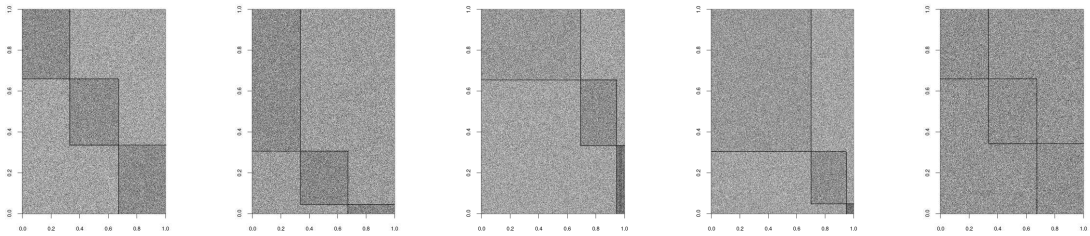
- EM denotes the soft-movMF algorithm proposed in (Banerjee et al., 2005b).
- CEM denotes the hard-movMF algorithm proposed in (Banerjee et al., 2005b).
- DAEM is the deterministic annealing version of soft-movMF proposed in (Zhong and Ghosh, 2005).
- Skmeans denotes the spherical k-means algorithm (Dhillon and Modha, 2001), which is also a simplified version of soft-movMF, where $\kappa_h = \kappa \to \infty$, $\alpha_h = \alpha$, for all h. It is also a restricted version of hard-movMF where $\kappa_h = \kappa$, $\alpha_h = \alpha$, for all h.

Notice that, in the context of text document clustering, it has been empirically shown on numerous real-world datasets, that the aforementioned baselines perform better than several existing clustering and co-clustering algorithms, namely: k-means using the euclidean distance, generative mixture models using Bernoulli, Gaussian, and Multinomial distributions,

spectral co-clustering, and Latent Dirichlet Allocation (LDA) (Blei et al., 2003). Therefore, we do not include these approaches in our comparisons. For further details see (Zhong and Ghosh, 2005; Gopal and Yang, 2014).

5.7.1 Simulated data sets

In the following, we validate the correctness of our model and implementations, on several simulated datasets corresponding to various particular situations, illustrated in Figure 5.4, namely balanced co-clusters, unbalanced co-clusters, clusters with equal concentration and with different concentrations and last overlapping clusters. We consider five different simulated datasets, each of them consists of a sample of 5000 unit vectors from a *Diagonal Block Mixture* of three 1000-dimensional vMF distributions. The different simulated datasets, i.e, sdata1, ..., sdata5 are described in Table 5.2. In this table, α, κ and μ denote the true parameters, while $\hat{\alpha}$, $\hat{\kappa}$ and $\hat{\mu}$ denote the estimated ones.



(a) (b) (c) (d) (e)

Fig. 5.4 Simulated datasets reorganized according to row and column partitions: (a) sdata1, (b) sdata2, (c) sdata3, (d) sdata4 (e) sdata5.

As this table shows clearly, when co-clusters are reasonably separated, both $\text{EM}_{\mathbf{b}}$ and $\text{CEM}_{\mathbf{b}}$ provide excellent performances, even in situations where data exhibit unbalanced cluster sizes and concentration parameters. However, in the case of poorly separated co-clusters, i.e, sdata5, $\text{EM}_{\mathbf{b}}$ and $\text{CEM}_{\mathbf{b}}$ suffer from a convergence to a poor local maximum of the likelihood function, due to hard column assignments at the beginning, i.e, during the first iterations the row clusters are mixed, hence each column is involved to describe several row clusters. The simulated annealing variant $\text{SAEM}_{\mathbf{b}}$, however, overcomes this difficulty thanks to stochastic assignments in the beginning and thereby, yields excellent performance on sdata5 although the co-clusters are poorly separated.

Table 5.2 Comparison of true and estimated parameters by our algorithms on different simulated dataset, (α,κ,μ) denote the true parameters while $(\hat{\alpha},\hat{\kappa},\hat{\mu})$ denote the estimated parameters.

Data	Components	True Parameters			Algorithms	Parameter Estimation				
		α	κ	μ_{hh}		$\hat{\alpha}$	$\lvert\alpha-\hat{\alpha}\rvert$	$\hat{\kappa}$	$\lvert\kappa-\hat{\kappa}\rvert$	$\mu^T\hat{\mu}$
sdata1	cluster1	0.34	500.00	$1/\sqrt{340}$	$\text{CEM}_\mathbf{b}$	0.339	0.001	501.38	1.38	1.00
					$\text{EM}_\mathbf{b}$	0.339	0.001	499.87	0.13	1.00
	cluster2	0.33	500.00	$1/\sqrt{330}$	$\text{CEM}_\mathbf{b}$	0.328	0.002	500.90	0.90	1.00
					$\text{EM}_\mathbf{b}$	0.329	0.001	499.33	0.67	1.00
	cluster3	0.33	500.00	$1/\sqrt{330}$	$\text{CEM}_\mathbf{b}$	0.333	0.003	500.56	0.56	1.00
					$\text{EM}_\mathbf{b}$	0.332	0.002	500.42	0.42	1.00
sdata2	cluster1	0.70	320.00	$1/\sqrt{340}$	$\text{CEM}_\mathbf{b}$	0.691	0.009	320.72	0.72	1.00
					$\text{EM}_\mathbf{b}$	0.700	0.00	319.62	0.38	1.00
	cluster2	0.25	400.00	$1/\sqrt{330}$	$\text{CEM}_\mathbf{b}$	0.261	0.011	398.23	1.77	1.00
					$\text{EM}_\mathbf{b}$	0.250	0.00	401.51	1.51	1.00
	cluster3	0.05	500.00	$1/\sqrt{330}$	$\text{CEM}_\mathbf{b}$	0.048	0.002	497.75	2.25	1.00
					$\text{EM}_\mathbf{b}$	0.050	0.00	499.28	0.72	1.00
sdata3	cluster1	0.34	320.00	$1/\sqrt{700}$	$\text{CEM}_\mathbf{b}$	0.345	0.005	319.82	0.18	0.998
					$\text{EM}_\mathbf{b}$	0.345	0.005	319.83	0.17	1.00
	cluster2	0.33	400.00	$1/\sqrt{250}$	$\text{CEM}_\mathbf{b}$	0.330	0.00	399.11	0.89	1.00
					$\text{EM}_\mathbf{b}$	0.330	0.00	399.30	0.70	1.00
	cluster3	0.33	500.00	$1/\sqrt{50}$	$\text{CEM}_\mathbf{b}$	0.325	0.005	487.82	12.18	0.980
					$\text{EM}_\mathbf{b}$	0.325	0.005	500.40	0.40	1.00
sdata4	cluster1	0.70	320.00	$1/\sqrt{700}$	$\text{CEM}_\mathbf{b}$	0.697	0.003	320.20	0.20	1.00
					$\text{EM}_\mathbf{b}$	0.702	0.002	320.10	0.10	1.00
	cluster2	0.250	400.00	$1/\sqrt{250}$	$\text{CEM}_\mathbf{b}$	0.254	0.004	400.37	0.37	1.00
					$\text{EM}_\mathbf{b}$	0.248	0.002	399.77	0.23	1.00
	cluster3	0.05	500.00	$1/\sqrt{50}$	$\text{CEM}_\mathbf{b}$	0.049	0.001	498.10	1.90	1.00
					$\text{EM}_\mathbf{b}$	0.05	0.00	501.12	1.12	1.00
sdata5	cluster1	0.34	70.00	$1/\sqrt{340}$	$\text{CEM}_\mathbf{b}$	0.34	0.00	59.22	10.78	0.334
					$\text{EM}_\mathbf{b}$	0.34	0.00	45.43	24.57	0.387
					$\text{SEM}_\mathbf{b}$	0.33	0.01	53.68	16.32	0.793
					$\text{SAEM}_\mathbf{b}$	0.33	0.01	69.96	0.04	0.991
	cluster2	0.33	70.00	$1/\sqrt{330}$	$\text{CEM}_\mathbf{b}$	0.330	0.00	55.66	14.34	0.430
					$\text{EM}_\mathbf{b}$	0.340	0.01	43.41	26.59	0.339
					$\text{SEM}_\mathbf{b}$	0.33	0.00	58.69	11.31	0.850
					$\text{SAEM}_\mathbf{b}$	0.33	0.00	70.38	0.38	0.995
	cluster3	0.33	70.00	$1/\sqrt{330}$	$\text{CEM}_\mathbf{b}$	0.330	0.00	53.39	16.61	0.487
					$\text{EM}_\mathbf{b}$	0.320	0.01	43.00	27.00	0.388
					$\text{SEM}_\mathbf{b}$	0.34	0.01	55.12	14.88	0.805
					$\text{SAEM}_\mathbf{b}$	0.34	0.01	70.13	0.13	0.989

5.7.2 Real-world datasets

As a practical example we select the domain of text mining and we concentrate on the challenging task of document clustering using high dimensional sparse document-term matrices. We retain six popular benchmark text datasets: CSTR used in (Li, 2005), CLASSIC4[1], WEBACE, the 20-newsgroups data NG20, SPORTS used in (Zhong and Ghosh, 2005), and TDT2[2]. All these datasets are carefully selected to represent various particular challenging situations in clustering: balanced clusters, unbalanced clusters, different number of clusters,

[1] http://www.dataminingresearch.com/

[2] http://www.cad.zju.edu.cn/home/dengcai/

different sizes, different degrees of cluster overlap (i.e, well separated clusters and poorly separated clusters). The characteristics of these different datasets are summarized in Table 5.3.

Note that, in contrast to document clusters, the true word cluster labels are not known in the above datasets, this is the reason why we mainly concentrate our experiments on document clustering. However, under our formulation row clusters induce column clusters and vice versa, hence the quality of row clustering is informative about the quality of column clustering. It is reasonable to expect that a good row partitioning induces a good column partitioning and vice versa.

Table 5.3 Description of Datasets

Datasets	Characteristics				
	#Documents	#Words	#Clusters	Sparsity (%)	Balance[3]
CSTR	475	1000	4	96.60	0.399
WEBACE	2340	1000	20	91.83	0.169
CLASSIC4	7094	5896	4	99.41	0.323
NG20	19949	43586	20	99.82	0.991
SPORTS	8580	14870	7	99.14	0.036
TDT2	9394	36771	30	99.64	0.028

Evaluation measures

Evaluating clustering results is not a trivial task, however, when the true category labels are known, a commonly used approach to validate clustering results consists to compare the estimated partition with the true one. To this end, several measures have been proposed, in our experiments we retain two widely used measures to assess the quality of clustering, namely the Normalized Mutual Information NMI (Strehl and Ghosh, 2003) and the Adjusted Rand Index ARI (Milligan and Cooper, 1986; Hubert and Arabie, 1985).

Intuitively, NMI quantifies how much the estimated clustering is informative about the true clustering, while the ARI measures the degree of agreement between an estimated clustering and a reference clustering. Both NMI and ARI are equal to 1 if the resulting clustering is identical to the true one, and close to zero for a random clustering.

Experimental setting

In all our experiments we use the TF-IDF normalized data representation, more precisely we use the TF-IDF weighting scheme proposed in Scikit-learn (Pedregosa et al., 2011). For each dataset we set g to the real number of clusters, and in order to make fair comparisons,

[3] The balance coefficient is the ratio of the minimum cluster size to the maximum cluster size.

all non-stochastic algorithms are initialized using the same row partition resulting from Skmeans[4] started using a random initial point, unless stated otherwise. For our algorithms, we further initialize the concentration parameters to 10 and the centroids to random initial vectors, in order to be able to estimate the initial column partition. Concerning our stochastic variants, i.e, $\text{SEM}_{\mathbf{b}}$, $\text{CAEM}_{\mathbf{b}}$ and $\text{SAEM}_{\mathbf{b}}$ they are initialized using the same random row and column partitions.

Evaluation of row clustering

Tables 5.4 and 5.5, summarize the results of the different methods in terms of NMI and ARI, over all datasets. All results are averaged over thirty different starting points, obtained using the initialization strategy described above. Between brackets, we report the results corresponding to the trial with the highest criterion. As it is evident from these tables, our *dbmovMFs*-based algorithms, i.e, $\text{EM}_{\mathbf{b}}$, $\text{CEM}_{\mathbf{b}}$ and dbSkmeans exhibit high performances as opposed to *movMFs*-based method. In fact, $\text{EM}_{\mathbf{b}}$, $\text{CEM}_{\mathbf{b}}$ and dbSkmeans achieve better performances than EM, CEM and Skmeans, in almost all situations, except in terms of ARI on CLASSIC4 and NMI on SPORTS, however, the difference is not significant. We note that both $\text{EM}_{\mathbf{b}}$ and $\text{CEM}_{\mathbf{b}}$ perform somewhat better than dbSkmeans, in fact the former generalize dbSkmeans. Once again we observe only a slight difference between $\text{EM}_{\mathbf{b}}$ and $\text{CEM}_{\mathbf{b}}$, which is statistically not significant.

Advantages of $\text{SAEM}_{\mathbf{b}}$ and $\text{CAEM}_{\mathbf{b}}$

We observe that our simulated annealing variants, $\text{SAEM}_{\mathbf{b}}$ and $\text{CAEM}_{\mathbf{b}}$, provide the best performances in almost all situations (although they are initialized randomly), except on SPORTS, where $\text{SEM}_{\mathbf{b}}$ achieves substantially better results than the other methods. As opposed to $\text{EM}_{\mathbf{b}}$, $\text{CEM}_{\mathbf{b}}$ and dbSkmeans which depend strongly on their starting positions, the stochastic variants, i.e, $\text{SAEM}_{\mathbf{b}}$ and $\text{CAEM}_{\mathbf{b}}$ are not sensitive to their initial positions. For instance, in the case of high overlapping clusters as in CLASSIC4, NG20 and SPORTS, we observe that $\text{EM}_{\mathbf{b}}$, $\text{CEM}_{\mathbf{b}}$ and dbSKmeans suffer from a convergence to a poor local maximum of the likelihood function, when they are initialized randomly. The $\text{SAEM}_{\mathbf{b}}$ and $\text{CAEM}_{\mathbf{b}}$ algorithms, however, avoid this difficulty and converge to a better local optimum of the likelihood function, see Table 5.6; this is due to the advantage of $\text{SEM}_{\mathbf{b}}$ in the begining. Figure 5.5 illustrates the behaviour of $\text{SAEM}_{\mathbf{b}}$ and its advantage compared to $\text{SEM}_{\mathbf{b}}$, the same behaviour is observed for $\text{CAEM}_{\mathbf{b}}$.

[4] We found that all algorithms, except stochastic variants, provide substantially better results when they are initialized using Skmeans (in almost all situations)

Table 5.4 Comparison of average NMI and ARI, on CSTR, CLASSIC4 and WEBACE datatsets.

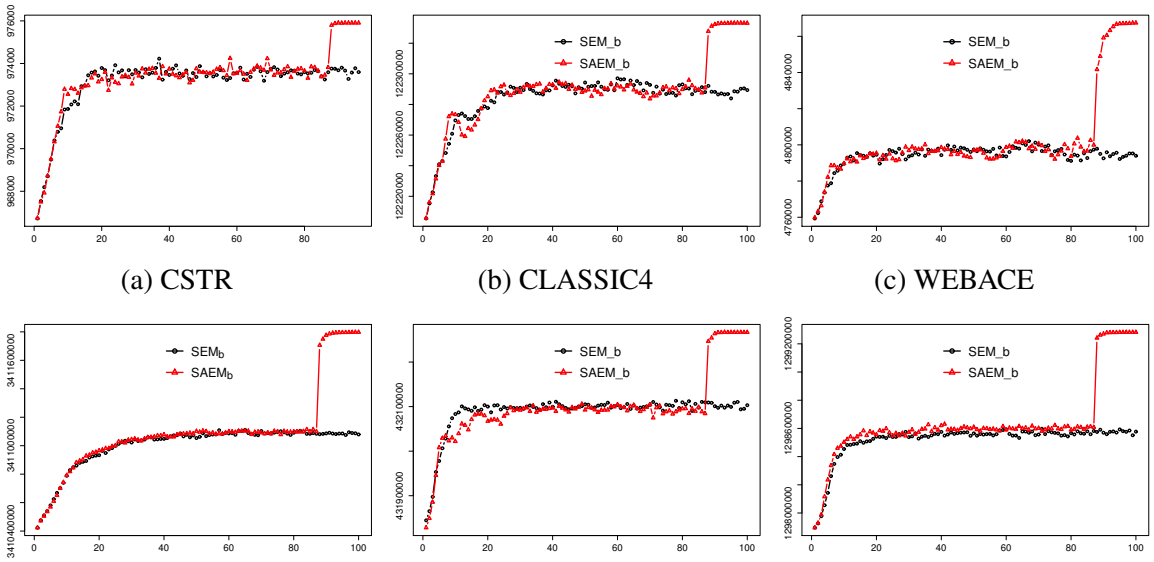
	CSTR		CLASSIC4		WEBACE	
	NMI	ARI	NMI	ARI	NMI	ARI
Skmeans	0.732±0.026 (0.759)	0.772±0.025 (0.807)	0.591±0.020 (0.595)	0.468±0.011 (0.476)	0.613±0.008 (0.620)	0.423±0.026 (0.394)
CEM	0.734±0.025 (0.759)	0.774±0.024 (0.807)	0.413±0.011 (0.410)	0.199±0.018 (0.194)	0.619±0.011 (0.623)	0.398±0.021 (0.412)
EM	0.741±0.026 (0.768)	0.777±0.026 (0.808)	0.406±0.013 (0.403)	0.190±0.015 (0.184)	0.614±0.014 (0.623)	0.385±0.034 (0.397)
DAEM	0.779±0.013 (0.783)	0.813±0.014 (0.817)	0.591±0.002 (0.592)	0.471±0.002 (0.472)	0.620±0.008 (0.628)	0.427±0.022 (0.468)
dbSkmeans	0.753±0.016 (0.771)	0.803±0.014 (0.816)	0.647±0.002 (0.653)	0.460±0.002 (0.464)	0.616±0.008 (0.620)	0.469±0.033 (0.507)
CEM_b	0.754±0.024 (0.789)	0.804±0.022 (0.830)	0.660±0.003 (0.665)	0.467±0.003 (0.473)	0.623±0.011 (0.637)	0.479±0.038 (0.519)
EM_b	0.754±0.022 (0.792)	0.803±0.022 (0.837)	0.660±0.002 (0.668)	0.466±0.002 (0.473)	0.624±0.008 (0.639)	0.481±0.025 (0.523)
SEM_b	0.776±0.022 (0.807)	0.820±0.024 (0.846)	0.691±0.031 (0.721)	0.705±0.053 (0.735)	0.567±0.044 (0.597)	0.582±0.035 (0.588)
$CAEM_b$	0.794±0.014 (0.817)	0.833±0.013 (**0.851**)	0.735±0.033 (0.746)	0.751±0.048 (0.772)	0.640±0.007 (0.658)	0.666±0.019 (0.688)
$SAEM_b$	0.795±0.011 (**0.821**)	0.830±0.010 (**0.851**)	0.746±0.023 (**0.773**)	0.756±0.039 (**0.798**)	0.644±0.015 (**0.661**)	0.656±0.021 (**0.689**)

Table 5.5 Comparison of average NMI and ARI, on NG20, SPORTS and TDT2 datasets.

	NG20		SPORTS		TDT2	
	NMI	ARI	NMI	ARI	NMI	ARI
Skmeans	0.542±0.013 (0.555)	0.375±0.016 (0.379)	0.614±0.044 (0.627)	0.405±0.053 (0.442)	0.790±0.012 (0.801)	0.492±0.031 (0.514)
CEM	0.467±0.013 (0.484)	0.149±0.021 (0.150)	0.446±0.048 (0.455)	0.151±0.067 (0.177)	0.750±0.021 (0.769)	0.436±0.048 (0.466)
EM	0.465±0.013 (0.481)	0.143±0.020 (0.141)	0.444±0.049 (0.453)	0.149±0.067 (0.174)	0.751±0.019 (0.771)	0.438±0.041 (0.489)
DAEM	0.556±0.018 (0.576)	0.369±0.022 (0.393)	0.618±0.005 (0.620)	0.398±0.016 (0.416)	0.795±0.007 (0.806)	0.462±0.011 (0.456)
dbSkmeans	0.579±0.013 (0.591)	0.417±0.020 (**0.440**)	0.572±0.032 (0.615)	0.451±0.084 (0.601)	0.794±0.012 (0.808)	0.636±0.028 (0.682)
CEM_b	0.582±0.011 (0.594)	0.388±0.024 (0.403)	0.558±0.039 (0.608)	0.508±0.080 (0.602)	0.799 ±0.014 (0.817)	0.657±0.032 (0.719)
EM_b	0.585±0.010 (0.601)	0.390±0.025 (0.425)	0.564±0.037 (0.617)	0.517±0.072 (0.605)	0.799±0.015 (**0.821**)	0.658±0.034 (0.699)
SEM_b	0.534±0.013 (0.553)	0.347±0.022 (0.367)	0.670±0.018 (**0.700**)	0.713±0.033 (**0.767**)	0.718±0.020 (0.748)	0.619±0.022 (0.656)
$CAEM_b$	0.605±0.010 (0.608)	0.381±0.019 (0.385)	0.601±0.013 (0.612)	0.602±0.021 (0.613)	0.794±0.011 (0.806)	0.723±0.020 (0.749)
$SAEM_b$	0.610±0.011 (**0.615**)	0.393±0.012 (0.390)	0.613±0.015 (0.621)	0.620±0.016 (0.625)	0.791±0.010 (**0.821**)	0.709±0.021 (**0.752**)

Table 5.6 Comparison of classification log-likelihood.

	CSTR	CLASSIC4	WEBACE	NG20	SPORTS	TDT2
EM_b	976,533.3	122,329,895	4,866,978	3,411,592,328	432,277,941	1,299,234,559
CEM_b	976,520.4	122,329,033	4,866,769	3,411,588,346	432,276,368	1,299,232,734
$CAEM_b$	976,540	122,341,547	4,867,537	3,411,836,825	432,282,583	1,299,303,290
$SAEM_b$	**976,547**	**122,341,562**	**4,868,739**	**3,411,838,557**	**432,286,341**	**1,299,304,500**



(a) CSTR (b) CLASSIC4 (c) WEBACE

(d) NG20 (e) SPORTS (f) TDT2

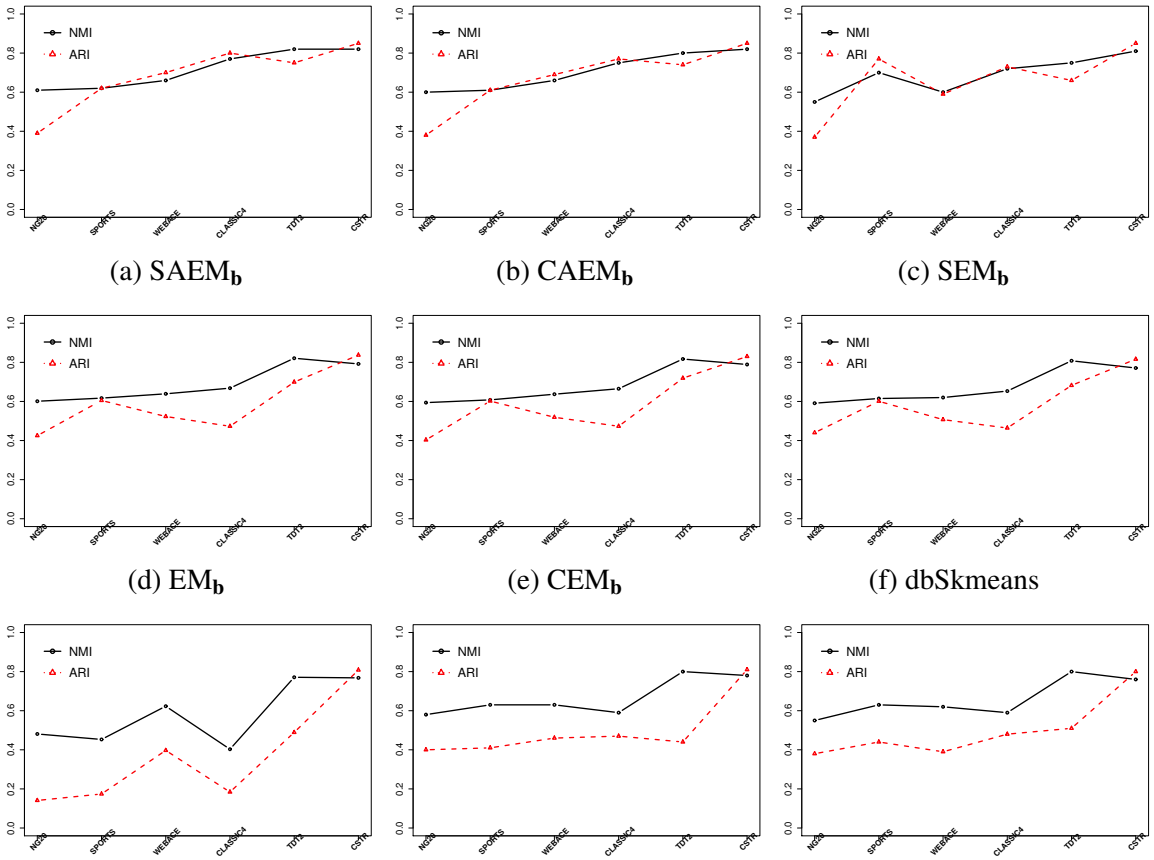
Fig. 5.5 Behaviour of log-likelihood over iterations, on the different datasets.

NMI vs ARI

Tables 5.4 and 5.5 show that our algorithms provide high performances in terms of both NMI and ARI, while the other *movMFs*-based methods sometimes provide good NMI but low ARI as this is the case with almost all datasets, except CSTR. Figure 5.6 confirms this remark; we observe that the behavior of NMI and that of ARI are more often in keeping with our algorithms rather than with *movMFs*-based algorithms[5], i.e, DAEM, EM, CEM and Skmeans.

The explanation is that *movMFs*-based clustering methods tend to merge small clusters and try to split larger ones into comparably sized clusters, as it has been already emphasized in (Banerjee et al., 2005b). In fact, unlike ARI, the NMI measure is less sensitive to clusters merging and/or splitting. Our *dbmovMFs*-based algorithms, however, thanks to the centroids orthonormality assumption, avoid the above difficulty, and are able to discover large as well as small clusters (see Figure 5.7 and confusion matrices of Table 5.7). Furthermore, as we can see from Figure 5.7, $\text{EM}_{\mathbf{b}}$ reveals a more interesting structure (i.e, block diagonal) than Skmeans.

[5] For presentation purpose we omit CEM in Figure 5.6, but from tables 5.4 and 5.5 it is straightforward to verify that the behaviour of CEM is too close to that of EM

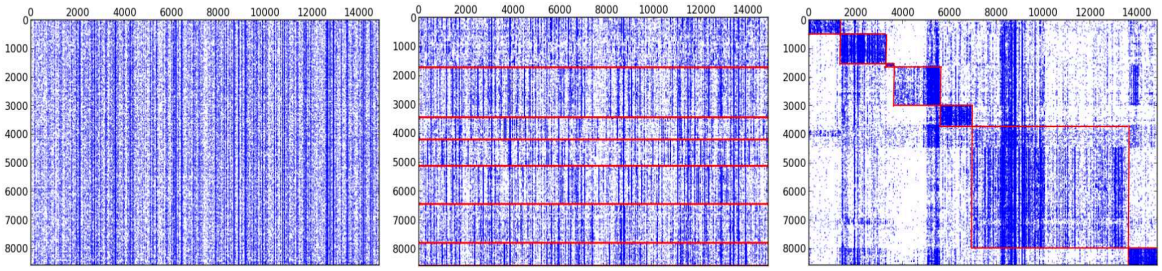


(a) SAEM$_\mathbf{b}$ (b) CAEM$_\mathbf{b}$ (c) SEM$_\mathbf{b}$

(d) EM$_\mathbf{b}$ (e) CEM$_\mathbf{b}$ (f) dbSkmeans

(g) EM (h) DAEM (i) Skmeans

Fig. 5.6 Best NMI and ARI over all datasets for each method. Our algorithms provide good performances in terms both NMI and ARI while *movMFs*-based approach sometimes provide good NMI but low ARI. For instance, on SPORTS EM$_\mathbf{b}$ provides NMI = 0.62 and ARI = 0.61 while SKmeans provides NMI = 0.63 and ARI = 0.44. (see confusion matrices of Table 5.7)



(a) (b) (c)

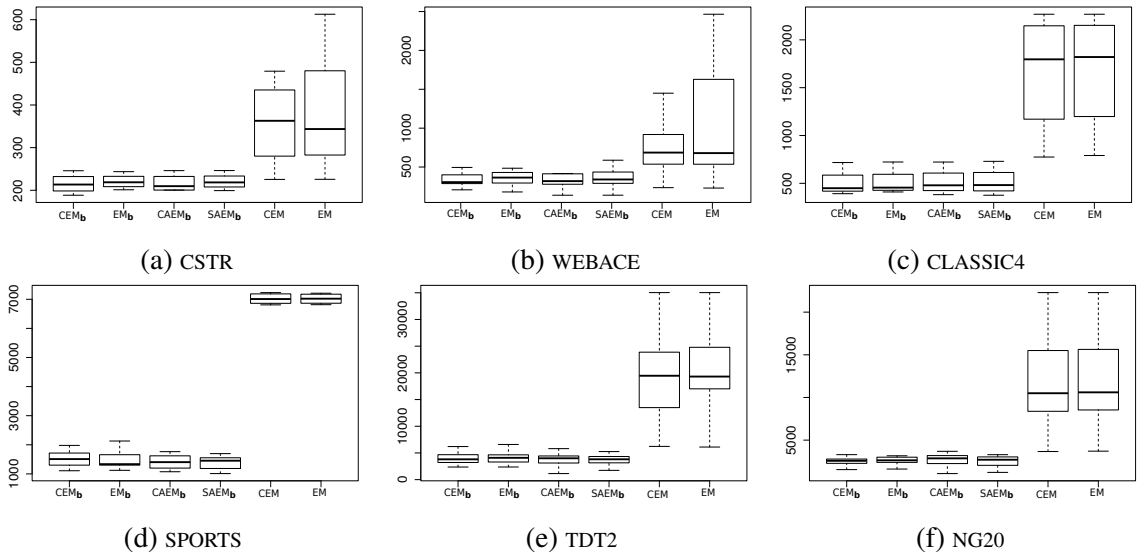
Fig. 5.7 Visualization of SPORTS dataset: (a) original, (b) reorganized according to Skmeans's row partition, (c) reorganized according to EM$_\mathbf{b}$'s row and column partitions.

Impact of concentration parameters

We observe that the Skmeans algorithm which is a restricted version of *movMFs* where all clusters share the same proportion and the same concentration parameter, yields better

Table 5.7 SPORTS dataset: confusion matrices crossing the row clusters obtained by both algorithms (rows) and the true row clusters (columns). The column $z_{.h}$ indicates the sizes of clusters.

	EM_b (NMI = 0.617, ARI = 0.605)								Skmeans (NMI = 0.627 , ARI = 0.442)							
	1	2	3	4	5	6	7	$z_{.h}$	1	2	3	4	5	6	7	$z_{.h}$
1	3338	219	41	27	487	48	82	4242	1705	1	0	1	1	0	0	1708
2	8	1008	0	0	16	0	0	1032	0	904	0	0	1	0	1	906
3	24	22	5	0	39	5	7	102	1	1	0	0	798	1	0	801
4	36	138	0	0	449	2	2	627	1326	0	0	0	6	0	0	1332
5	3	14	0	1	1345	1	10	1374	81	30	24	6	1171	5	4	1321
6	3	8	99	94	10	280	0	494	298	473	121	115	369	330	23	1729
7	0	1	0	0	0	0	708	709	1	1	0	0	0	0	781	783



(a) CSTR (b) WEBACE (c) CLASSIC4

(d) SPORTS (e) TDT2 (f) NG20

Fig. 5.8 Distribution of concentration parameters

results than the other *movMFs*-based algorithms, as it has been already emphasized by Gopal and Yang (2014). The low performance of EM and CEM as opposed to Skmeans, is due to high concentration parameters κ_h in the normalization terms $c_d(\kappa_h)$ that involve Bessel functions. As it has been highlighted by Banerjee et al. (2005b), in the case of large positive matrices, all the data lie on the first orthant of a d-dimentional hypersphere, thereby the concentration of such data is implicitly high. As a result, the concentration parameters κ_h of the vMF distributions are high and increase exponentially with the dimensionality of the data. Our model *dbmovMFs*, however, alleviates this problem of high concentration parameters, thanks to its implicitly adaptive dimensionality reduction property. In Figure 5.8 are reported the distribution of the concentration parameters estimated by our *dbmovMFs*-based algorithms and *movMFs*-based algorithms. We clearly observe that our algorithms,

$\text{CEM}_\mathbf{b}$, $\text{EM}_\mathbf{b}$, $\text{CAEM}_\mathbf{b}$ and $\text{SAEM}_\mathbf{b}$, yield substantially lower concentration parameters than EM and CEM.

Interpretation of column clusters

Although we focused on document clustering, notice that our algorithms offer word clusters. Each of them describes a single document cluster and thereby, allows to understand the semantic meaning of document clusters. Tables 5.8, 5.9 and 5.10 provide typical examples, where the four, seven and top six word clusters resulting from $\text{SAEM}_\mathbf{b}$ on CLASSIC4 and TDT2 respectively, are represented by their top 15 terms.

The top terms of each co-cluster were obtained by keeping only the terms that appear in most documents in the considered cluster. The obtained word clusters are meaningful and help to make sense of the corresponding document clusters; they are semantically coherent. More interestingly, and as illustrated in Table 5.9 on the CLASSIC4 datatset, when the requested number of clusters (7) is greater than the natural number of clusters (4), our algorithm try to split the natural clusters into semantically coherent sub-clusters. Hence, when the true number of clusters is not known, one can request a relatively high number of clusters and then merge them adequately, based on the obtained word clusters.

Table 5.8 Word clusters resulting from $\text{SAEM}_\mathbf{b}$ on the CLASSIC4 dataset. Each cluster is represented by its top 15 terms, sorted according to their popularity. Clusters C1, C2, C3 and C4 correspond respectively to CACM, CISI, CRANFIELD and MEDLINE.

C1	C2	C3	C4
algorithm	librari	flow	cell
program	inform	boundari	patient
system	scienc	layer	rat
comput	research	pressur	growth
languag	book	heat	blood
method	servic	wing	acid
function	scientif	number	hormon
gener	index	bodi	tissu
problem	journal	solut	diseas
data	retriev	shock	cancer
time	studi	theori	treatment
structur	develop	equat	tumor
integr	literatur	mach	renal
process	public	effect	kidnei
matrix	univers	plate	dai

Table 5.9 Clustering of CLASSIC4 into 7 co-clusters by SAEM$_{\mathbf{b}}$: the obtained word clusters represented by their top 15 terms, sorted according to their popularity.

CACM		CISI	CRANFIELD	MEDLINE		
C1	C2	C3	C4	C5	C6	C7
algorithm	program	librari	flow	cell	children	patient
function	system	inform	boundari	rat	child	cancer
integr	comput	scienc	layer	growth	speech	ventricular
matrix	languag	research	pressur	acid	autist	diseas
polynomi	data	book	heat	hormon	anxieti	arteri
permut	problem	servic	wing	tissu	disord	therapi
squar	structur	index	number	activ	joint	treatment
invers	process	scientif	bodi	dna	visual	pulmonari
fit	time	retriev	solut	protein	syndrome	defect
random	gener	journal	shock	kidnei	autism	breast
complex	techniqu	studi	equat	antigen	mother	aortic
root	code	develop	theori	dai	earli	cardiac
exponenti	algol	literatur	mach	human	childhood	renal
error	oper	docum	effect	len	symptom	hydrocephalu
gamma	set	public	plate	cultur	infantil	hypothermia

Table 5.10 The top 6 word clusters resulting from SAEM$_{\mathbf{b}}$ on dataset TDT2. Each cluster is represented by its top 15 terms, sorted according to their popularity.

C10	C24	C28	C12	C25	C3
iraq	percent	lewinsky	suharto	nuclear	tobacco
un	asian	starr	indonesia	pakistan	smoking
weapons	economic	clinton	jakarta	india	industry
iraqi	financial	president	indonesian	test	bill
inspectors	economy	white	habibie	treaty	senate
baghdad	market	house	imf	kashmir	legislation
united	stock	monica	suhartos	sharif	companies
saddam	crisis	grand	political	islamabad	settlement
annan	yen	jury	student	conducted	congress
council	dollar	counsel	reform	five	cigarette
military	currency	independent	protests	ban	republicans
secretary	billion	investigation	rupiah	device	documents
sites	banks	intern	riots	intelligence	tax
security	stocks	sexual	step	explosions	lawsuits
inspections	growth	lawyers	demonstration	powers	democrats

5.7.3 Assessing the number of clusters

Often in practice the number of clusters is not known in advance and should be determined by the user. Assessing the number of clusters is, however, not straightforward and still remains one of the biggest challenges in machine learning. Fortunately, in our case we can rely to

well-established statistical theory of model selection since our algorithms are based on the maximization of the likelihood. More precisely, we can use information criteria, such the Akaike information criterion (AIC) (Akaike, 1974, 1992), AIC3 (Bozdogan, 1994), Bayesian information criterion (BIC) (Schwarz et al., 1978) or integrated classification likelihood (ICL) (Biernacki et al., 2000). The aforementioned information criteria (IC) measure the quality of a model given some observed data, and they are formally given by

$$IC(k) = -2\ln\hat{L} + 2\gamma \times k \tag{5.13}$$

where k denotes the number of parameters to be estimated, γ a penalty coefficient and $\hat{L}$ is the maximized value of the log-likelihood function. With $\gamma = 1$ we have the AIC criterion, $\gamma = 3/2$ we obtain AIC3 and $\gamma = (\log n)/2$ leads to the BIC criterion where n is the simple size—the number of rows/documents in our case. And last, ICL can be obtained from (5.13) by replacing $\hat{L}$ with the complete data log-likelihood $\hat{L}_c$ and setting γ to $(\log n)/2$.

Intuitively the above information criteria penalize the log-likelihood according to the complexity of the model in terms of the number of parameters to be estimated; the lower the information criterion, the better is the model quality. Hence, information criteria can be used to select the best model, which yields the lowest value in terms of the selected information criterion, among a set of models. In our case we are interested in selecting the number of clusters, we therefore study the behaviour of AIC, AIC3 and BIC when varying the number of clusters, on each dataset.

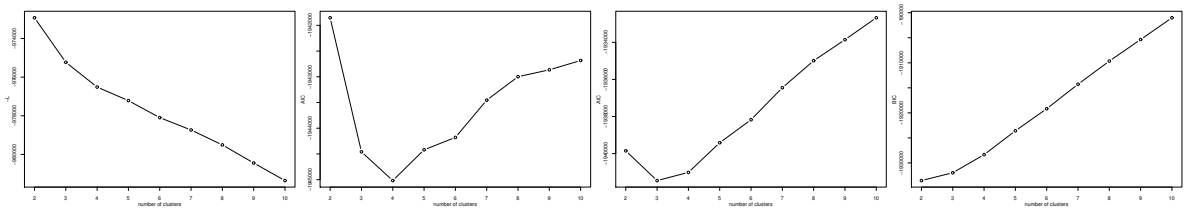
Note that this problem has been investigated by many authors for different mixture models including the vMF mixture model (Bouberima et al., 2010) where the authors have performed Monte Carlo simulations taking into account the overlap degree of clusters and the size of data. Although their experiments were performed on moderate size and non sparse data, the authors emphasized the good behavior of AIC and AIC3 compared to BIC and ICL.

For our experiments, we use the value of $\hat{L}$ resulting from the simulated annealing *dbmovMFs* ($\texttt{SAEM}_{\mathbf{b}}$) algorithm as it seems to provide the best performances in almost all cases. Determining the number of free parameters in *dbmovMFs* is not straightforward due to the mixing of continuous and discrete parameters. The parameters α and κ contain $g-1$ and g parameters, respectively. The column partition $\mathbf{w}$—that is treated as a parameter in our case—contains the same number of parameters whatever the number of column clusters. Nevertheless, as pointed by van Dijk et al. (2009), the number of possible values for each parameter increases with the number of clusters, it is therefore more convenient to consider $\mathbf{w}$ under its matrix form $\mathbf{W}$ of size $(d \times g)$, where each row indicates to which cluster the corresponding column belongs. The effective number of free parameters in *dbmovMFs* is

therefore $k = g \times (d+2) - 1$ due to g concentration parameters κ_h's, $g-1$ mixing proportions α_h's and $d \times g$ column cluster parameters w_{jh}'s.

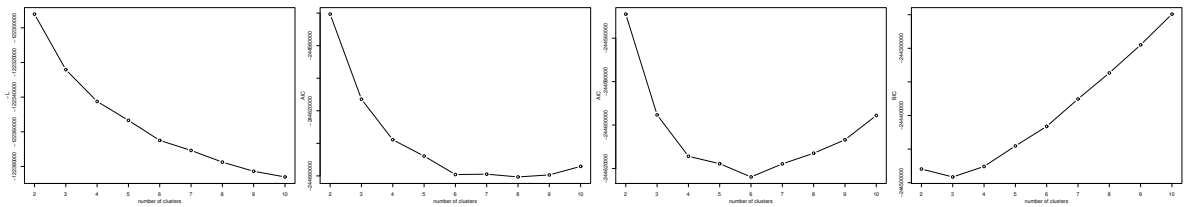
In figures 5.9 to 5.14 are depicted the values of the log-likelihood, AIC, AIC3 and BIC as a function of the number of clusters, over different data sets (we found that the ICL criterion, not reported here for presentation purpose, behaves like BIC). First of all, we observe that the log-likelihood can clearly not be used to assess the right number of clusters as it varies, as expected, monotonically when the number of clusters increases. The penalty term of the BIC criterion seems to be too strong due to the high number of parameters to be estimated, as a result BIC, increases monotonically and, thereby, does not allows us to identify the number of clusters, except on CLASSIC4 and WEBACE where BIC and ICL identify 3 and 6 clusters, respectively, instead of 4 and 20. From figures 5.9 to 5.14 we observe that AIC and AIC3 are able to identify a number of clusters that is close or even equal to the right number of clusters on almost all data sets, except on TDT2 where both criteria under estimate the number of clusters. This suggests that, in our case, a convenient penalty term γ lies between 1 and $3/2$.

Furthermore, the AIC criterion seems to be slightly better than AIC3 since the latter tends to underestimate the right number of clusters on some data sets such as NG20 and CSTR. Another notable result is that, on the CLASSIC4 data set, that contains 4 classes,



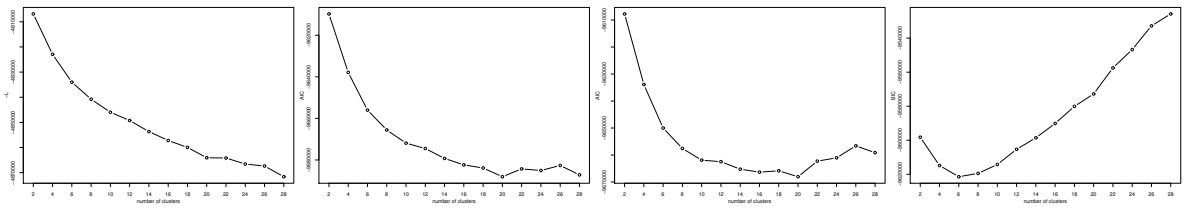
(a) $-\hat{L}$ (b) AIC (c) AIC3 (d) BIC

Fig. 5.9 Behaviour of the log-likelihood, AIC, AIC3 and BIC when varying the number of clusters on the CSTR dataset (True number of classes: 4).



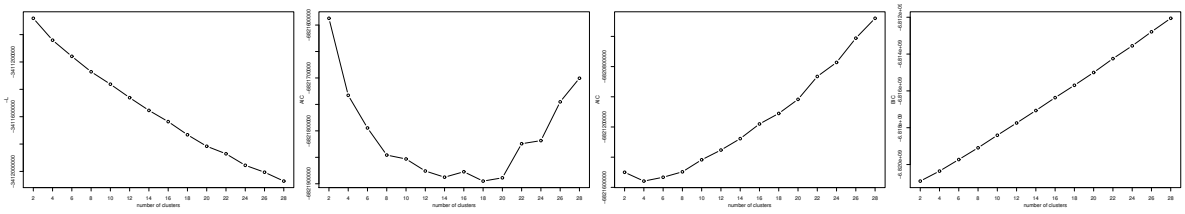
(a) $-\hat{L}$ (b) AIC (c) AIC3 (d) BIC

Fig. 5.10 Behaviour of the log-likelihood, AIC, AIC3 and BIC when varying the number of clusters on the CLASSIC4 dataset (True number of classes: 4).



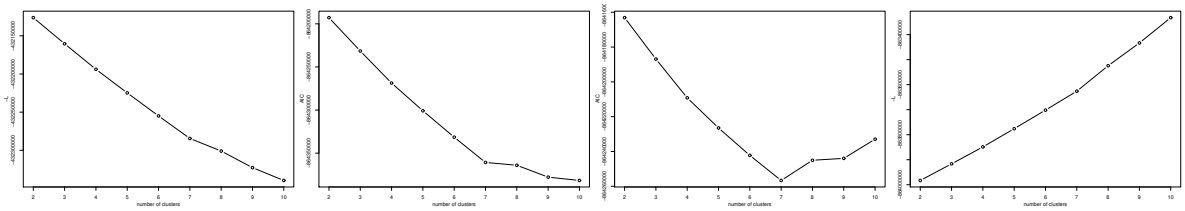
(a) $-\hat{L}$ (b) AIC (c) AIC3 (d) BIC

Fig. 5.11 Behaviour of the log-likelihood, AIC, AIC3 and BIC when varying the number of clusters on the WEBACE dataset (True number of classes: 20).



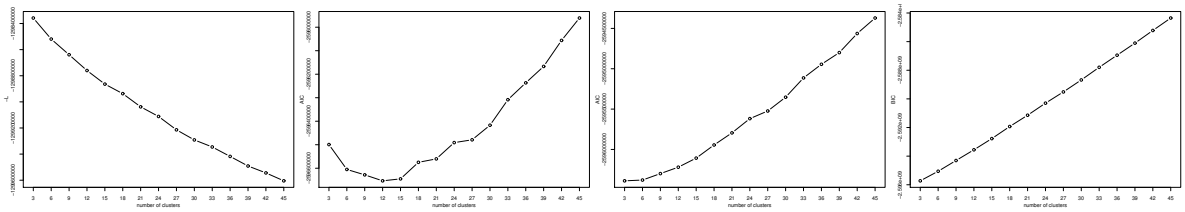
(a) $-\hat{L}$ (b) AIC (c) AIC3 (d) BIC

Fig. 5.12 Behaviour of the log-likelihood, AIC, AIC3 and BIC when varying the number of clusters on the NG20 dataset (True number of classes: 20).



(a) $-\hat{L}$ (b) AIC (c) AIC3 (d) BIC

Fig. 5.13 Behaviour of the log-likelihood, AIC, AIC3 and BIC when varying the number of clusters on the SPORTS dataset (True number of classes: 7).



(a) $-\hat{L}$ (b) AIC (c) AIC3 (d) BIC

Fig. 5.14 Behaviour of the log-likelihood, AIC, AIC3 and BIC when varying the number of clusters on the TDT2 dataset (True number of classes: 30).

AIC3 suggests 6 clusters while AIC suggests a number of clusters between 6 and 8, this is, however, not an issue. In fact, as illustrated in table 5.9 some classes in CLASSIC4 are composed of several semantically coherent sub-clusters. Thereby, an interesting approach would be to consider the number of clusters determined by AIC/AIC3 as a starting point, which can then be tuned by exploring the top terms of the obtained word clusters so as to decide whether the number of clusters should be increased, decreased or kept unchanged.

5.8 Conclusion and perspectives

In this chapter we proposed a novel generative model based on a mixture of vMF distributions, called *dbmovMFs*, that successfully integrates a directional measure, namely the *cosine* similarity, into a co-clustering framework. The high performances of *dbmovMFs*-based algorithms confirm again the benefits of the vMF modelling assumption when we are dealing with some high dimensional sparse datasets, as it has been already emphasized in (Banerjee et al., 2005b; Zhong and Ghosh, 2005; Gopal and Yang, 2014).

The introduction of column clustering into a mixture of vMF distribution seems to be beneficial from several perspectives: (i) in terms of inference this allows us to reduce substantially the complexity of the model regarding the number of free parameters to be estimated and thereby yielding a parsimonious model, (ii) it makes it possible to exploit the inherent duality between rows and columns which improves the performance, in terms of document clustering, of vMF-based mixture models by a noticeable amount, as demonstrated in our experiments, (iii) it alleviates the high concentration parameter κ issue, by performing an implicitly adaptive dimensionality reduction at each stage, and finally, (iiii) it has the advantage of producing directly interpretable clusters and co-clusters, which may also help to assess the number of clusters when the latter is not known, as emphasized in section 5.7.2.

By setting the estimate of the model parameters under the ML and CML approaches, we proposed six novel co-clustering algorithms corresponding to different variants of the EM algorithm. We derived soft, hard, stochastic and simulated annealing variants. Extensive experiments, conducted on numerous simulated and real-world datasets, provide empirical evidence about the advantages and effectiveness of the proposed algorithms for simultaneous clustering of documents and words, using high dimensional sparse document-term matrices. In particular, we recommend the simulated annealing variants that seem to reach the best performances in almost all situations, especially when clusters are poorly separated. However, if scalability is a requirement, the hard variants $\text{CEM}_{\mathbf{b}}$ and dbSkmeans may turn out to be a wise choice.

The good performances of our algorithms motivate further investigations. For instance, it would be interesting to propose a way to simultaneously use estimates obtained by $\text{SEM}_{\mathbf{b}}$ and $\text{EM}_{\mathbf{b}}$/$\text{CEM}_{\mathbf{b}}$ in an iterative process to overcome therefore the difficulty of managing parameter $\mathbf{w}$. We could also improve *dbmovMFs* by considering a Bayesian formulation that would enable sharing of information between co-clusters (Gopal and Yang, 2014). Future improvements could also involve the development of temporal, incremental and on-line variants of our algorithms. Such variants would be of great interest for applications such as collaborative filtering where available information evolves frequently. Finally, it would be opportune to tackle the problem of the number of co-clusters which remains one of the most important challenge in co-clustering. For instance, an interesting strategy would be to treat the problem of parameter estimation and assessing the number of co-clusters simultaneously (Wyse and Friel, 2012).

Chapter 6

Frequency Sensitive Co-clustering

It is well known that, because of the high dimensionality and sparsity of the data, co-clustering approaches, like one-sided clustering methods, tend to generate highly unbalanced clusters, i.e., clusters of widely varying sizes, in particular when the number of required clusters is large (Dhillon et al., 2002; Cho et al., 2004; Banerjee et al., 2004; Zhang et al., 1999). However, in some situations it may be more beneficial to avoid such very skewed solutions. This chapter describes a novel directional co-clustering approach that addresses the above issue, that is a co-clustering method that makes it possible to take into account the directional properties of some data sets as well as avoid bad solutions with very unbalanced clusters. More precisely, the approach presented here builds on the block mixture of von Mises-Fisher distributions, which is described in the previous chapter, and uses a conscience mechanism that helps to avoid extremely skewed solutions with very small/large or empty clusters. Experimental results on numerous real-world datasets provide empirical evidence about the benefits of the proposed approach.

6.1 Introduction

Although vMF-based clustering approaches alleviate the problem of high dimensionality and sparsity, they tend to produce very unbalanced clusters, i.e., clusters of widely varying sizes, due to high dimensionality and sparsity, especially when the number of required clusters is large (Banerjee et al., 2004).

Obviously, when dealing with real-world datasets, it seems more reasonable to expect unbalanced clusters. However, in many situations it is more beneficial to have balanced clusters, i.e., clusters of comparable sizes. For instance, in text document clustering, balancing, as pointed in (Banerjee et al., 2004), is of great interest when generating topic hierarchies as it prevents the generation of heavily skewed hierarchies, which eases navigation. In

collaborative filtering (CF), partitioning users into comparably sized groups makes it possible to make fair recommendations for all users. In fact, a clustering-based CF system (Salah et al., 2015) makes recommendations to users according to the shared preferences within their groups. Thus, no relevant recommendation can be made for users from small clusters, similarly it is difficult to make personalized recommendations for users from large clusters. For more examples of real-life applications where balanced clustering is useful, the reader can refer to (Banerjee et al., 2004).

Motivated by balanced clustering and directional based clustering, the authors in (Banerjee et al., 2004) proposed a variant of `Skmeans` called Frequency Sensitive `Skmeans` (`FS-Skmeans`) for balanced clustering on high dimensional hyperspheres. The `FS-Skmeans` algorithm is inspired from the Frequency Sensitive Competitive Learning (FSCL) approach (Ahalt et al., 1990) that uses the "conscience" mechanism (DeSieno, 1988). The latter was first used in competitive learning in order to address the problem of highly unbalanced clusters. It consists in penalizing frequently winning "centroids" by making them less likely to win in the future. Based on the conscience mechanism, FSCL was proposed in the context of Vector Quantization to address the problem of under utilisation of parts of codebook. More precisely, FSCL penalizes the representative units by introducing multiplicative terms proportional to some function of the number of their winning. Notice that it only focuses on clustering along one dimension of data matrices, i.e., either rows or columns clustering. However, as pointed in the previous chapter, in the case of high dimensional sparse data, it turns out to be more useful to treat simultaneously both dimensions of data matrices even if one is primarily interested in clustering along one dimension.

However, co-clustering methods, like traditional one-sided clustering, tend to produce highly unbalanced/empty row and column clusters when dealing high dimensional sparse data (Cho et al., 2004), particularly if the number of desired clusters is large. In this chapter, we relay on the block mixture of vMF distributions *dbmovMFs*, presented in the previous chapter, that constitutes a general framework for co-clustering directional data. Then, in order to avoid obtaining clusters of widely varying sizes, we modify *dbmovMFs* in a principled way by introducing a "conscience" mechanism that penalizes the clusters relative to the number of object that have been assigned to them. This yields a very scalable co-clustering algorithm that is guaranteed to increase a spherical k-means like criterion by alternating row and column clusterings at each stage.

To the best of our knowledge, the work that we present, is the first that introduces a conscience mechanism into a co-clustering model in order to avoid highly skewed solution with very small or empty clusters. Experimental results on various real-world datasets provide empirical evidence about the advantages of our approach.

6.2 Frequency Sensitive Co-clustering

Inspired by Frequency Sensitive Competitive Learning and `FS-Skmeans`, we propose hereafter to modify *dbmovMFs*, described in the previous chapter, in an adequate manner by introducing a conscience mechanism, in order to address the problem of obtaining strongly unbalanced clusters and thereby avoiding poor solutions. More precisely, we aim to penalize each row and column cluster by a function of its cardinality, i.e, the number of its elements. To this end, we assume that the concentration parameters are inversely proportional to a function of the number of objects (rows) that has been assigned to their corresponding cluster, i.e, $\kappa_h = \ell / f(z_{.h})$, $\forall h$, where $z_{.h}$ represents the cardinality of row cluster h, ℓ is a constant and f denotes any increasing function. The choice of ℓ and f are discussed in more details below. The complete data log-likelihood of $\mathbf{X}$ is given by:

$$\begin{aligned} L_c(\Theta|\mathbf{X},\mathbf{Z}) &= \sum_h z_{.h} \log \alpha_h + \sum_h z_{.h} \log(c_d(\kappa_h)) + \sum_{i,h} z_{ih} \kappa_h \mu_{hh} \sum_j w_{jh} x_{ij} \\ &= \sum_h z_{.h} \log \alpha_h + \sum_h z_{.h} \log(c_d(\kappa_h)) + \sum_{i,h} z_{ih} \kappa_h (\mu_h^{\mathbf{w}})^\top \mathbf{x}_i. \end{aligned} \tag{6.1}$$

The logarithm of the normalization constant is equal to

$$\log c_d(\kappa_h) = \left(\frac{d}{2} - 1\right) \log \kappa_h - \frac{d}{2} \log 2\pi - \log I_{\frac{d}{2}-1}(\kappa_h). \tag{6.2}$$

We have, $I_r(\kappa_h) = \sum_{m=0}^{\infty} \frac{1}{m!\Gamma(m+r+1)} \left(\frac{\kappa_h}{2}\right)^{2m+r}$ where $\Gamma(x)$ is the Gamma function. Now, let us suppose that each concentration parameter κ_h is sufficiently small (i.e, we assume that clusters are not well separated), it is sufficient to choose $\ell = 1$ and $f(x) \geq 1$ leading to $\kappa_h \in [0,1]$, so that we can ignore terms of the above series with higher power of κ_h. Thus, we can approximate $I_r(\kappa_h)$ with the first two terms of this series as follows

$$\begin{aligned} I_r(\kappa_h) &\approx \frac{\kappa_h^r}{2^r r!} + \frac{\kappa_h^{r+2}}{2^{r+2}(r+1)!} \\ &= \kappa_h^r \left(\frac{1}{2^r r!} + \frac{\kappa_h^2}{2^{r+2}(r+1)!}\right) \\ &= \kappa_h^r \left(\frac{4(r+1) + \kappa_h^2}{2^{r+2}(r+1)!}\right) \end{aligned} \tag{6.3}$$

for high dimensionality, i.e, large order $r = d/2 - 1$, and for small κ_h, we have $4(r+1) + \kappa_h^2 \approx 4(r+1)$ and $I_r(\kappa_h) \approx \kappa_h^r \left(\frac{4(r+1)}{2^{r+2}(r+1)!}\right)$ thereby, $\log I_{\frac{d}{2}-1}(\kappa_h) \approx \left(\frac{d}{2} - 1\right) \log \kappa_h + \log c$ where

$c = \frac{4(r+1)}{2^{r+2}(r+1)!}$. Substituting this approximation in (6.2) gives us,

$$\begin{aligned}\log c_d(\kappa_h) &= \left(\frac{d}{2}-1\right)\log\kappa_h - \frac{d}{2}\log 2\pi - \left(\frac{d}{2}-1\right)\log\kappa_h - \log c \\ &= -\frac{d}{2}\log 2\pi - \log c \end{aligned} \tag{6.4}$$

Substituting (6.4) in (6.1) and assuming that all the mixing proportions are equal, i.e, $\alpha_h = \frac{1}{g}\ \forall h$ leads to

$$\begin{aligned}L_c(\Theta|\mathbf{X},\mathbf{Z}) &= \sum_h z_{.h}\log(1/g) + \sum_h z_{.h}\left(-\frac{d}{2}\log 2\pi - \log c\right) + \sum_{i,h} z_{ih}\kappa_h(\mu_h^{\mathbf{w}})^\top \mathbf{x}_i \\ &= \underbrace{n\left[\log(1/g) - \frac{d}{2}\log 2\pi - \log c\right]}_{constant} + \sum_{i,h} z_{ih}\kappa_h(\mu_h^{\mathbf{w}})^\top \mathbf{x}_i. \end{aligned} \tag{6.5}$$

Making concentration parameter κ_h inversely proportional to a function of the size of cluster h, i.e, $\kappa_h \propto 1/f(z_{.h})$, where f is an increasing function greater than 1, we choose the root square function $f(x) = \sqrt{x}$ and obtain

$$L_c(\Theta|\mathbf{X},\mathbf{Z}) \equiv \sum_{i,h} \frac{1}{\sqrt{z_{.h}}} z_{ih}(\mu_h^{\mathbf{w}})^\top \mathbf{x}_i \tag{6.6}$$

From (6.6) it is obvious that object assignments are done by maximizing a weighted `Skmeans` like criterion, and we observe that the weights penalize larger clusters. Similarly, we can express (6.6) in terms of column assignment as

$$L_c(\Theta|\mathbf{X},\mathbf{Z}) \equiv \sum_{j,h} \frac{1}{\sqrt{w_{.h}}} w_{jh}(\mu_h^{\mathbf{z}})^\top \mathbf{x}^j \tag{6.7}$$

where $w_{.h}$, $\mu_h^{\mathbf{z}}$, i.e, $\|\mu_h^{\mathbf{z}}\| = 1$ denote the cardinality and the centroid of hth column cluster, respectively, and $\mathbf{x}^j$ denotes the jth column. Hence, as for objects clustering, feature clustering consists in maximizing a weighted `Skmeans` like criterion that penalizes larger column clusters by a multiplicative term. It can easily be deduced as follows

$$\begin{aligned}L_c(\Theta|\mathbf{X},\mathbf{Z}) &\equiv \sum_{i,h} \frac{1}{\sqrt{z_{.h}}} z_{ih}(\mu_h^{\mathbf{w}})^\top \mathbf{x}_i = \sum_{j,h} \frac{1}{\sqrt{z_{.h}}} w_{jh}\mu_{hh} \sum_i z_{ih}x_{ij} \\ &= \sum_{j,h} \frac{1}{\sqrt{z_{.h}}} w_{jh} \frac{1}{\sqrt{w_{.h}}} \sum_i z_{ih}x_{ij} = \sum_{j,h} \frac{1}{\sqrt{w_{.h}}} w_{jh}(\mu_h^{\mathbf{z}})^T \mathbf{x}^j \\ &\equiv \sum_{j,h} \frac{1}{\sqrt{w_{.h}}} w_{jh} cos(\mu_h^{\mathbf{z}}, \mathbf{x}^j) \end{aligned} \tag{6.8}$$

Thanks to our co-clustering formulation, penalizing larger row clusters using an adequate choice of concentration parameters, i.e, $\kappa_h = 1/f(z_{.h})$, $\forall h$, with $f(x) = \sqrt{x}$, naturally yields a penalization of larger column clusters with an appropriate multiplicative term, equal to $1/\sqrt{w_{.h}}$. Another choice of function f will give a different criterion. Our Frequency Sensitive co-clustering algorithm, called Frequency Sensitive Diagonal Block Spherical k-means (`FSdbSkm`) is described in more details by Algorithm 8. It can be shown that the computational complexity of `FSdbSkm` is given in $O(it \cdot nz)$, where it and nz denote respectively the number of iterations and non-zero entries. Thereby `FSdbSkm` is very efficient, therefore suitable for high dimensional sparse data.

Algorithm 8: `FSdbSkm`.

Input: $\mathbf{X}$ ($\mathbf{x}_i \in \mathbb{S}^{d-1}$), g the number of co-clusters.
Output: $\mathbf{Z}$ and $\mathbf{W}$,
Steps:
Initialization: $\Theta \leftarrow \Theta^{(0)}$;
repeat
1. Assignment of objects:
for $i = 1$ **to** n **do**
$z_i \leftarrow h$ where $h = \arg\max_{h'}(z_{.h'})^{-0.5} cos(\mu^{\mathbf{w}}_{h'}, \mathbf{x}_i)$
end for
2. Assignement of Features:
for $j = 1$ **to** d **do**
$w_j \leftarrow h$ where $h = \arg\max_{h'}(w_{.h'})^{-0.5} cos(\mu^{\mathbf{z}}_{h'}, \mathbf{x}^j)$
end for
3. Computation of μ_{hh}'s maximizing (6.6):
for $h = 1$ **to** g **do**
$\mu_{hh} \leftarrow \frac{1}{\sqrt{w_{.h}}}$
end for
until convergence

Herein, we prove that `FSdbSkm` monotonically increases criterion (6.6), and terminates in a finite number of iterations at a local optimal solution.

At iteration (t), $L_c(\Theta|\mathbf{X},\mathbf{Z})$ noted L_c, we have

$$\begin{aligned}
L_c^{(t)} &= \sum_{i,h}(z_{.h}^{(t)})^{-0.5} z_{ih}^{(t)} ((\mu_h^{\mathbf{w}})^{(t)})^{\top}\mathbf{x}_i \\
&= \sum_{i,h}(z_{.h}^{(t)})^{-0.5} z_{ih}^{(t)} (\mu_{hh})^{(t)} \sum_j w_{jh}^{(t)} x_{ij} \\
&= \sum_{i,h}(z_{.h}^{(t)})^{-0.5} z_{ih}^{(t)} (w_{.h}^{(t)})^{-0.5} \sum_j w_{jh}^{(t)} x_{ij} \\
&= \sum_{i,h}(w_{.h}^{(t)})^{-0.5} z_{ih}^{(t)} (z_{.h}^{(t)})^{-0.5} \sum_j w_{jh}^{(t)} x_{ij} \\
&= \sum_{i,h}(w_{.h}^{(t)})^{-0.5} z_{ih}^{(t)} (\mu_{hh}^{\mathbf{z}})^{(t)} \sum_j w_{jh}^{(t)} x_{ij} \\
&\leq \sum_{i,h}(w_{.h}^{(t)})^{-0.5} z_{ih}^{(t+1)} (\mu_{hh}^{\mathbf{z}})^{(t)} \sum_j w_{jh}^{(t+1)} x_{ij} \qquad (6.9)
\end{aligned}$$

the latter inequality follows from Steps (1) and (2) of `FSdbSkm`. Further, as we have $(\mu_{hh}^{\mathbf{z}})^{(t+1)} = \arg\max_{\mu_{hh}^{\mathbf{z}}} L_c^{(t)}$, subject to the constraint $\|\mu_h^{\mathbf{z}}\| = 1$ therefore $(\mu_{hh}^{\mathbf{z}})^{(t+1)} = (z_{.h}^{(t+1)})^{-0.5}$, thereby we obtain,

$$\begin{aligned}
L_c^{(t)} &\leq \sum_{i,h}(z_{.h}^{(t)})^{-0.5} z_{ih}^{(t+1)} (\mu_{hh}^{\mathbf{z}})^{(t+1)} \sum_j w_{jh}^{(t)} x_{ij} \\
&\leq \sum_{i,h}(w_{.h}^{(t)})^{-0.5} z_{ih}^{(t+1)} (z_{.h}^{(t+1)})^{-0.5} \sum_j w_{jh}^{(t)} x_{ij} \\
&\leq \sum_{i,h}(z_{.h}^{(t+1)})^{-0.5} z_{ih}^{(t+1)} (\mu_{hh})^{(t+1)} \sum_j w_{jh}^{(t+1)} x_{ij} \\
&\leq \sum_{i,h}(z_{.h}^{(t+1)})^{-0.5} z_{ih}^{(t+1)} ((\mu_h^{\mathbf{w}})^{(t+1)})^{\top}\mathbf{x}_i = L_c^{(t+1)}
\end{aligned}$$

the latter inequality follows from step (3) of `FSdbSkm`. The convergence of `FSdbSkm` to a local optimum of criterion (6.6) follows, since the number of distinct row and column partitions is finite.

6.3 Experimental results

To show the benefits of our approach, we conduct extensive experiments on numerous real-world text datasets, in which we benchmark our algorithm `FSdbSkm` against several strong baselines: the `soft-movMF` proposed in (Banerjee et al., 2005b), `da-movMF` is the deterministic annealing version of movMF proposed in (Zhong and Ghosh, 2005) and

`Skmeans` (Dhillon and Modha, 2001) which is also a simplified version of `soft-movMF`, where $\kappa_h = \kappa \to \infty$, $\alpha_h = \alpha$, for all h.

For text documents clustering, it has been empirically shown on numerous real-world datasets, that the aforementioned baselines perform better than several existing clustering and co-clustering algorithms including: K-means with the euclidean distance, generative model using Gaussian, Bernoulli and Multinomial distributions, spectral co-clustering, and Latent Dirichlet Allocation (LDA) (see (Zhong and Ghosh, 2005; Gopal and Yang, 2014)). Therefore, we do not include these approaches in our comparisons.

6.3.1 Datasets

As a practical example we select the domain of text mining and we concentrate on the challenging task of text document clustering using high dimensional sparse document-term matrices. Hence, we retain six popular benchmark text datasets, CLASSIC4[1], LA12, the 20-newsgroups data NG20, SPORTS used in (Zhong and Ghosh, 2005), NG2 and NG17-19 consisting respectively of two and three overlapping classes[2] of NG20. All these datasets are carefully selected to represent various particular challenging situations in clustering: different degrees of cluster balance, different numbers of clusters, different sizes, different degrees of cluster overlap. The characteristics of these datasets are summarized in Table 6.1. Note that, in the above datasets, the true document cluster labels are available. The

Table 6.1 Description of Datasets.

Datasets	Characteristics						
	n	d	g	Sparsity (%)	Balance	RME	SDCS
NG2	500	2000	2	96.85	1.00	1.00	0.00
CLASSIC4	7094	5896	4	99.41	0.323	0.582	972.18
NG20	19949	43586	20	99.82	0.991	0.993	2.28
SPORTS	8580	14870	7	99.14	0.036	0.099	1253.01
LA12	6279	31472	6	99.52	0.281	0.498	526.49
NG17-19	2998	23233	3	99.51	0.998	0.999	1.15

word cluster labels, however, are not known, this is the reason why we mainly concentrate our experiments on document clustering. Nevertheless, under our formulation row clusters induce column clusters and vice versa thereby the quality of row clustering is informative about the quality of column clustering.

[1] http://www.dataminingresearch.com/

[2] *talk.politics.misc*, *talk.politics.guns* and *talk.politics.mideast*

6.3.2 Evaluation measures

Evaluating clustering results is not a trivial task, however, when the true category labels are known, a commonly used approach to validate clustering results consists in comparing the estimated partition with the true one. To this end, several measures have been proposed to assess the "similarity" between the estimated clustering and the true clustering. In our experiments we retain two widely used measures to assess the quality of clustering, namely the Normalized Mutual Information NMI (Strehl and Ghosh, 2003) and the Adjusted Rand Index ARI (Hubert and Arabie, 1985).

To assess cluster balancing we use three measures: the Standard Deviation in Cluster Sizes (SDCS) (Banerjee et al., 2004), the ratio of minimum cluster size to the expected cluster size RME (Banerjee et al., 2004) and the Balance coefficient, i.e, the ratio of the minimum cluster size to the maximum cluster size. The latter makes it possible to know whether very small/large clusters are obtained.

6.3.3 Empirical results and discussion

In all our experiments we use the TF-IDF normalized data representation, in particular we use the TF-IDF weighting scheme proposed in Scikit-learn (Pedregosa et al., 2011). For each dataset we set g to the real number of clusters, and in order to make consistent comparisons, all algorithms are initialized using the same random row partition. For our algorithm we further randomly initialize the column partition. Each algorithm is run until there is no significant increase of the optimized criterion.

Notice that, in the case of document-term matrices, during the first iterations, document clusters are mixed. Hence each word is involved to describe several document clusters, thus imposing hard word assignments at the beginning, as in Algorithm 8, could lead to a poor local solution. To avoid this difficulty, we perform a stochastic column assignment during the first 70% iterations of `FSdbSkm` as follows: for each column j select a cluster w_j according to a Multinomial distribution $\mathcal{M}(\tilde{w}_{j1},\ldots,\tilde{w}_{jg})$ where $\tilde{w}_{jh} \propto \frac{1}{\sqrt{w_{.h}}} cos(\boldsymbol{\mu}_h^{\mathbf{z}}, \mathbf{x}^j)$.

Figure 6.1 illustrates the structure revealed by `Skmeans` and our algorithm *FSdbSkeamns* on the NG2 dataset. As we can see from this Figure, *FSdbSkeamns* not only reaches better performances in terms of NMI, ARI and Balance but also reveals a more interesting structure, i.e, block diagonal. Tables 6.2 and 6.3, summarize the results of the different methods in terms of NMI and ARI, over all datasets. All results are averaged over thirty different starting points, obtained using the initialization strategy described above. Between brackets, we report the result corresponding to the trial with the highest criterion. As these tables show clearly, our algorithm `FSdbSkm` provides substantially better results than the other vMF-based

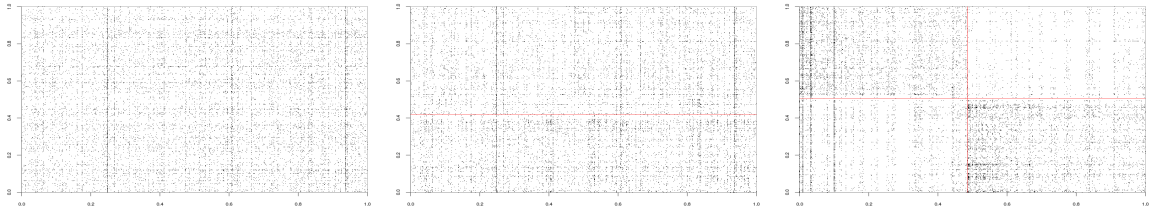


Fig. 6.1 NG2 dataset (Balance = 1). Left: original, middle: reorganized according to `Skmeans` result(NMI = 0.57, ARI = 0.67, Balance = 0.73), right: Reorganized according to `FSdbSkm` results(NMI = 0.74, ARI = 0.83, Balance = 0.98).

Table 6.2 Comparison of Average NMI, ARI on the NG20, SPORTS, LA12 datasets.

	NG20		SPORTS		LA12	
	NMI	ARI	NMI	ARI	NMI	ARI
`soft-movMF`	0.44±0.01 (0.46)	0.12±0.01 (0.13)	0.43±0.03 (0.43)	0.12±0.05 (0.10)	0.40±0.04 (0.46)	0.21 ±0.07 (0.29)
`da-movMF`	0.57±0.01 (0.58)	0.40±0.02 (0.40)	0.62±0.02 (0.63)	0.40±0.03 (0.41)	0.53±0.01 (0.55)	0.49±0.01 (0.52)
`Skmeans`	0.54±0.01 (0.55)	0.37±0.02 (0.38)	0.61±0.04 (0.63)	0.40±0.05 (0.44)	0.53±0.05 (0.56)	0.48±0.05 (0.50)
`FSdbSkm`	0.61±0.01 (**0.62**)	0.50±0.01 (**0.51**)	0.62±0.03 (**0.67**)	0.52±0.05 (**0.62**)	0.56±0.02 (**0.58**)	0.53±0.02 (**0.55**)

Table 6.3 Comparison of Average NMI, ARI on the NG17-19, CLASSIC4 and NG2 datasets.

	NG17-19		CLASSIC4		NG2	
	NMI	ARI	NMI	ARI	NMI	ARI
`soft-movMF`	0.27±0.02 (0.29)	0.14 ±0.01 (0.14)	0.48±0.01 (0.49)	0.29±0.02 (0.30)	0.14±0.06 (0.26)	0.06±0.06 (0.16)
`da-movMF`	0.37±0.01 (0.37)	0.31±0.01 (0.30)	0.59±0.003 (0.59)	0.47±0.002 (0.47)	0.55±0.06 (0.57)	0.64±0.08 (0.66)
`Skmeans`	0.42±0.04 (0.41)	0.36±0.06 (0.35)	0.59±0.02 (0.59)	0.47±0.01 (0.48)	0.51 ±0.06 (0.57)	0.58 ±0.08 (0.67)
`FSdbSkm`	0.53±0.04 (**0.57**)	0.56±0.04 (**0.59**)	0.70±0.01 (**0.72**)	0.67±0.01 (**0.70**)	0.71±0.02 (**0.74**)	0.81±0.01 (**0.83**)

clustering algorithms. In particular when the data exhibits overlapping clusters as this is the case for NG2, CLASSIC4, NG20 and NG17-19. In fact, `FSdbSkm` is derived from a vMF-based model by assuming that clusters are not well separated, i.e., by assuming small concentration parameters. More interestingly, even when clusters are extremely unbalanced in nature, as in the SPORTS dataset, `FSdbSkm` provides high performances in terms of NMI and ARI. In fact, `FSdbSkm` tries to discover balanced clusters (see Table 6.4) without destroying the overall structure of data. Hence in such situation, the conscience mechanism of `FSdbSkm` has a regularizing effect that makes it possible to avoid poor solutions with empty or very large clusters.

Table 6.4 Comparison BALANCE, RME and SDCS on the NG20, SPORTS, LA12 datasets.

	NG20			SPORTS			LA12		
	BALANCE	RME	SDCS	BALANCE	RME	SDCS	BALANCE	RME	SDCS
`soft-movMF`	0.018	0.125	1534.42	0.074	0.293	1618.73	0.069	0.159	751.63
`da-movMF`	0.120	0.256	465.26	0.252	0.517	644.86	0.603	0.782	232.82
`Skmeans`	0.175	0.332	401.42	0.271	0.476	545.37	0.372	0.594	361.74
`FSdbSkm`	0.482	0.627	147.79	0.446	0.715	383.80	0.534	0.679	223.07

Table 6.5 Comparison BALANCE, RME and SDCS on NG17-19, CLASSIC4, NG2.

	NG17-19			CLASSIC4			NG2		
	BALANCE	RME	SDCS	BALANCE	RME	SDCS	BALANCE	RME	SDCS
`soft-movMF`	0.079	0.188	1204.23	0.251	0.547	1398.46	0.275	0.432	200.82
`da-movMF`	0.272	0.508	752.79	0.457	0.635	591.90	0.825	0.904	33.94
`Skmeans`	0.171	0.347	902.26	0.467	0.635	582.49	0.735	0.834	48.08
`FSdbSkm`	0.710	0.838	172.40	0.407	0.581	615.34	0.976	0.988	4.24

Moreover, as Tables 6.2, 6.3, 6.4 and 6.5 show, when clusters are naturally balanced as in NG2, NG17-19 and NG20 `FSdbSkm` yields substantially better performances in terms of all measures. Finally, Figure 6.2 shows that `FSdbSkm` offers a good trade off between cluster-balancing and quality of clustering.

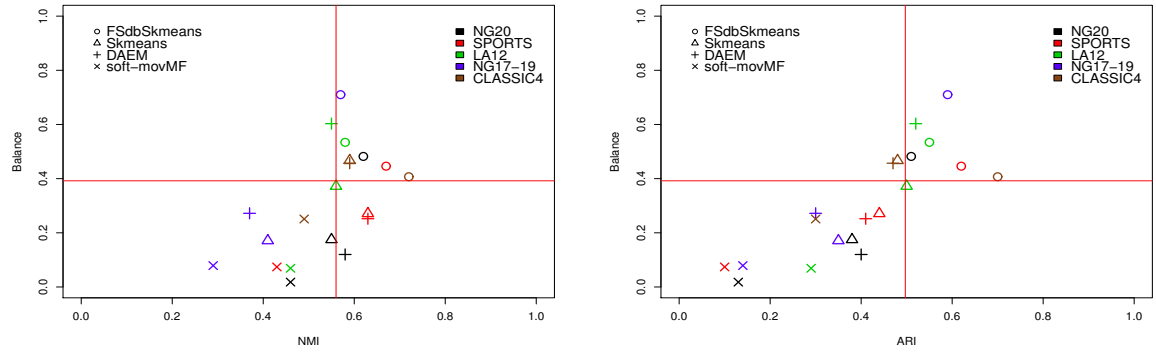


Fig. 6.2 Left Balance vs NMI, Righ Balance vs ARI. The circles denote the scores recorded by `FSdbSkm`.

So far, we focused on document clustering but `FSdbSkm` offers word clusters as well. Each word cluster describes a single document cluster thereby easing semantic interpretations. Table 6.6 provides a typical example, where the four word clusters, discovered by `FSdbSkm` on CLASSIC4, are represented by their top 15 terms. The obtained word clusters are meaningful and help to make sense of the corresponding document clusters.

Table 6.6 CLASSIC4 dataset: the term clusters discovered by FSdbSkm represented by their top 15 terms sorted according to *cosine* similarity. The clusters C1, C2, C3 and C4 corresponds respectively to CACM, CRANFIELD, CISI and MEDLINE.

C1	C2	C3	C4
algorithm	flow	librari	patient
program	boundari	inform	cell
system	layer	scienc	case
comput	pressur	index	rat
languag	equat	retriev	growth
method	solut	research	treatment
function	heat	book	blood
gener	wing	servic	acid
matrix	number	scientif	hormon
data	bodi	docum	diseas
time	theori	search	children
problem	shock	develop	increas
structur	mach	journal	tissu
integr	effect	studi	normal
set	plate	user	group

6.4 Conclusion and perspectives

In this chapter we proposed FSdbSkm, a novel co-clustering algorithm tailored for high dimensional sparse data. Unlike existing co-clustering algorithms, FSdbSkm is derived from a mixture of vMF distributions. It is more suitable than existing co-clustering methods for directional data distributed on a unit hypersphere, a typical situation in text mining where data are often normalized in order to remove the bias induced by the length of documents. Moreover, FSdbSkm, like Frequency Sensitive Competitive Learning, uses a conscience mechanism that penalizes larger clusters in order to avoid highly skewed solutions with very large/small clusters. Empirical results obtained on numerous real-world datasets show that FSdbSkm performs better than other strong baselines, which confirms the advantages of our approach. Moreover, even if balancing is not a requirement, the conscience mechanism of FSdbSkm has an interesting regularizing effect that makes it possible to avoid poor solutions.

Although the FSdbSkm algorithm is inspired by FSCL, it does not have the incremental behaviour of FSCL. Hence, possible future work could include further research to develop incremental and on-line variants of FSdbSkm that would be of great interest for applications, such as Collaborative Filtering (see chapter 2), where available data change frequently.

Conclusion and Perspectives

Through this thesis, we proposed and studied a novel class of clustering algorithms tailored for directional data distributed on the surface of a unit-hypersphere—L_2 normalized data. The algorithms studied in this thesis are derived from rigorous probabilistic models that build on the von Mises-Fisher (vMF) distribution from directional statistics (Mardia and Jupp, 2000). Our focus on the above modeling assumption is largely motivated by the substantial empirical evidence that some high dimensional sparse data sets, such as text and collaborative filtering (CF) data, posses intrinsic directional properties that match well with the modeling assumption of the vMF distribution. In fact, several previous studies in information retrieval have empirically demonstrated that directional measures such as the *cosine* similarity are superior to several other measures, such *Euclidean* distortions, for clustering text documents or assessing similarities between users (resp. items) in CF.

The main contributions and key results of this thesis can be summarized as follows:

- **Chapter 2** exposes a scalable incremental variant of the spherical k-means algorithm suitable for CF, and which is able to handle effectively the frequent changes in the data such as submission of new ratings, update of existing ratings, appearance of new users and items. Extensive experiments conducted on popular benchmark data sets demonstrate the effectiveness and scalability of our method denoted as `EICF`. It offers a better quality of recommendations than other incremental CF systems while requiring less computation time, in both the static situation, where available data are kept unchanged, and dynamic situation, where new information are incorporated incrementally. In addition, to evaluate the performances of our approach under different circumstances, we also emphasized and illustrated why prediction metrics such as mean average error (MAE) and root mean square error (RMSE), widely used in the context of recommender systems, are inadequate to evaluate CF models. In fact, in many situation low MAE and RMSE do not necessarily equate to best user satisfaction.

- **Chapter 3** presents a novel social collaborative filtering model which is based on the vMF assumption. The proposed model (*Social-movMFs*) is motivated by the benefits

of both modeling CF data as directional data distributed on a unit-hypersphere and incorporating information from online social-networks so as to alleviate the sparsity related issues in CF. The *Social-movMFs* model is based on the natural assumption that socially connected users tend to share similar tastes. Indeed, in the physical world people usually turn to their friends, as they are familiar with their tastes, to ask for suggestions before choosing a movie, restaurant, book, etc. In this spirit, *Social-movMFs* simultaneously seeks for groups of users who tend to express similar preferences and brings the distributions over clusters (groups of users) of socially connected users closer to each other so as to capture the relations between users. An experimental study on various real-wold data sets, that include both the user-item preference matrix and the user-user social network, demonstrates the advantages of both the vMF modeling assumption and using social network information in order to alleviate the sparsity problem and improve CF performances, especially when there are many *cold start* users, i.e., who expressed very few ratings or even none at all.

- **Chapter 5** describes a novel co-clustering model that is well suited for co-clustering L_2 normalized data. While existing co-clustering approaches are based on popular assumption such Gaussian, Multinomial or Bernoulli that are inadequate for L_2 normalized data, the proposed model *dbmovMFs* builds on the vMF assumption, that arises naturally for data lying on the surface of a unit-hypersphere, and successfully integrates a directional measure, namely the *cosine* similarity into a co-clustering framework. Hence, *dbmovMFs* leverages the advantages of both the co-clustering and the modeling assumption of the vMF distribution when dealing with high dimensional sparse data sets. Co-clustering by using the vMF distribution is not straightforward. Most existing co-clustering approaches are based on the principle of the latent block model (LBM) (Govaert and Nadif, 2003, 2013) which uses univariate probability distributions; the vMF distribution, however, is a probability distributions of L_2 normalized vectors. In order to overcome the aforementioned difficulty, we moved the co-clustering into the centroid parameter of the vMF distribution by making some appropriate assumptions on its structure. In other words, by contrast to LBM-based co-clustering where an entire model is designed for co-clustering, in our case we are using a one-sided clustering model and put the co-clustering aim in the model parameters. The *dbmovMFs* can give rise to several new co-clustering algorithms, in this thesis we derived and studied six algorithms, including hard, soft and stochastic variants. Empirical results on numerous simulated and real-world data sets provide strong support for the effectiveness of our approach and the benefits of the vMF modelling assumption in the case of high

dimensional sparse text data. The introduction of column clustering into a mixture of vMF distribution turns out to be very useful from several perspectives:

  - In terms of inference it reduces substantially the complexity of the model regarding the number of free parameters to be estimated, which yields a parsimonious model and very scalable algorithms.
  - It makes it possible to exploit the inherent duality between rows and columns which improves the performance, in terms of document clustering, of vMF-based mixture models noticeably, as illustrated in our experiments.
  - It performs an implicitly adaptive dimensionality reduction at each stage, which makes possible to alleviate the high concentration parameter κ issue because of high dimensionality.
  - It provides us with semantically coherent and directly interpretable results where each document cluster is described by a term cluster, which may also help to assess the number of clusters when the latter is not known, as emphasized in section 5.7.2.
  - More interestingly, when assessing the number of clusters, the word cluster may be exploited so as to tune the number of cluster returned by well-established information criteria, AIC and AIC3, for model selection.

- **Chapter 6** presents a novel co-clustering approach that uses a *conscience* mechanism in order to escape extremely skewed solutions, with very unbalanced or empty clusters, because of high dimensionality and sparsity, in particular when the number of desired clusters is large. The proposed algorithm `FSdbSkm` is derived from the *dbmovMFs* model, it is therefore well suited for L_2 normalized data. Empirical results obtained on several real-world data sets show that `FSdbSkm` succeeds in avoiding bad local solutions with very unbalanced clusters sizes and performs substantially better than other strong baselines. More interestingly, even when the balancing is not a requirement, and the clusters are by nature very unbalanced, the conscience mechanism of `FSdbSkm` is still of great interest. In such situation, instead of destroying the natural cluster structure of the data so as to provide balanced clusters, the conscience mechanism acts as a regularizer that prevents from poor locally optimal solutions. This behaviours suggests that `FSdbSkm` seeks for balanced clusters but not to the detriment of the inherent structure of data.

Overall, the results from our thesis highlight the importance of accounting for the intrinsic directional properties of some high dimensional sparse data sets such as those encountered in

text mining and collaborative filtering. Relying on the directional assumption allows us to develop not only very effective but also very scalable algorithms, which are well suited for handling high dimensionality and sparsity.

The studies presented in this thesis motivate further investigations that may include

- Incorporating context-aware information (Tang et al., 2013; Bobadilla et al., 2013) such as time into `EICF` and `Social-movMF` so as to handle the sparsity problem even more effectively and thereby improve recommendations. Because the sparsity in CF is caused by missing (or unobserved) ratings, and both `EICF` and `Social-movMF` are based on mixture models, another possibility to better address the sparsity problem is to rely to well-established statistical theory of missing data analysis (Little and Rubin, 2002).

- Extend *dbmovMFs* to collaborative filtering in order to treat the sets of users and items simultaneously. This would be useful for several reasons: the dimensionality reduction property of *dbmovMFs* will help to alleviate the sparsity and develop very scalable CF systems, it will provide us with more interpretable results by grouping users and items into meaningful clusters, i.e., each group of users may be described by a cluster of items they prefer, and vice versa. Moreover the social component of *Social-movMFs* can be easily incorporated into *dbmovMFs* in order to account for social interactions among users in the latter.

- It is interesting to consider a Bayesian formulation of *dbmovMFs* (Gopal and Yang, 2014) that would enable sharing of information between co-clusters, consider a soft column partition $\mathbf{w}$ and, more interestingly, allow the number of co-clusters to be determined automatically as part of the learning process.

- The von Mises-Fisher distribution is one of the simplest distributions for directional data lying on the surface of a unit-hypersphere, and other distributions from directional statistics deserve to be considered. For instance the Kent distribution (Mardia and Jupp, 2000), which is the analogous on a unit-hypersphere of the multivariate Gaussian distribution with an unconstrained covariance matrix, is more sophisticated than the vMF distribution and can model clusters of more flexible shapes. Nevertheless, it is substantially more difficult to work with a high dimensional Kent distribution regarding parameter estimation.

List of Publications

Published articles

1. Aghiles Salah, Nicoleta Rogovschi, and Mohamed Nadif. Stochastic Co-clustering for Document-Term Data. *(To appear) in Proceedings of the SIAM International Conference on Data Mining (SDM'16). 2016.*

2. Aghiles Salah, Nicoleta Rogovschi, and Mohamed Nadif. Model-based Co-clustering for High Dimensional Sparse Data. *In Proceedings of the 19th International Conference on Artificial Intelligence and Statistics (AISTAT'16), pp. 866-874. 2016.*

3. Aghiles Salah, Nicoleta Rogovschi, and Mohamed Nadif. Classification Croisée via un Modèle de Mélange de von Mises-Fisher. *Société Francophone de Classification (SFC). 2016.*

4. Aghiles Salah, Nicoleta Rogovschi, and Mohamed Nadif. A dynamic collaborative filtering system via a weighted clustering approach. *Neurocomputing 175 (2016): 206-215.*

5. Aghiles Salah, Nicoleta Rogovschi, and Mohamed Nadif. An Efficient Incremental Collaborative Filtering System. *In International Conference on Neural Information Processing, pp. 375-383. Springer International Publishing, 2015.*

Submitted articles

1. Aghiles Salah, Nicoleta Rogovschi, and Mohamed Nadif. Co-clustering via a mixture of von Mises-Fisher Distributions. *under review in Journal of Machine Learning Research. 2016.*

2. Aghiles Salah, and Mohamed Nadif. Directional Model-based Social Collaborative Filtering to Alleviate the Sparsity related Issues. *Under review.*

References

M. Abramowitz and I. A. Stegun. *Handbook of mathematical functions: with formulas, graphs, and mathematical tables*. Number 55. Courier Corporation, 1964.

C. C. Aggarwal and C. K. Reddy. *Data clustering: algorithms and applications*. CRC Press, 2013.

S. C. Ahalt, A. K. Krishnamurthy, P. Chen, and D. E. Melton. Competitive learning algorithms for vector quantization. *Neural networks*, 3(3):277–290, 1990.

H. J. Ahn. A new similarity measure for collaborative filtering to alleviate the new user cold-starting problem. *Inf. Sci.*, 178(1):37–51, 2008.

M. Ailem, F. Role, and M. Nadif. Co-clustering document-term matrices by direct maximization of graph modularity. In *Proceedings of the 24th ACM International on Conference on Information and Knowledge Management*, pages 1807–1810. ACM, 2015.

H. Akaike. A new look at the statistical model identification. *IEEE transactions on automatic control*, 19(6):716–723, 1974.

H. Akaike. Information theory and an extension of the maximum likelihood principle. In *Breakthroughs in statistics*, pages 610–624. Springer, 1992.

X. Amatriain, P. Castells, A. de Vries, and C. Posse. Workshop on recommendation utility evaluation: beyond rmse–rue 2012. In *ACM RecSys*, pages 351–352, 2012.

M. Ankerst, M. M. Breunig, H.-P. Kriegel, and J. Sander. Optics: ordering points to identify the clustering structure. In *ACM Sigmod Record*, volume 28, pages 49–60. ACM, 1999.

A. Banerjee, I. Dhillon, J. Ghosh, S. Merugu, and D. S. Modha. A generalized maximum entropy approach to bregman co-clustering and matrix approximation. In *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 509–514, 2004.

A. Banerjee, I. S. Dhillon, J. Ghosh, and S. Sra. Clustering on the unit hypersphere using von mises-fisher distributions. *J. Mach. Learn. Res.*, 6:1345–1382, 2005a.

A. Banerjee, S. Merugu, I. S. Dhillon, and J. Ghosh. Clustering with bregman divergences. *The Journal of Machine Learning Research*, 6:1705–1749, 2005b.

A. Banerjee, I. S. Dhillon, J. Ghosh, S. Merugu, and D. S. Modha. A generalized maximum entropy approach to bregman co-clustering and matrix approximations. *Journal of Machine Learning Research (JMLR)*, 8, 2007.

J. D. Banfield and A. E. Raftery. Model-based gaussian and non-gaussian clustering. *Biometrics*, pages 803–821, 1993.

A. B. Barragáns-Martínez, E. Costa-Montenegro, J. C. Burguillo, M. Rey-López, F. A. Mikic-Fonte, and A. Peleteiro. A hybrid content-based and item-based collaborative filtering approach to recommend tv programs enhanced with singular value decomposition. *Inf. Sci.*, pages 4290–4311, 2010.

M. Belkin, P. Niyogi, and V. Sindhwani. Manifold regularization: A geometric framework for learning from labeled and unlabeled examples. *The Journal of Machine Learning Research*, 7:2399–2434, 2006.

C. Biernacki, G. Celeux, and G. Govaert. Assessing a mixture model for clustering with the integrated completed likelihood. *IEEE transactions on pattern analysis and machine intelligence*, 22(7):719–725, 2000.

D. M. Blei, A. Y. Ng, and M. I. Jordan. Latent dirichlet allocation. *the Journal of machine Learning research*, 3:993–1022, 2003.

J. Bobadilla, F. Ortega, A. Hernando, and A. Gutiérrez. Recommender systems survey. *Knowledge-Based Systems*, pages 109 – 132, 2013.

H.-H. Bock. Two-way clustering for contingency tables: Maximizing a dependence measure. In *Between data science and applied data analysis*, pages 143–154. Springer, 2003.

H.-H. Bock. Clustering methods: A history of k-means algorithms. In *Selected contributions in data analysis and classification*, pages 161–172. Springer, 2007.

W. P. Bouberima, M. Nadif, and Y. K. Bencheikh. Assessing the number of clusters from a mixture of von mises-fisher. In *Proceedings of the World Congress on Engineering*, volume 3, 2010.

H. Bozdogan. Mixture-model cluster analysis using model selection criteria and a new informational measure of complexity. In *Proceedings of the first US/Japan conference on the frontiers of statistical modeling: An informational approach*, pages 69–113. Springer, 1994.

J. S. Breese, D. Heckerman, and C. Kadie. Empirical analysis of predictive algorithms for collaborative filtering. In *Proceedings of the Fourteenth conference on Uncertainty in artificial intelligence*, pages 43–52, 1998.

R. Burke. Hybrid recommender systems: Survey and experiments. *User Modeling and User-Adapted Interaction*, 12(4):331–370, Nov. 2002.

D. Cai, Q. Mei, J. Han, and C. Zhai. Modeling hidden topics on document manifold. In *Proceedings of the 17th ACM conference on Information and knowledge management*, pages 911–920. ACM, 2008.

G. Celeux and J. Diebolt. The SEM algorithm: a probabilistic teacher algorithm derived from the EM algorithm for the mixture problem. *Computational statistics quarterly*, 2(1): 73–82, 1985.

G. Celeux and J. Diebolt. A stochastic approximation type em algorithm for the mixture problem. *Stochastics: An International Journal of Probability and Stochastic Processes*, 41(1-2):119–134, 1992.

G. Celeux and G. Govaert. A classification em algorithm for clustering and two stochastic versions. *Computational statistics & Data analysis*, 14(3):315–332, 1992.

G. Celeux and G. Govaert. Gaussian parsimonious clustering models. *Pattern recognition*, 28(5):781–793, 1995.

P. K. Chan, M. D. Schlag, and J. Y. Zien. Spectral k-way ratio-cut partitioning and clustering. *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, 13(9): 1088–1096, 1994.

A. J. Chaney, D. M. Blei, and T. Eliassi-Rad. A probabilistic model for using social networks in personalized item recommendation. In *ACM RecSys*, pages 43–50, 2015.

M. Charrad, Y. Lechevallier, M. ben Ahmed, and G. Saporta. Block clustering for web pages categorization. In *Intelligent Data Engineering and Automated Learning-IDEAL 2009*, pages 260–267. 2009.

Y. Cheng and G. M. Church. Biclustering of expression data. In *Ismb*, volume 8, pages 93–103, 2000.

H. Cho, I. S. Dhillon, Y. Guan, and S. Sra. Minimum sum-squared residue co-clustering of gene expression data. In *SDM*, volume 3, page 3. SIAM, 2004.

M. Corneli, P. Latouche, and F. Rossi. Exact icl maximization in a non-stationary time extension of the latent block model for dynamic networks. In *European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning (ESANN)*, pages 225–230, 2015.

P. Cremonesi, Y. Koren, and R. Turrin. Performance of recommender algorithms on top-n recommendation tasks. In *ACM RecSys*, pages 39–46, 2010.

J.-J. Daudin, F. Picard, and S. Robin. A mixture model for random graphs. *Statistics and computing*, 18(2):173–183, 2008.

J. Davidson, B. Liebald, J. Liu, P. Nandy, T. Van Vleet, U. Gargi, S. Gupta, Y. He, M. Lambert, B. Livingston, and D. Sampath. The youtube video recommendation system. In *RecSys*, pages 293–296, 2010.

J. Delporte, A. Karatzoglou, T. Matuszczyk, and S. Canu. Socially enabled preference learning from implicit feedback data. In *ECML PKDD'13*, pages 145–160. Springer, 2013.

A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the em algorithm. *Journal of the royal statistical society. Series B (methodological)*, pages 1–38, 1977.

M. Deodhar and J. Ghosh. Scoal: a framework for simultaneous co-clustering and learning from complex data. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 4(3): 11, 2010.

D. DeSieno. Adding a conscience to competitive learning. In *IEEE International Conference on Neural Networks*, pages 117–124, 1988.

I. S. Dhillon. Co-clustering documents and words using bipartite spectral graph partitioning. In *Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 269–274. ACM, 2001.

I. S. Dhillon and D. S. Modha. Concept decompositions for large sparse text data using clustering. *Mach. Learn.*, 42(1-2):143–175, 2001.

I. S. Dhillon and S. Sra. Modeling data using directional distributions. Technical report, Department of Computer Sciences, The University of Texas at Austin, 2003.

I. S. Dhillon, Y. Guan, and J. Kogan. Iterative clustering of high dimensional text data augmented by local search. In *Data Mining, 2002. ICDM 2003. Proceedings. 2002 IEEE International Conference on*, pages 131–138. IEEE, 2002.

I. S. Dhillon, S. Mallela, and D. S. Modha. Information-theoretic co-clustering. In *Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 89–98. ACM, 2003.

E. Diday. Une nouvelle méthode en classification automatique et reconnaissance des formes la méthode des nuées dynamiques. *Revue de statistique appliquée*, 19(2):19–33, 1971.

C. Ding, T. Li, W. Peng, and H. Park. Orthogonal nonnegative matrix t-factorizations for clustering. In *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 126–135, 2006.

C. H. Ding, X. He, H. Zha, M. Gu, and H. D. Simon. A min-max cut algorithm for graph partitioning and data clustering. In *Data Mining, 2001. ICDM 2001, Proceedings IEEE International Conference on*, pages 107–114. IEEE, 2001.

M. Ester, H.-P. Kriegel, J. Sander, and X. Xu. A density-based algorithm for discovering clusters in large spatial databases with noise. In *Kdd*, volume 96, pages 226–231, 1996.

B. S. Everitt, S. Landau, M. Leese, and D. Stahl. *Cluster Analysis*. John Wiley & Sons, Ltd, 2011.

H. Gao, J. Tang, X. Hu, and H. Liu. Content-aware point of interest recommendation on location-based social networks. In *Proceedings of the 29th AAAI Conference on Artificial Intelligence*, 2015.

T. George and S. Merugu. A scalable collaborative filtering framework based on co-clustering. In *Fifth IEEE International Conference on Data Mining*, pages 625–628, 2005.

S. Gopal and Y. Yang. Von mises-fisher clustering models. In *Proceedings of the 31st International Conference on Machine Learning (ICML-14)*, pages 154–162, 2014.

G. Govaert. Classification binaire et modeles. *Revue de statistique appliquée*, 38(1):67–81, 1990.

G. Govaert. Simultaneous clustering of rows and columns. *Control and Cybernetics*, 24: 437–458, 1995.

G. Govaert. *Data Analysis*. ISTE-Wiley, 2009.

G. Govaert and M. Nadif. Comparison of the mixture and the classification maximum likelihood in cluster analysis with binary data. *Computational Statistics & Data Analysis*, 23(1):65–81, 1996.

G. Govaert and M. Nadif. Clustering with block mixture models. *Pattern Recognition*, 36(2): 463–473, 2003.

G. Govaert and M. Nadif. An EM algorithm for the block mixture model. *IEEE Trans. Pattern Anal. Mach. Intell.*, 27(4):643–647, 2005.

G. Govaert and M. Nadif. Clustering of contingency table and mixture model. *European Journal of Operational Research*, 183(3):1055–1066, 2007.

G. Govaert and M. Nadif. Block clustering with bernoulli mixture models: Comparison of different approaches. *Computational Statistics & Data Analysis*, 52(6):3233–3245, 2008.

G. Govaert and M. Nadif. Latent block model for contingency table. *Communications in Statistics—Theory and Methods*, 39(3):416–425, 2010.

G. Govaert and M. Nadif. *Co-Clustering*. John Wiley & Sons, 2013.

Q. Gu, J. Zhou, and C. H. Ding. Collaborative filtering: Weighted nonnegative matrix factorization incorporating user and item graphs. In *SDM*, pages 199–210. SIAM, 2010.

R. Guigourès, M. Boullé, and F. Rossi. Discovering patterns in time-varying graphs: a triclustering approach. *Advances in Data Analysis and Classification*, pages 1–28, 2015.

G. Guo, J. Zhang, and N. Yorke-Smith. A novel bayesian similarity measure for recommender systems. In *IJCAI*, pages 2619–2625, 2013.

G. Guo, J. Zhang, D. Thalmann, and N. Yorke-Smith. Etaf: An extended trust antecedents framework for trust prediction. In *ASONAM*, pages 540–547, 2014.

G. Guo, J. Zhang, and N. Yorke-Smith. Trustsvd: Collaborative filtering with both the explicit and implicit influence of user trust and of item ratings. In *AAAI*, pages 123–129, 2015.

N. Gupta and S. Aggarwal. Mib: Using mutual information for biclustering gene expression data. *Pattern Recognition*, 43(8):2692–2697, 2010.

M. Hahsler. recommenderlab: A framework for developing and testing recommendation algorithms, 2011.

S. Han, Y. Yang, and W. Liu. Incremental learning for dynamic collaborative filtering. *Journal of Software*, 6(6):969–976, 2011.

B. Hanczar and M. Nadif. Ensemble methods for biclustering tasks. *Pattern Recognition*, 45 (11):3938–3949, 2012.

J. A. Hartigan. Direct clustering of a data matrix. *Journal of the american statistical association*, 67(337):123–129, 1972.

J. A. Hartigan. *Clustering Algorithms*. John Wiley & Sons, Inc., New York, NY, USA, 99th edition, 1975.

T. Hastie, R. Mazumder, J. Lee, and R. Zadeh. Matrix Completion and Low-Rank SVD via Fast Alternating Least Squares. *ArXiv e-prints*, 2014.

X. He, D. Cai, Y. Shao, H. Bao, and J. Han. Laplacian regularized gaussian mixture model for data clustering. *Knowledge and Data Engineering, IEEE Transactions on*, 23(9): 1406–1418, 2011.

A. Hinneburg and D. A. Keim. An efficient approach to clustering in large multimedia databases with noise. In *KDD*, volume 98, pages 58–65, 1998.

T. Hofmann and J. Puzicha. Latent class models for collaborative filtering. In *IJCAI*, volume 99, pages 688–693, 1999.

T. Hofmann, J. Puzicha, and M. I. Jordan. Learning from dyadic data. *Advances in neural information processing systems*, pages 466–472, 1999.

K. Hornik, I. Feinerer, M. Kober, and C. Buchta. Spherical k-means clustering. *Journal of Statistical Software*, 50(10):1–22, 2012.

L. Hubert and P. Arabie. Comparing partitions. *Journal of classification*, 2(1):193–218, 1985.

M. Jamali and M. Ester. A matrix factorization technique with trust propagation for recommendation in social networks. In *ACM RecSys*, pages 135–142, 2010.

F.-X. Jollois and M. Nadif. Clustering large categorical data. In *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pages 257–263. Springer, 2002.

L. Kaufman and P. Rousseeuw. Clustering by means of medoids. *Statistical Data Analysis Based on the L1-Norm and Related Methods*, pages North–Holland, 1987.

L. Kaufman and P. J. Rousseeuw. *Finding groups in data: an introduction to cluster analysis*, volume 344. John Wiley & Sons, 2009.

C. Keribin, V. Brault, G. Celeux, and G. Govaert. Estimation and selection for the latent block model on categorical data. *Statistics and Computing*, 25(6):1201–1216, 2015.

M. Khoshneshin and W. N. Street. Incremental collaborative filtering via evolutionary co-clustering. In *Proceedings of the fourth ACM conference on Recommender systems*, pages 325–328, 2010.

Y.-D. Kim and S. Choi. Bayesian binomial mixture model for collaborative prediction with non-random missing data. RecSys '14, pages 201–208, 2014.

Y. Koren. Factorization meets the neighborhood: a multifaceted collaborative filtering model. In *Proceedings of the 14th ACM SIGKDD*, pages 426–434. ACM, 2008.

Y. Koren. The bellkor solution to the netflix grand prize, 2009.

Y. Koren, R. Bell, and C. Volinsky. Matrix factorization techniques for recommender systems. *Computer*, (8):30–37, 2009.

H.-P. Kriegel, P. Kröger, J. Sander, and A. Zimek. Density-based clustering. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 1(3):231–240, 2011.

L. Labiod and M. Nadif. Co-clustering under nonnegative matrix tri-factorization. In *Neural Information Processing*, pages 709–717. Springer, 2011a.

L. Labiod and M. Nadif. Co-clustering for binary and categorical data with maximum modularity. In *2011 11th IEEE International Conference on Data Mining*, pages 1140–1145, 2011b.

P. Latouche, E. Birmelé, and C. Ambroise. *Bayesian methods for graph clustering*. Springer, 2009.

T. Le and H. W. Lauw. Semantic visualization for spherical representation. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 1007–1016. ACM, 2014.

T. Leighton and S. Rao. An approximate max-flow min-cut theorem for uniform multicommodity flow problems with applications to approximation algorithms. In *Foundations of Computer Science, 1988., 29th Annual Symposium on*, pages 422–431. IEEE, 1988.

B. Li, Q. Yang, and X. Xue. Can movies and books collaborate?: Cross-domain collaborative filtering for sparsity reduction. In *Proceedings of the 21st International Jont Conference on Artifical Intelligence*, pages 2052–2057, 2009.

T. Li. A general model for clustering binary data. In *Proceedings of the eleventh ACM SIGKDD international conference on Knowledge discovery in data mining*, pages 188–197, 2005.

G. Linden, B. Smith, and J. York. Amazon.com recommendations: Item-to-item collaborative filtering. *IEEE Internet Computing*, 7(1):76–80, 2003. ISSN 1089-7801.

R. J. A. Little and D. B. Rubin. *Statistical analysis with missing data (second edition)*. Chichester: Wiley, 2002.

H. Liu, Z. Hu, A. Mian, H. Tian, and X. Zhu. A new user similarity model to improve the accuracy of collaborative filtering. *Knowledge-Based Systems*, 56:156–166, 2014.

D. Loiacono, A. Lommatzsch, and R. Turrin. An analysis of the 2014 recsys challenge. In *ACM RecSys*, page 1, 2014.

A. Lomet, G. Govaert, and Y. Grandvalet. Model selection for gaussian latent block clustering with the integrated classification likelihood. *Advances in Data Analysis and Classification*, pages 1–20, 2014.

B. Long, Z. M. Zhang, and P. S. Yu. Co-clustering by block value decomposition. In *Proceedings of the eleventh ACM SIGKDD international conference on Knowledge discovery in data mining*, pages 635–640. ACM, 2005.

H. Ma, H. Yang, M. R. Lyu, and I. King. Sorec: social recommendation using probabilistic matrix factorization. In *CIKM*, pages 931–940, 2008.

H. Ma, I. King, and M. R. Lyu. Learning to recommend with social trust ensemble. In *ACM SIGIR*, pages 203–210. ACM, 2009.

H. Ma, D. Zhou, C. Liu, M. R. Lyu, and I. King. Recommender systems with social regularization. In *ACM WSDM*, pages 287–296, 2011.

J. MacQueen et al. Some methods for classification and analysis of multivariate observations. In *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability*, volume 1, pages 281–297. Oakland, CA, USA., 1967.

S. C. Madeira and A. L. Oliveira. Biclustering algorithms for biological data analysis: a survey. *IEEE/ACM Transactions on Computational Biology and Bioinformatics (TCBB)*, 1 (1):24–45, 2004.

K. V. Mardia and P. E. Jupp. *Directional statistics*. Wiley series in probability and statistics. Wiley, 2000.

B. M. Marlin, R. S. Zemel, S. Roweis, and M. Slaney. Collaborative filtering and the missing at random assumption. In *In Proceedings of the 23rd Conference on Uncertainty in Artificial Intelligence (UAI)*, 2007.

R. Mazumder, T. Hastie, and R. Tibshirani. Spectral regularization algorithms for learning large incomplete matrices. *J. Mach. Learn. Res.*, 11:2287–2322, 2010.

A. McCallum, K. Nigam, et al. A comparison of event models for naive bayes text classification. In *AAAI-98 workshop on learning for text categorization*, volume 752, pages 41–48. Citeseer, 1998.

G. McLachlan and T. Krishnan. *The EM algorithm and extensions*, volume 382. John Wiley & Sons, 2007.

G. McLachlan and D. Peel. *Finite mixture models*. John Wiley & Sons, 2004.

G. J. McLachlan and K. E. Basford. Mixture models. inference and applications to clustering. *Statistics: Textbooks and Monographs, New York: Dekker, 1988*, 1, 1988.

Q. Mei, D. Cai, D. Zhang, and C. Zhai. Topic modeling with network regularization. In *Proceedings of the 17th international conference on World Wide Web*, pages 101–110, 2008.

G. W. Milligan and M. C. Cooper. A study of the comparability of external criteria for hierarchical cluster analysis. *Multivariate Behavioral Research*, 21(4):441–458, 1986.

F. Murtagh and P. Contreras. Algorithms for hierarchical clustering: an overview. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 2(1):86–97, 2012.

M. Nadif and G. Govaert. Clustering for binary data and mixture models—choice of the model. *Applied Stochastic Models and Data Analysis*, 13(3-4):269–278, 1998.

M. Nadif and G. Govaert. Model-based co-clustering for continuous data. In *Machine Learning and Applications (ICMLA), 2010 Ninth International Conference on*, pages 175–180, 2010.

M. C. Nascimento and A. C. De Carvalho. Spectral methods for graph clustering–a survey. *European Journal of Operational Research*, 211(2):221–231, 2011.

R. Neal and G. E. Hinton. A view of the em algorithm that justifies incremental, sparse, and other variants. In *Learning in Graphical Models*, pages 355–368. Kluwer Academic Publishers, 1998.

M. E. Newman and M. Girvan. Finding and evaluating community structure in networks. *Physical review E*, 69(2):026113, 2004.

M. Papagelis, I. Rousidis, D. Plexousakis, and E. Theoharopoulos. Incremental collaborative filtering for highly-scalable recommendation algorithms. In *Foundations of Intelligent Systems*, pages 553–561. Springer, 2005.

M. J. Pazzani and D. Billsus. Content-based recommendation systems. In *The adaptive web*, pages 325–341. Springer, 2007.

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, et al. Scikit-learn: Machine learning in python. *The Journal of Machine Learning Research*, 12:2825–2830, 2011.

J. Reisinger, A. Waters, B. Silverthorn, and R. J. Mooney. Spherical topic models. In *Proceedings of the 27th International Conference on Machine Learning (ICML-10)*, pages 903–910, 2010.

A. Salah, N. Rogovschi, and M. Nadif. A dynamic collaborative filtering system via a weighted clustering approach. *Neurocomputing*, 2015.

R. Salakhutdinov and A. Mnih. Probabilistic matrix factorization. In *NIPS*, volume 20, pages 1257–1264, 2008.

B. Sarwar, G. Karypis, J. Konstan, and J. Riedl. Item-based collaborative filtering recommendation algorithms. In *Proceedings of the 10th international conference on World Wide Web (ICWWW)*, pages 285–295. ACM, 2001.

B. Sarwar, G. Karypis, J. Konstan, and J. Riedl. Incremental singular value decomposition algorithms for highly scalable recommender systems. In *Fifth International Conference on Computer and Information Science*, pages 27–28, 2002.

B. M. Sarwar, G. Karypis, J. A. Konstan, and J. T. Riedl. Application of dimensionality reduction in recommender system – a case study. In *ACM WEBKDD WORKSHOP*, 2000.

S. E. Schaeffer. Graph clustering. *Computer Science Review*, 1(1):27–64, 2007.

G. Schwarz et al. Estimating the dimension of a model. *The annals of statistics*, 6(2): 461–464, 1978.

A. J. Scott and M. J. Symons. Clustering methods based on likelihood ratio criteria. *Biometrics*, pages 387–397, 1971.

H. Shan and A. Banerjee. Bayesian co-clustering. In *Data Mining, 2008. ICDM'08. Eighth IEEE International Conference on*, pages 530–539, 2008.

J. Shi and J. Malik. Normalized cuts and image segmentation. In *Computer Vision and Pattern Recognition, 1997. Proceedings., 1997 IEEE Computer Society Conference on*, pages 731–737. IEEE, 1997.

J. Shi and J. Malik. Normalized cuts and image segmentation. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 22(8):888–905, 2000.

R. Sibson. Slink: an optimally efficient algorithm for the single-link cluster method. *The Computer Journal*, 16(1):30–34, 1973.

V. Sindhwani, S. Bucak, J. Hu, and A. Mojsilovic. A family of non-negative matrix factorizations for one-class collaborative filtering problems. In *Proceedings of the ACM Recommender Systems Conference, RecSys*, 2009.

R. R. Sokal. A statistical method for evaluating systematic relationships. *Univ Kans Sci Bull*, 38:1409–1438, 1958.

T. Sorensen. A method of establishing groups of equal amplitude in plant sociology based on similarity of species and its application to analyses of the vegetation on danish commons. *Biol. Skr.*, 5:1–34, 1948.

S. Sra. A short note on parameter approximation for von mises-fisher distributions: and a fast implementation of i s (x). *Computational Statistics*, 27(1):177–190, 2012.

A. Strehl and J. Ghosh. Cluster ensembles—a knowledge reuse framework for combining multiple partitions. *The Journal of Machine Learning Research*, 3:583–617, 2003.

A. Strehl, J. Ghosh, and R. Mooney. Impact of similarity measures on web-page clustering. In *Workshop on Artificial Intelligence for Web Search (AAAI 2000)*, pages 58–64, 2000.

M. J. Symons. Clustering criteria and multivariate normal mixtures. *Biometrics*, pages 35–43, 1981.

A. Tanabe, K. Fukumizu, S. Oba, T. Takenouchi, and S. Ishii. Parameter estimation for von mises–fisher distributions. *Computational Statistics*, 22(1):145–157, 2007.

J. Tang, H. Gao, and H. Liu. mtrust: discerning multi-faceted trust in a connected world. In *ACM WSDM*, pages 93–102, 2012.

J. Tang, X. Hu, and H. Liu. Social recommendation: a review. *Social Network Analysis and Mining*, 3(4):1113–1133, 2013.

L. H. Ungar and D. P. Foster. Clustering methods for collaborative filtering. In *AAAI workshop on recommendation systems*, volume 1, pages 114–129, 1998.

B. van Dijk, J. van Rosmalen, and R. Paap. A bayesian approach to two-mode clustering, 2009.

I. Van Mechelen, H.-H. Bock, and P. De Boeck. Two-mode clustering methods: a structured overview. *Statistical methods in medical research*, 13(5):363–394, 2004.

U. Von Luxburg. A tutorial on spectral clustering. *Statistics and computing*, 17(4):395–416, 2007.

J. Wang and L. Ke. Feature subspace transfer for collaborative filtering. *Neurocomputing*, 136:1 – 6, 2014.

W. Wang, J. Yang, R. Muntz, et al. Sting: A statistical information grid approach to spatial data mining. In *VLDB*, volume 97, pages 186–195, 1997.

J. H. Ward Jr. Hierarchical grouping to optimize an objective function. *Journal of the American statistical association*, 58(301):236–244, 1963.

J. Wyse and N. Friel. Block clustering with collapsed latent block models. *Statistics and Computing*, 22(2):415–428, 2012.

G. Xu, Y. Zong, P. Dolog, and Y. Zhang. Co-clustering analysis of weblogs using bipartite spectral projection approach. In *Knowledge-Based and Intelligent Information and Engineering Systems*, pages 398–407. 2010.

R. Xu, D. Wunsch, et al. Survey of clustering algorithms. *Neural Networks, IEEE Transactions on*, 16(3):645–678, 2005.

W. Xu, X. Liu, and Y. Gong. Document clustering based on non-negative matrix factorization. In *Proceedings of the 26th annual international ACM SIGIR conference on Research and development in informaion retrieval*, pages 267–273, 2003.

B. Yang, Y. Lei, D. Liu, and J. Liu. Social collaborative filtering by trust. In *Proceedings of the Twenty-Third international joint conference on Artificial Intelligence*, pages 2747–2753. AAAI Press, 2013.

X. Yang, Z. Zhang, and K. Wang. Scalable collaborative filtering using incremental update and local link prediction. In *Proceedings of the 21st ACM international conference on Information and knowledge management*, pages 2371–2374. ACM, 2012.

J. Yoo and S. Choi. Orthogonal nonnegative matrix tri-factorization for co-clustering: Multiplicative updates on stiefel manifolds. *Information processing & management*, 46(5): 559–570, 2010.

W. W. Zachary. An information flow model for conflict and fission in small groups. *Journal of anthropological research*, pages 452–473, 1977.

H. Zanghi, C. Ambroise, and V. Miele. Fast online graph clustering via erdős–rényi mixture. *Pattern Recognition*, 41(12):3592–3599, 2008.

H. Zanghi, S. Volant, and C. Ambroise. Clustering based on random graph model embedding vertex features. *Pattern Recognition Letters*, 31(9):830–836, 2010.

B. Zhang, G. Kleyner, and M. Hsu. A local search approach to k-clustering. *HP LABORATORIES TECHNICAL REPORT HPL*, (119), 1999.

S. Zhang, W. Wang, J. Ford, and F. Makedon. Learning from incomplete ratings using non-negative matrix factorization. In *SDM*, volume 6, pages 548–552. SIAM, 2006.

S. Zhong. Efficient online spherical k-means clustering. In *IJCNN*, pages 3180–3185, 2005.

S. Zhong and J. Ghosh. Generative model-based document clustering: a comparative study. *Knowledge and Information Systems*, 8(3):374–384, 2005.

X. Zhu and J. Lafferty. Harmonic mixtures: combining mixture models and graph-based methods for inductive and scalable semi-supervised learning. In *ICML*, pages 1052–1059, 2005.

Appendix A

Parameters Estimation

In this appendix, we provide the derivation details of maximum likelihood estimates for parameters of the diagonal block mixture of von Mises-Fisher distributions (*dbmovMFs*).

A.1 Maximum Likelihood Estimate

The expectation of the complete data log-likelihood of *dbmovMFs* is given by

$$\begin{aligned} E[L_c(\Theta|\mathcal{X},\mathbf{Z})] &= \sum_{h,i}\tilde{z}_{ih}\log\alpha_h + \sum_{h,i}\tilde{z}_{ih}\log(c_d(\kappa_h)) + \sum_{h,i,j}\tilde{z}_{ih}w_{jh}\kappa_h\mu_{hh}x_{ij} \\ &= \sum_{h}\tilde{z}_{.h}\log\alpha_h + \sum_{h}\tilde{z}_{.h}\log(c_d(\kappa_h)) + \sum_{h,i,j}\tilde{z}_{ih}w_{jh}\kappa_h\mu_{hh}x_{ij} \end{aligned} \tag{A.1}$$

where $\tilde{z}_{.h} = \sum_i \tilde{z}_{ih}$. We first maximize the expectation of the complete log-likelihood (A.1) with respect to α_h, subject to the constraint $\sum_h \alpha_h = 1$. The corresponding Lagrangian, up to terms which are not function of α_h, is given by

$$L(\alpha,\lambda) = \sum_{h}\tilde{z}_{.h}\log\alpha_h + \lambda_h(1-\sum_{h}\alpha_h) \tag{A.2}$$

Taking derivatives with respect to α_h, we obtain $\frac{\partial L(\alpha,\lambda)}{\partial\alpha_h} = \frac{\tilde{z}_{.h}}{\alpha_h} - \lambda$. Setting this derivative to zero leads to $\tilde{z}_{.h} = \lambda\alpha_h$. Summing both sides over all h yields $\lambda = n$, thereby the maximizing value of the parameter α_h is given by:

$$\hat{\alpha}_h = \frac{\tilde{z}_{.h}}{n}. \tag{A.3}$$

In the same manner, to maximize expectation (A.1) with respect to $\mu_h^{\mathbf{w}}$, subject to the constraint $(\mu_h^{\mathbf{w}})^T \mu_h^{\mathbf{w}} = 1$, we form the corresponding Lagrangian by isolating the terms which depends on $\mu_h^{\mathbf{w}}$, this leads to

$$L(\mu, \lambda) = \sum_{h,i,j} \tilde{z}_{ih} w_{jh} \kappa_h \mu_{hh} x_{ij} + \lambda_h (1 - \sum_j w_{jh} \mu_{hh}^2)$$

Taking the derivative with respect to μ_{hh}, we obtain:

$$\frac{\partial L(\mu, \lambda)}{\partial \mu_h} = \sum_{i,j} \tilde{z}_{ih} w_{jh} \kappa_h x_{ij} - 2\lambda w_{.h} \mu_{hh}$$

where $w_{.h} = \sum_j w_{jh}$. Setting this derivative to zero, we obtain $\lambda \mu_{hh} = \frac{\sum_{i,j} \tilde{z}_{ih} w_{jh} \kappa_h x_{ij}}{2 w_{.h}}$. Thus,

$$\lambda^2 \mu_{hh}^2 = \frac{(\sum_{i,j} \tilde{z}_{ih} w_{jh} \kappa_h x_{ij})^2}{4 w_{.h}^2}.$$

Multiplying both sides by $w_{.h}$, yields:

$$\lambda^2 w_{.h} \mu_{hh}^2 = \frac{(\sum_{i,j} \tilde{z}_{ih} w_{jh} \kappa_h x_{ij})^2}{4 w_{.h}}. \tag{A.4}$$

Hence, we obtain $\lambda = \kappa_h \frac{\sqrt{w_{.h} (\sum_{i,j} \tilde{z}_{ih} w_{jh} x_{ij})^2}}{2 w_{.h}} = \kappa_h \frac{\|\mathbf{r}_h^{\mathbf{w}}\|}{2 w_{.h}}$ where $\mathbf{r}_h^{\mathbf{w}}$ is a d dimensional vector, i.e, let $j' = 1, \ldots, d$, $r_{hj'}^{\mathbf{w}} = r_h^{\mathbf{w}} = \sum_{i,j} \tilde{z}_{ih} w_{jh} x_{ij}$ if $w_{jh} = 1$ and $r_{hj'}^{\mathbf{w}} = 0$, otherwise. Hence, the maximizing value of the parameter μ_{hh} is given by:

$$\begin{aligned} \hat{\mu}_{hh} &= \frac{\sum_{i,j} \tilde{z}_{ih} w_{jh} x_{ij}}{\|\mathbf{r}_h^{\mathbf{w}}\|} \\ &= \frac{\sum_{i,j} \tilde{z}_{ih} w_{jh} x_{ij}}{\sqrt{w_{.h} (\sum_{i,j} \tilde{z}_{ih} w_{jh} x_{ij})^2}} \\ &= \pm \frac{1}{\sqrt{w_{.h}}} \end{aligned} \tag{A.5}$$

according to whether $r_h^{\mathbf{w}} = \sum_{i,j} \tilde{z}_{ih} w_{jh} x_{ij}$ is positive or negative. It follows from equation (A.5) that given the column partition $\mathbf{w}$ and the sign of $r_h^{\mathbf{w}}$, the centroid parameter $\mu_h^{\mathbf{w}}$ can be deduced directly.

Next we concentrate on maximizing equation (A.1), with respect to the concentration parameters κ_h, subject to the constraint $\kappa_h > 0$, $\forall h$. The Lagrangian up to terms which do

not contains κ_h is given by

$$L(\kappa) = \sum_h \tilde{z}_{.h} \log(c_d(\kappa_h)) + \sum_{h,i,j} \tilde{z}_{ih} w_{jh} \kappa_h \hat{\mu}_{hh} x_{ij}$$

note that, by KKT conditions, the Lagrangian multiplier for the constraint $\kappa_h > 0$ has to be equal to zero. Taking the partial derivative of equation (A.6) with respect to κ_h, we obtain

$$\frac{\partial L(\kappa)}{\partial \kappa_h} = \tilde{z}_{.h} \frac{c'_d(\kappa_h)}{c_d(\kappa_h)} + \sum_{i,j} \tilde{z}_{ih} w_{jh} \hat{\mu}_{hh} x_{ij}$$

Setting this derivative equal to zero, leads to:

$$\frac{c'_d(\kappa_h)}{c_d(\kappa_h)} = -\frac{\hat{\mu}_{hh} \times \sum_{i,j} \tilde{z}_{ih} w_{jh} x_{ij}}{\tilde{z}_{.h}}.$$

Replacing $\hat{\mu}_{hh}$ by $\frac{\sum_{i,j} \tilde{z}_{ih} w_{jh} x_{ij}}{\|\mathbf{r}_h^{\mathbf{w}}\|}$ (see, equation A.5), we obtain $\frac{c'_d(\kappa_h)}{c_d(\kappa_h)} = -\frac{\|\mathbf{r}_h^{\mathbf{w}}\|}{\tilde{z}_{.h} \hat{w}_{.h}}$. Let $s = d/2 - 1$, then:

$$\begin{aligned} c'_d(\kappa_h) &= \frac{s\kappa_h^{s-1}(2\pi)^{s+1} I_s(\kappa_h) - \kappa_h^s (2\pi)^{s+1} I'_s(\kappa_h)}{(2\pi)^{2s+2} I_s^2(\kappa_h)} \\ &= \frac{s\kappa_h^{s-1}}{(2\pi)^{s+1} I_s(\kappa_h)} - \frac{\kappa_h^s I'_s(\kappa_h)}{(2\pi)^{s+1} I_s^2(\kappa_h)} \\ &= c_d(\kappa_h) \left(\frac{s}{\kappa_h} - \frac{I'_s(\kappa_h)}{I_s(\kappa_h)} \right). \end{aligned} \tag{A.6}$$

Hence, we obtain

$$\frac{-c'_d(\kappa_h)}{c_d(\kappa_h)} = \frac{I_{s+1}(\kappa_h)}{I_s(\kappa_h)} = \frac{I_{d/2}(\kappa_h)}{I_{d/2-1}(\kappa_h)}. \tag{A.7}$$

The latter equation (A.7), arises from the use of the following recurrence formula (Abramowitz and Stegun (1964), page 376):

$$\kappa_h I_{s+1}(\kappa_h) = \kappa_h I'_s(\kappa_h) - s I_s(\kappa_h). \tag{A.8}$$

Note that computing the maximizing value $\hat{\kappa}_h$ from equation (A.6) implies to inverse a ratio of Bessel function, a problem for which no a closed-form solution can be obtained. Thus, following (Banerjee et al., 2005b), we propose to derive an accurate approximation of the concentration parameter, by using the following continued fraction formula:

$$\frac{I_{d/2}(\kappa_h)}{I_{d/2-1}(\kappa_h)} = \frac{1}{\frac{d}{\kappa_h} + \frac{1}{\frac{d+2}{\kappa_h} + \ldots}}. \tag{A.9}$$

Letting $\bar{r}_h^{\mathbf{w}} = \frac{\|\mathbf{r}_h^{\mathbf{w}}\|}{\tilde{z}_{.h}\hat{w}_{.h}} = \frac{I_{d/2}(\kappa_h)}{I_{d/2-1}(\kappa_h)}$ and using equation (A.9), we obtain: $\frac{1}{\bar{r}_h^{\mathbf{w}}} \approx \frac{d}{\kappa_h} + \bar{r}_h^{\mathbf{w}}$ which yields the following approximation:

$$\hat{\kappa}_h = \frac{d\bar{r}_h^{\mathbf{w}}}{1-(\bar{r}_h^{\mathbf{w}})^2}.$$

Finally, Banerjee et al. (2005b) have empirically shown that adding the following correction term $\frac{-(\bar{r}_h^{\mathbf{w}})^3}{1-(\bar{r}_h^{\mathbf{w}})^2}$ results in a better approximation of $\hat{\kappa}_h$, which leads to:

$$\hat{\kappa}_h = \frac{d\bar{r}_h^{\mathbf{w}} - (\bar{r}_h^{\mathbf{w}})^3}{1-(\bar{r}_h^{\mathbf{w}})^2}. \tag{A.10}$$

A.2 Concentration parameters: *dbmovMFs* vs *movMFs*

Hereafter, we provide the proofs for Proposition 1 and Theorem 1. Recall that as opposed to the classical *movMFs* where it is easy to verify that $\bar{r}_h \leq 1$ (see equation 6*c*) given the definition of $\mathbf{r}$, it is not straightforward to verify that $\bar{r}_h^{\mathbf{w}} \leq 1$, without careful analysis. Proposition 1 provides theoretical guarantee about the fact that $0 \leq \bar{r}_h^{\mathbf{w}} \leq 1$, which is imperative, to guarantee that the concentration parameters are positive, i.e, $\kappa_h > 0$, $\forall h$, specially when using the approximation of equation (A.10). The following Proposition is useful and necessary to prove both Proposition 1 and Theorem 1.

Proposition 3 *Let* $\mathbf{r}$ *be a non-zero vector in* $\mathbb{R}^d$ *(i.e.,* $\mathbf{r} = (r_1, \ldots, r_d)^T$*, such as* $d \geq 1$*). Let* $\mathbf{r}^d$ *be a vector in* $\mathbb{R}^d$*, such as all its component are equal to the sum of elements of* $\mathbf{r}$ *(i.e,* $\mathbf{r}^d = \sum_{j=1}^d r_j \mathbb{1}$ *where* $\mathbb{1}$ *denotes the constant one vector). Then* $\frac{\|\mathbf{r}^d\|}{d} \leq \|\mathbf{r}\|$ *with equality only if all components of* $\mathbf{r}$ *are equal (i.e,* $r_1 = \cdots = r_d$*).*

Proof. Let $\mathbf{d}$ and $\mathbf{r}^+$ be two vectors in $\mathbb{R}^d$ defined as follows: $\mathbf{d} = \frac{1}{\sqrt{d}}\mathbb{1}$ and $\mathbf{r}^+$ with $r_j^+ = |r_j|, \quad \forall j \in \{1, \ldots, d\}$. We have

$$\frac{\|\mathbf{r}^d\|}{d} = \frac{\sqrt{d} \times \left|\sum_j r_j\right|}{d} \leq \frac{1}{\sqrt{d}} \times \sum_j \left|r_j\right| = \mathbf{d}^T.\mathbf{r}^+ = \|\mathbf{d}\|\|\mathbf{r}^+\| \cos(\mathbf{d}, \mathbf{r}^+).$$

By definition of $\mathbf{r}^+$ and $\mathbf{d}$, we have $\|\mathbf{r}^+\| = \|\mathbf{r}\|$ and $\|\mathbf{d}\| = 1$, hence $\frac{\|\mathbf{r}^d\|}{d} \leq \|\mathbf{r}\| \cos(\mathbf{d}, \mathbf{r}^+)$. Since both $\mathbf{d}$ and $\mathbf{r}^+$ are non-zero vectors and lie on the first orthant of a d-dimensional unit hypersphere, by dividing both sides of the above inequality by $\|\mathbf{r}\|$, we get

$$0 \leq \frac{\|\mathbf{r}^d\|}{d\|\mathbf{r}\|} \leq \cos(\mathbf{d}, \mathbf{r}^+) \leq 1$$

The right hand side equality holds only if $\mathbf{d}$ and $\mathbf{r}^+$ are collinear, thereby all components of $\mathbf{r}$ are equal (i.e,$r_1 = \cdots = r_d$). Hereafter we provide the proof of Proposition 1.

Proposition 1 *Let* $\mathbf{r}$ *be a non-zero vector in* $\mathbb{R}^d$ *(i.e.,* $\mathbf{r} = (r_1, \ldots, r_d)^T$*, such as* $d \geq 1$*) which results from a weighted sum of n d-dimensional unit vector, i.e,* $\mathbf{r} = \sum_i p_i \mathbf{x}_i$*,* $\mathbf{x}_i \in \mathbb{R}^d$ *and* $\|\mathbf{x}_i\| = 1$, $\forall i \in \{1, \ldots, n\}$, $n \geq 2$*, the weights* $p_i \geq 0$*,* $\forall i$*. Let* $\mathbf{r}^d$ *be a vector in* $\mathbb{R}^d$*, such as all its components are equal to the sum of elements of* $\mathbf{r}$ *(i.e,* $\mathbf{r}^d = \sum_{j=1}^{d} r_j \mathbb{1}$*). Then* $\|\mathbf{r}^d\| \leq d \times \sum_i p_i$ *with equality only if all unit vectors* $\mathbf{x}_i$ *are equal/collinear.*

Proof. Based on Proposition 3 and the following inequality: $\|\mathbf{r}\| = \|p_1\mathbf{x}_1 + \cdots + p_n\mathbf{x}_n\| \leq \|p_1\mathbf{x}_1\| + \cdots + \|p_n\mathbf{x}_n\| = \sum_i p_i$, it is straightforward to verify that

$$0 \leq \|\mathbf{r}^d\| \leq d \times \sum_i p_i$$

In the following, we prove Theorem 1, which states that for a given row clustering $\mathbf{z}$, *dbmovMFs*-based algorithms lead to a concentration parameter that is less or equal to that of movMFs-based algorithms, for each cluster, and whatever the column partition $\mathbf{w}$. The following Lemma will be useful in the proof of Theorem 1.

Lemma 1 *Let a and b be two real numbers in the interval* $[0,1]$ *(i.e,* $0 \leq a \leq 1$ *and* $0 \leq b \leq 1$*). Then for all natural number* $n \geq 2$

$$|a^n - b^n| \leq n|a-b|$$

with equality only if $a = b$*.*

Proof. For all natural number $n \geq 2$, we can use the following well known remarkable identity:

$$a^n - b^n = (a-b)\sum_{k=0}^{n-1} a^{n-1-k}b^k$$

since both a and b are positive, we have

$$|a^n - b^n| = |a-b|\sum_{k=0}^{n-1} a^{n-1-k}b^k$$

as both a and b are in $[0,1]$, we have

$$\begin{aligned} a^{n-1-0}b^0 &\leq 1 \\ a^{n-1-1}b^1 &\leq 1 \\ &\vdots \\ a^{n-1-(n-1)}b^{n-1} &\leq 1 \end{aligned}$$

Taking the sum of the above n inequalities we obtain,

$$\sum_{k=0}^{n-1} a^{n-1-k}b^k \leq n$$

Thereby,

$$|a^n - b^n| = |a-b| \sum_{k=0}^{n-1} a^{n-1-k}b^k \leq n|a-b|$$

Theorem 1 *Let* $\mathbf{X}$ *be a* $n \times d$ *matrix, its* i^{th} *row (object)* $\mathbf{x}_i$ *is a* d*-dimensional unit vector in* $\mathbb{S}^{d-1}$ *(i.e,* $\mathbf{x}_i \in \mathbb{R}^d$ *and* $\|\mathbf{x}_i\| = 1$, $\forall i \in \{1,\ldots,n\}$, $n \geq 2$, $d \geq 3$*). Let* $\mathbf{z} = (z_1,\ldots,z_n)$ *denote a partition of the set of objects of* $\mathbf{X}$ *into* g *disjoint clusters. Then, whatever the partition* $\mathbf{w}$ *of attributes of* $\mathbf{X}$ *into* g *disjoints clusters, the concentration parameter of each dbvMF component estimated via approximation (5.6d) is always less or equal to the concentration parameter of the corresponding vMF component estimated via approximation (3.13c). That is,*

$$\hat{\kappa}_h^{\mathbf{w}} \approx \frac{\bar{r}_h^{\mathbf{w}} d - \left(\bar{r}_h^{\mathbf{w}}\right)^3}{1 - \left(\bar{r}_h^{\mathbf{w}}\right)^2} \leq \hat{\kappa}_h \approx \frac{\bar{r}_h d - (\bar{r}_h)^3}{1 - (\bar{r}_h)^2}$$

with equality only if $\bar{r}_h^{\mathbf{w}} = \bar{r}_h$.

Proof. We first prove that

$$\frac{\bar{r}_h^{\mathbf{w}} d}{1 - \left(\bar{r}_h^{\mathbf{w}}\right)^2} \leq \frac{\bar{r}_h d}{1 - (\bar{r}_h)^2}$$

which corresponds to the approximation of the concentration parameters under *dbmovMFs* and *movMFs* without the correction term. Since both $\bar{r}_h^{\mathbf{w}}$ and $\bar{r}_h$ are in $]0,1[$, it is straightforward to verify that if $\bar{r}_h^{\mathbf{w}} \leq \bar{r}_h$ then the above inequality is always verified. Thus, in what follows we aim to demonsrate that $\bar{r}_h^{\mathbf{w}} \leq \bar{r}_h$. By definition, we have

$$\bar{r}_h = \frac{\|\mathbf{r}_h\|}{\sum_i \tilde{z}_{ih}} \quad \text{where} \quad \mathbf{r}_h = \sum_i \tilde{z}_{ih}\mathbf{x}_i$$

Now, let's define $\bar{r}'_h$ as follows

$$\bar{r}'_h = \frac{\|\mathbf{r}'_h\|}{\sum_i \tilde{z}_{ih}} \quad \text{where} \quad r'_{hj} = \begin{cases} r_{hj}, & \text{if } w_{jh} = 1 \\ 0, & \text{otherwise.} \end{cases}$$

$\mathbf{r}'_h$ is a $w_{.h}$ dimensional sub vector of $\mathbf{r}_h$, it follows from the above definition that $\bar{r}'_h \leq \bar{r}_h$. On the other hand,

$$\bar{r}^{\mathbf{w}}_h = \frac{\|\mathbf{r}^{\mathbf{w}}_h\|}{\sum_i \tilde{z}_{ih} \sum_j \hat{w}_{jh}}$$

where $\mathbf{r}^{\mathbf{w}}_h$ denotes a $w_{.h}$ dimensional vector, each its elements are equal to sum of elements of the h^{th} co-cluster (i.e, $r^{\mathbf{w}}_{h1} = \cdots = r^{\mathbf{w}}_{hw_{.h}} = r^{\mathbf{w}}_h = \sum_{i,j} \tilde{z}_{ih} \hat{w}_{jh} x_{ij}$), similarly we can show that each elements of $\mathbf{r}^{\mathbf{w}}_h$ are equal to sum of elements of r'_h (i.e, $r^{\mathbf{w}}_h = \sum_j r'_{hj}$). Hence using Proposition 3, where the dimensionality $d = w_{.h}$, we have

$$\frac{\|\mathbf{r}^{\mathbf{w}}_h\|}{\sum_j \hat{w}_{jh}} \leq \|\mathbf{r}'_h\|$$

thus,

$$\bar{r}^{\mathbf{w}}_h = \frac{\|\mathbf{r}^{\mathbf{w}}_h\|}{\sum_i \tilde{z}_{ih} \sum_j \hat{w}_{jh}} \leq \bar{r}'_h = \frac{\|\mathbf{r}'_h\|}{\sum_i \tilde{z}_{ih}} \leq \bar{r}_h$$

Thereby,

$$\frac{\bar{r}^{\mathbf{w}}_h d}{1 - \left(\bar{r}^{\mathbf{w}}_h\right)^2} \leq \frac{\bar{r}_h d}{1 - (\bar{r}_h)^2}.$$

We now show that,

$$\frac{\bar{r}^{\mathbf{w}}_h d - (\bar{r}^{\mathbf{w}}_h)^3}{1 - \left(\bar{r}^{\mathbf{w}}_h\right)^2} \leq \frac{\bar{r}_h d - \bar{r}^3_h}{1 - (\bar{r}_h)^2}.$$

As $\bar{r}^{\mathbf{w}}_h \leq \bar{r}_h$ we have $\frac{\bar{r}^{\mathbf{w}}_h d - (\bar{r}^{\mathbf{w}}_h)^3}{1 - \left(\bar{r}^{\mathbf{w}}_h\right)^2} \leq \frac{\bar{r}^{\mathbf{w}}_h d - (\bar{r}^{\mathbf{w}}_h)^3}{1 - (\bar{r}_h)^2}$, then it is sufficient to demonstrate that,

$$\frac{\bar{r}^{\mathbf{w}}_h d - (\bar{r}^{\mathbf{w}}_h)^3}{1 - (\bar{r}_h)^2} \leq \frac{\bar{r}_h d - \bar{r}^3_h}{1 - (\bar{r}_h)^2}.$$

We have

$$\frac{\bar{r}^{\mathbf{w}}_h d - (\bar{r}^{\mathbf{w}}_h)^3}{1 - (\bar{r}_h)^2} - \frac{\bar{r}_h d - \bar{r}^3_h}{1 - (\bar{r}_h)^2} = \frac{d(\bar{r}^{\mathbf{w}}_h - \bar{r}_h) + (\bar{r}^3_h - (\bar{r}^{\mathbf{w}}_h)^3)}{1 - (\bar{r}_h)^2}.$$

Based on Lemma 1, for all $d \geq 3$ we have $\left|\bar{r}_h^3 - (\bar{r}_h^{\mathbf{w}})^3\right| \leq d\left|\bar{r}_h^{\mathbf{w}} - \bar{r}_h\right|$. As $\bar{r}_h^{\mathbf{w}} \leq \bar{r}_h$ it is easy to verify that

$$d(\bar{r}_h^{\mathbf{w}} - \bar{r}_h) + (\bar{r}_h^3 - (\bar{r}_h^{\mathbf{w}})^3) \leq 0.$$

Based on the fact that $1 - (\bar{r}_h)^2 > 0$ we have $\frac{\bar{r}_h^{\mathbf{w}} d - (\bar{r}_h^{\mathbf{w}})^3}{1-(\bar{r}_h)^2} \leq \frac{\bar{r}_h d - \bar{r}_h^3}{1-(\bar{r}_h)^2}$. Hence, using the above inequalities we get (for all $d \geq 3$)

$$\frac{\bar{r}_h^{\mathbf{w}} d - (\bar{r}_h^{\mathbf{w}})^3}{1 - \left(\bar{r}_h^{\mathbf{w}}\right)^2} \leq \frac{\bar{r}_h d - \bar{r}_h^3}{1 - (\bar{r}_h)^2}.$$

A.3 Computational Complexity in the Worst Case

Hereafter, we provide the proofs for the computational complexity of (ii) soft-dbmovMF ($\text{EM}_{\mathbf{b}}$) and (iii) hard-dbmovMF ($\text{CEM}_{\mathbf{b}}$) given in Proposition 2 (section 5.6).

Proof (ii). The computational bottleneck for *hard-dbmovMF* is with row, column assignments and concentration parameters updates. From Proposition 2 (i), the total cost of row and column assignments is $O(it \cdot nz)$. We show that the computational cost for updating concentration parameters is $O(it \cdot nz)$. The main computation for updating the h^{th} concentration parameter is with the computation of $r_h^{\mathbf{w}}$. The computational cost of the latter term is given in $O(x_h^*)$, where x_h^* the number of non-zeros entries in the h^{th} co-cluster. Thus, the complexity for updating all concentrations parameters is $O(nz)$, based on $O(x_1^* + \cdots + x_g^*)$ and the fact that at most all non-zeros entries in the matrix $\mathbf{X}$ are contained in the g diagonal co-clusters. Hence, the cost for updating the concentration parameters in *hard-dbmovMF* is $O(it \cdot nz)$, thereby the total cost of *hard-dbmovMF* is $O(it \cdot nz)$.

Proof (iii). As for *hard-dbmovMF* it is easy to verify that the total cost of row assignments and concentration parameters updates is given in $O(it \cdot nz)$ for *soft-dbmovMF*. Now we prove that in contrast to *hard-dbmovMF* the computational cost of column assignments for *soft-dbmovMF* is $O(it \cdot g \cdot nz)$. The computational bottleneck for column assignment step of *soft-dbmovMF* is with the terms $v_{hj} \leftarrow \sum_i \tilde{z}_{ih} x_{ij}$, $h \in \{1,\ldots,g\}$, $j \in \{1,\ldots,d\}$. The cost of v_{hj} is given in $O(x_j^*)$, where x_j^* is the number of non-zeros entries in the j^{th} column. During the column assignment step for each cluster h and column j we compute v_{hj}, hence the computational cost of this step is given in $O(g \cdot nz)$. Therefore, the total computational cost of *soft-dbmovMF* is $O(it \cdot g \cdot nz)$, based on $O(it \cdot nz \cdot (g+1))$.