UNIVERSITY OF CALIFORNIA
RIVERSIDE

Dimensionality Reduction Algorithms With Applications to Collaborative Data and Images

A Dissertation submitted in partial satisfaction
of the requirements for the degree of

Doctor of Philosophy

in

Computer Science

by

Guobiao Mei

August 2008

Acknowledgements

The journey of writing a PhD dissertation can be discouraging and lonely. I could not have achieved these results without all the kind help from numerous people around me.

I could not have started my PhD career and hence written the thesis without the guidance from Dr. Christian R. Shelton, my esteemed advisor. I joined Dr. Shelton's research group in September 2004 as a PhD student. Since then he gave me careful guidance for my research directions. Every single process of my work also has his effort to make it happen, and make it better. He is almost always available for helping me when I get stuck. I would like to show my most sincere appreciation to Dr. Shelton. Thank you!

Many thanks goes to my wife, Jing Xu. Also a PhD student in the same group as I, Jing is always willing to discuss problems with me, is always helpful when I need suggestions, and is always forgiving when I make mistakes. More importantly, as my wife, Jing gave me enormous support during my PhD studies.

My thanks also goes to Dr. Eamonn Keogh and Dr. Vassilis Tsotras for their kindness in serving as my defense committee members. They gave great suggestions for making the dissertation better.

I would like to thank Yu Fan, Kevin Horan, Kin Fai Kan, Antony Lam, and Teddy Yap, Jr., the other graduate students in our research group. I have benefitted much from discussions with them, and my PhD life would have been less exciting and fruitful without them.

Finally, I would like to thank my parents in China. We had very little time get together during my time as a PhD student here in UC Riverside. They gave me whole-hearted support, without which my studying would have been much more difficult, if even possible.

ABSTRACT OF THE DISSERTATION

Dimensionality Reduction Algorithms With Applications to Collaborative Data and Images

by

Guobiao Mei

Doctor of Philosophy, Graduate Program in Computer Science
University of California, Riverside, August 2008
Dr. Christian R. Shelton, Chairperson

General dimensionality reduction techniques play important roles in various fields in machine learning. As a well studied problem, many existing algorithms have achieved wide success in specific fields. In this work, we view this problem from a different viewpoint.

We first focuses on collaborative data, which consist of ratings relating two distinct sets of objects: users and items. Much of the work with such data focuses on filtering: predicting unknown ratings for pairs of users and items. In this work, we propose a well-structured Bayesian network to model the collaborative data, and employ loopy belief propogation to estimate parameters of the network and perform filtering tasks. In addition, we are interested in the problem of visualizing the information in the collaborative data. Given all of the ratings, our task is to embed all of the users and items as points in the same Euclidean space. We would like to place users near items that they have rated (or would rate) high, and far away from those they would give low ratings. We pose this problem as a real-valued non-linear Bayesian network and employ Markov chain Monte Carlo and expectation maximization to find an embedding. We present a metric by which to judge the quality of a visualization.

We then extend the visualization framework to images, specifically to embed images. Embedding images into a low dimensional space has a wide range of applications: visualization, clustering, and preprocessing for supervised learning. Traditional dimension reduction algorithms assume that the examples densely populate the manifold. Image databases tend to break this assumption, having isolated islands of similar images instead. Here we extend our framework to achieve the embedding goal of preserving local image similarities based on their scale invariant feature transform (SIFT) vectors. We make no neighborhood assumptions in our embedding. Our algorithm can also embed the images in a discrete grid, useful for many visualization tasks.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

When shopping online, would it not be great if the website could recommend the right items for us? Or if we have rated some movies[1], would it not be helpful if they could give us a graphical display of both movies and people, so that we can navigate easily for more movies we may be interested in, and see what our friends like? Also, consider the situation that we have a collection of personal photographs which are abundant and disordered, would it not be nice if we had an automatic tool to generate a layout of the images so that similar ones lie close to each other, allowing us to navigate through the photos more easily and with more fun?

This dissertation works to provide solutions to the above tasks. We first introduce the problem of collaborative filtering in Section 1.1, then the problem of visualization of collaborative data in Section 1.2, and finally give a brief introduction to image embedding problem

[1]For example *http://www.imdb.com*.

in Section 1.3. In our work, all the three problems are solved with similar frameworks and techniques. They all fall into the more general class of dimensionality reduction techniques.

1.1 Collaborative Filtering

Our work starts with looking into a special kind of data, namely collaborative data. Collaborative data, which are composed of correlated *users* and *items*, are abundant: movie recommendations, music rankings, and book reviews, for example. They can be very useful to make accurate recommendations to users about which items they might favor. This problem is generally recognized as collaborative filtering, and we try to tackle the problem with a novel graphical model based approach.

Collaborative data can be found in many places involving users and items. Here we give a more detailed introduction to them with the example of *BoardGameGeek*[2] (BGG), a well-known board game rating and recommendation website. The first major component of collaborative data are users. In BGG, users refers to the registered accounts with the website. The other component of collaborative data are items: board games in the case of BGG. Without loss of generality, users and items are identified by single integers IDs. They are correlated by the ratings between them. Each user can rate a subset of the items (games) that he or she has played or is familiar with. BGG has integer ratings with 10 the highest score. Figure 1.1 shows sample rating breakdown for the game “Die Macher” as of the date of this dissertation.

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

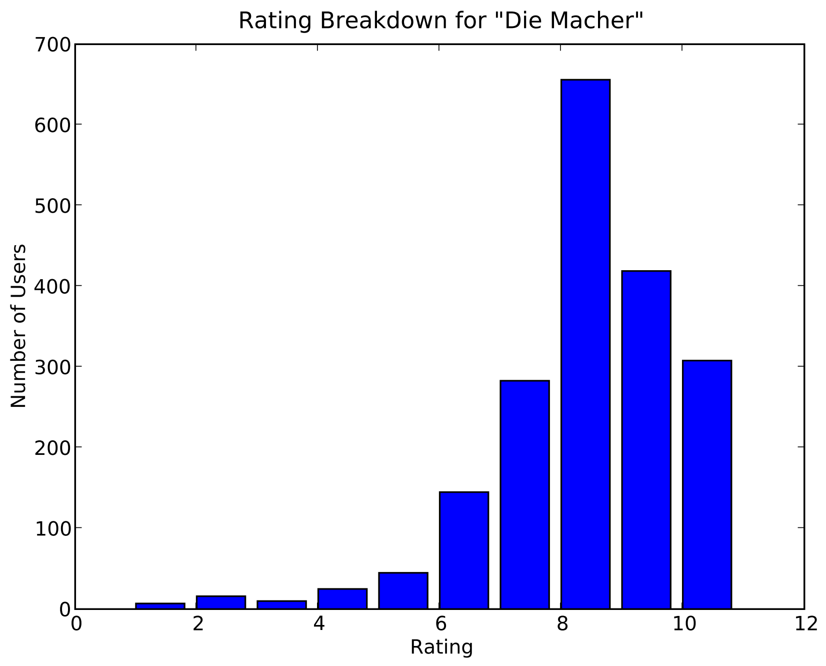


Figure 1.1: Sample rating breakdown for the game “Die Macher” from BGG

Typical collaborative data contain a small set of “popular” items with lots of ratings, and a large subset of items with only a few ratings. Similar trends also happen to the users. In general, the overall rating density (average percentage of games each user has rated) is less than 0.5% in BGG. Figure 1.2 shows the available ratings for top 500 users and top 500 games.

Collaborative filtering is the task of providing useful information of what items people might like or dislike, based on items they have previously rated. Collaborative filtering can help us suggest new items a user might be interested in, help users navigate items according to their preferences, or reconstruct data by predictions. Collaborative filtering is equivalently the task of predicting missing ratings for any user based on his or her existing ratings and

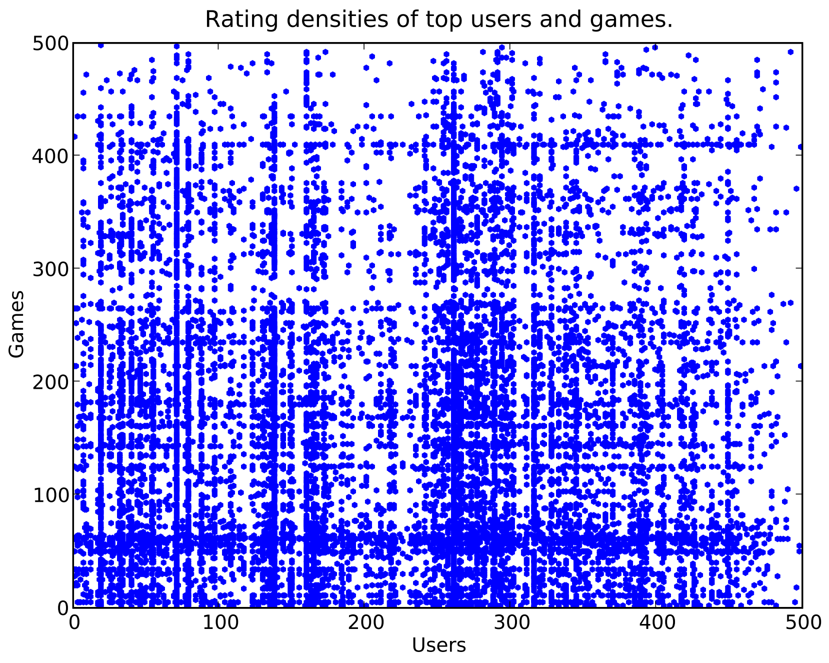


Figure 1.2: Top rated users and games for BGG. Black dots indicate the existence of a rating of the corresponding user and game.

other users' ratings. It is call "collaborative" because for prediction, we not only take into account the users own ratings, but also other users' preferences.

In this work, we try to model collaborative data with Bayesian networks. In the collaborative data, we have m users $\mathbf{U} = \{U_1, U_2, \dots, U_m\}$ and n items $\mathbf{G} = \{G_1, G_2, \dots, G_n\}$. We set $\delta_{ij} = 1$ if there is a rating r_{ij} from U_i to G_j, otherwise set $\delta_{ij} = 0$. We assume that ratings are discrete.

We assume that there is some hidden characteristic variable representing the personal preferences of each user, and similarly some hidden property variable representing the properties of each item. Let u_i be the corresponding variable for user U_i, and likewise g_j is the property variable for item G_j. We further assume that the rating r_{ij} is solely dependent on the latent variables u_i and g_j: once u_i and g_j are given, r_{ij} is independent of other ratings in the data. Figure 1.3 shows a sample problem of collaborative filtering with three users, three items, and binary ratings.

We explain in detail of our approach to employ Bayesian network learning and inference algorithms to perform collaborative filtering in Chapter 3, with some essential Bayesian network knowledge introduced in Appendix A.

1.2 Visualization of Collaborative Data

In addition to predicting missing ratings, we might like to visualize the data. Spatial layouts may potentially increase interest in exploration and to aid in finding information. The visual-

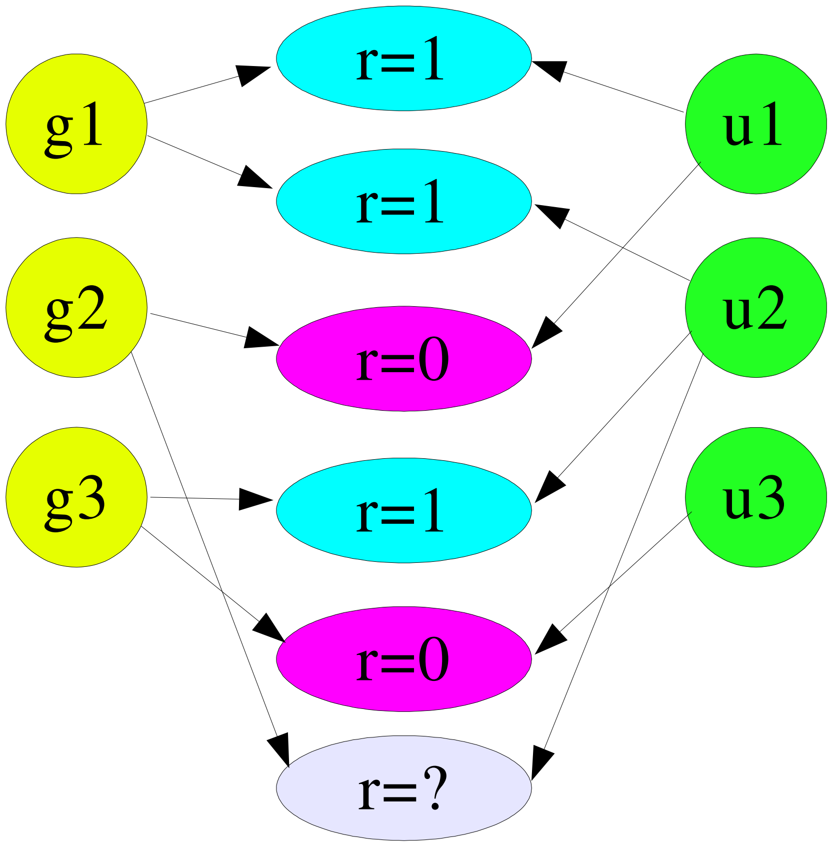


Figure 1.3: Sample collaborative filtering problem with binary ratings.

ization problem of collaborative data is new to the research literature. We extensively explore the approach and analysis for this problem in this work.

Using all the ratings, the visualization problem is to extract the intrinsic similarities or dissimilarities between all the users and items involved, and represent them graphically. This has a wide range of applications, for example guided on-line shopping. Traditional stores allow for easy browsing by physically walking up and down the aisles and visually inspecting the store's contents. Such browsing is not easy on-line. *Amazon.com*, for instance, has thousands of items in many categories. While collaborative filtering allows an on-line seller to recommend a list of objects that the buyer might also like, it does not supply a good way of browsing an on-line collection in a more free-form fashion. We propose building an embedded graph of all the items using the collaborative rating data, and allowing the shopper to zoom in on a portion of the graph and scroll around as he or she searches for items of interest. If constructed well, nearby items will also be of interest to the shopper and local directions in the space will have "meaning" to the user. Spatial layouts have been shown in the past to increase interest in exploration and to aid in finding information [Chennawasin et al., 1999].

We assume a D-dimensional Euclidean space, which we call the *embedded space* (for most computer interfaces, $D = 2$). Each user or item is represented by a point in this space. Intuitively, if two user (or item) points are near each other in the embedded space, the two users (items) are likely to have similar preferences (properties). In the same manner, the closer a user point is to an item point in the embedded space, the higher the rating the user has given (or would give) the item.

No previous algorithms have approached the problem of visualizing collaborative information. Here we initiate this problem and propose an approach. Many visualization problems are “soft” in nature and it is difficult to compare alternative methods. For this task, we introduce a simple evaluation criterion which is natural and allows for numeric comparisons of possible visualizations. We explain the details of our method in Chapter 4.

1.2.1 Connection with Dimensionality Reduction

Our proposed visualization of collaborative data problem imposes a new perspective view for the data. It is essentially a dimensionality reduction problem for the specially organized collaborative data, with possibly missing feature values (ratings), and usually with high dimensionality (total number of users or items).

This also suggests that our framework may well be applicable to general dimensionality reduction tasks. If we convert the raw input data into some type of collaborative data, we can apply our algorithm to perform the embedding. More importantly, our approach not only embeds the input data, it also co-embeds features in the same embedded space.

1.3 Embedding Images

The collaborative visualization framework can be applied to other seemingly unrelated domains, image embedding for an example. By representing images with sets of features, images can be embedded into a Euclidean space with a similar method.

Such image embeddings can have a wide range of applications. Image search engines are one obvious example. Often a good portion of the proposed images for a query are not related to the desired goal of the user. By embedding the images, the user can more quickly find the set of interest. As a consequence, searching results will be much improved. With the abundance of digital photography, many households have thousands of images stored on their home computers without a suitable method for searching them. Graphic visualization of the entire database can be a great aid in allowing quick and easy retrieval of desired photographs. More generally, embedding images into a low dimensional space has a wide range of other applications: visualization, clustering, and pre-processing for supervised learning.

Traditional dimension reduction algorithms assume that the examples densely populate the manifold. Image databases tend to break this assumption, having isolated islands of similar images instead. In this work, we propose a novel approach that embeds images into a low dimensional Euclidean space, while preserving local image similarities based on their scale invariant feature transform (SIFT) vectors, which are first introduced in [Lowe, 2003]. We make no neighborhood assumptions in our embedding. Our algorithm can also embed the images in a discrete grid, useful for many visualization tasks. We demonstrate the algorithm on images with known categories and compare our accuracy favorably to those of competing algorithms.

We modify the framework proposed for visualization of collaborative data to suite for the need of image embedding. The goal is similar to the visualization problem: to put images into some Euclidean space so that each image lies close to similar other images, while far apart from those with large distinctions. It is essentially a dimension reduction problem for images. We give the details of our algorithm in Chapter 5.

1.4 Summary

We propose an approach to solve the collaborative filtering, visualization, and image embedding problems listed above. We give detailed formulation of the problems and present our algorithms in the following chapters.

Chapter 2

Background

In this chapter, we introduce the background of our work, and list some of the related work. We first introduce the problem of dimensionality reduction in Section 2.1. We then show some of the most popular existing algorithms for the problem. In Section 2.2 we introduce the problem of collaborative filtering. We show how this problem is essentially a dimensionality reduction problem with missing data components.

2.1 Dimensionality Reduction Algorithms

Dimensionality reduction (DR) is one of the most widely used problems in the machine learning field. In short, it is the problem of finding low dimensional structure of given high dimensional data.

Formally, assume the input data consists of n points $\{x_1, x_2, \ldots, x_n\}$, each point x_i is a D dimensional vector, *i.e.* $x_i \in \mathbf{R}^D$. The problem of dimension reduction is to find a mapping

ϕ such that $y_i = \phi(x_i)$, where $y_i \in \mathbf{R}^d$ is a d (usually much smaller than D) dimensional vector and y_i best "represents" x_i. We hereafter use $\mathbf{X}$, a $D \times n$ matrix consisting of all the data points as columns to denote the entire input data. Likewise, we use $\mathbf{Y}$, a $d \times n$ matrix to denote the corresponding output.

We also use the term "embedding" to represent the problem of dimensionality reduction, especially for the case of small d. Intuitively, it is the problem of embedding the high dimensional raw data x_i into a low dimensional space where the results are easily viewable. We refer to the target $\mathbf{R}^d$ space as the *embedded space*. Note that it is difficult to present a universal view of what a good embedding is. Different applications will have different ways of interpreting what is a good representation of the original data.

Dimensionality reduction is useful in many domains in the machine learning literature. In supervised learning, for example [Fukumizu et al., 2004], where training data x_i are given along with a label y_i indicating their categories, the task is to train a classifier f such that when a previously unseen testing data point x is given, $f(x)$ predicts the class label. When the data points have many dimensions, it is both time consuming and often inaccurate to train the classifier directly on the raw data. Pre-processing data points with dimensionality reduction algorithms is not only suitable, but also necessary in many cases.

In general, the dimensionality reduction problem only focuses on the relative positioning of the input data. Without loss of generality, in this section we assume that the data points have zero empirical mean. *i.e.* $\sum_i x_i = \mathbf{0}$. This can be easily achieved by computing the empirical mean of given input data points and subtracting it from the original data points:

$x_i \Leftarrow x_i - \frac{1}{n}\sum_j x_j$.

Dimensionality reduction algorithms fall into two categories: linear and non-linear. We briefly discuss both in the remainder of this section.

2.1.1 Linear Algorithms

Linear dimensionality reduction algorithms seek to find a linear transformation for the input data such that some criterion is met. In particular, they try to find a $D \times d$ matrix $\mathbf{W}$ such that $y_i = \phi(x_i) = \mathbf{W}^\top x_i$ is the embedding. Or in other words $\mathbf{Y} = \mathbf{W}^\top \mathbf{X}$.

Principal Component Analysis

Principal component analysis (PCA) is one of the most widely used linear dimensionality reduction algorithm. In 1901 Karl Pearson presented this algorithm. The goal of PCA is to project the data while preserving the greatest variance.

More formally, let w_1 be a unit vector along which to project the original data to. PCA seeks for

$$w_1 = \arg\max_{\|w\|=1} \sum_i (w^\top x_i)^2 \ .$$

Recursively, once we have obtained the first $k-1$ such projection vectors, the k-th projection vector is

$$w_k = \arg\max_{\|w\|=1} \sum_i (w^\top (x_i - \sum_{j=1}^{k-1} w_j w_j^\top x_i))^2 \ .$$

In [Pearson, 1901], the author also showed that the solution for PCA is just an eigenvalue decomposition problem for the covariance matrix of input data $\mathbf{X}$. Let $\mathbf{C} = \mathbf{X}\mathbf{X}^\top$ be the $D \times D$ empirical covariance matrix of $\mathbf{X}$. Let $w_1, w_2, \ldots, w_d$ be the d eigenvectors of $\mathbf{C}$ with largest eigenvalues. Then the solution of PCA for the data X is given by $\mathbf{W} = [w_1, w_2, \ldots, w_d]$. The embedding is given by

$$\mathbf{Y} = \mathbf{W}^\top \mathbf{X} \ .$$

Multi-Dimensional Scaling

Multidimensional scaling (MDS) is another well known linear dimensionality reduction algorithm. It is first introduced in [Torgerson, 1952]. Let $S_{ij} = \|x_i - x_j\|$ be the Euclidean distance between the points x_i and x_j in the original $\mathbf{R}^D$ space, and $T_{ij} = \|y_i - y_j\|$ be the distance in the embedded $\mathbf{R}^d$ space between the two points. A good embedding should preserve the distance well: close points in the original space should remain close in the embedded space, and vice versa.

MDS seeks to minimize the objective function

$$\sum_{i,j} \psi(S_{ij} - T_{ij}) \ .$$

Most of the time the function ψ is set to be the square of the argument. In this case, the objective function of MDS becomes $\sum_{ij}(S_{ij} - T_{ij})^2$.

There is a closed form solution for MDS when ψ is the square function, as shown in

[Cox and Cox, 2001]. Let $\mathbf{P}_{ij} = \|x_i - x_j\|^2$, and define $\mathbf{B} = -\frac{\mathbf{HPH}}{2}$, where $\mathbf{H}_{ij} = \delta_{ij} - \frac{1}{n}$. Here $\delta_{ij} = 0$ unless $i = j$, in which case $\delta_{ij} = 1$. The objective function is minimized if the embedding $\mathbf{Y}$ is set to the top d eigenvectors of the matrix $\mathbf{B}$.

Further research [Williams, 2001] showed that PCA and MDS should produce identical embedding given the same input data if MDS uses the square function for ψ. So we can view MDS as a generalization of PCA.

2.1.2 Nonlinear Algorithms

Linear dimensionality reduction algorithms work well for input data lying in a regular hyperplane. However, in many cases the data points lie in a structured manifold. Applying linear transformations for dimensionality reduction produces a poor representation of the data. We introduce some of the most popular nonlinear embedding algorithms in the following text.

Locally Linear Embedding

In [Roweis and Saul, 2000], the authors present locally linear embedding (LLE). As a nonlinear embedding algorithm, LLE assumes that the nearby points (local neighbors) in the original space should remain neighbors in the embedded space. In particular, the embedding of a point should be able to be linearly reconstructed from its local neighbors (points with smallest Euclidean distances) in the original space.

The LLE algorithm has two phases. The first step is to compute the reconstructing weights for each of the data points by its neighbors. The reconstruction cost to be minimized is defined as

$$\epsilon(\mathbf{W}) = \sum_i |x_i - \sum_j \mathbf{W}_{ij} x_j|^2 \ .$$

They only consider reconstructing x_i from its neighbors, so they enforce $\mathbf{W}_{ij} = 0$ when x_j is not in the neighbor set of x_i. To get rid of the scaling freedom, they further require that $\sum_j \mathbf{W}_{ij} = 1$.

The second phase of the algorithm is to use the local weights $\mathbf{W}$ to generate the embedding $\mathbf{Y}$ such that the following cost function is minimized:

$$\Phi(\mathbf{Y}) = \sum_i |y_i - \sum_j \mathbf{W}_{ij} y_j|^2 \ .$$

To remove the rotational freedom in the final embedding, they further require

$$\begin{aligned} \sum_i y_i &= \mathbf{0} \\ \sum_i [y_i y_i^\top] &= \mathbf{I} \end{aligned}$$

Minimizing both cost functions results in an eigenvector decomposition problem of a gram matrix defined in [Saul and Roweis, 2003]. The computational cost of LLE is similar to that of MDS.

Isomap

In [Tenenbaum et al., 2000] the authors present *Isomap*, another nonlinear embedding algorithm. The idea of Isomap is that the distance between two points in the original space should not just be computed as the Euclidean distance. Instead, it should be the shortest sum of distances along a path consisting of points near each other.

They first construct a connectivity graph $\mathbf{G}$, with nodes $\mathbf{X} = \{x_1, x_2, \ldots, x_n\}$. An edge between x_i and x_j exists if x_j is one of the K nearest neighbors of x_i. Then they compute the distance between any two points x_i and x_j to be the shortest distance of the nodes x_i and x_j in the graph $\mathbf{G}$. Finally they run MDS on the distance defined above.

Semi-Definite Embedding

Semi-definite embedding (SDE), or maximum variance unfolding (MVU) is another popular nonlinear dimensionality reduction technique. It was first presented in
[Weinberger and Saul, 2006].

Similar to Isomap, the idea of SDE is to build a connectivity graph $\mathbf{G}$, with nodes $\mathbf{X} = \{x_1, x_2, \ldots, x_n\}$. They connect each node x_i with its K nearest neighbors. The constraints imposed by SDE preserve the lengths and angles of these edges during embedding. To reduce the complexity of having to dealing with the angles, they further add an edge between any two neighbors of a point x_i, if it did not already exist.

Similar to LLE, to produce a unique embedding, they further require the embedding result to be centered at origin: $\sum_i y_i = \mathbf{0}$. The constraint-preserving embedding leads to a semi-definite programming (SDP) problem and can be solved with many existing SDP solvers.

2.1.3 Other Dimensionality Reduction Algorithms

Dimensionality reduction has been researched and developed for a long time, and there are many other algorithms available.

In [Scholkopf et al., 1998], the authors introduced Kernal PCA, which uses a Kernel function (matrix) to implicitly map raw data vectors into some feature space and then perform PCA there. The resulting embedding is thus a non-linear projection of the data. In [Donoho and Grimes, 2003], the authors propose Hessian LLE (hLLE). hLLE adapts the LLE method but adjust the reconstruction weight $\mathbf{W}$ to minimize the Hessian operator. This method is designed for some non-convex data sets.

2.2 Collaborative Filtering Algorithms

There are many existing collaborative filtering algorithms focusing on the task of prediction. [Breese et al., 1998] classifies the approaches into two major categories: *memory based* and *model based*. Memory based collaborative filtering algorithms make predictions according to all the existing preferences stored beforehand, while model based algorithms first try to learn the parameters of a particular model for the existing user preferences, and then make predictions according to the learned model. [Sarwar et al., 2001] has an exploration of many item-based algorithms.

2.2.1 Eigentaste

[Goldberg et al., 2001] propose Eigentaste (ET). It has two phases: offline and online. It treats the entire rating matrix as a high dimensional space and employs principal component analysis for dimensionality reduction during the offline phase. It projects the users into a low dimensional space and then partitions the embedded space into sets of users and uses the maximally rated items in a given set as predictions during the online phase. While this algorithm does place the users in a geometric space, it does not place the items in the same space, and it requires that there be a set of items (the *gauge set*) which every user has rated. This is a severe restriction because not all collaborative data will have this property.

2.2.2 Co-occurrence Data Embedding

Co-occurrence data have been used to produce embeddings of two classes of objects in the same space. CODE [Globerson et al., 2005] is one such recent example. It tries to embed objects of two types into the same Euclidean space based on their co-occurrence statistics. Unfortunately, collaborative filtering data are usually given as ratings and not co-occurrence statistics. Even if we take the ratings to be proportional to the corresponding co-occurrences (an unjustified assumption), we still have missing statistics (which cannot be taken to be zero). Co-occurrence algorithms do not currently deal with such missing values.

2.2.3 Other Collaborative Filtering Algorithms

Collaborative filtering has attracted a good amount of research in the past years. Many other algorithms exist in the literature.

In [Hofmann and Puzicha, 1999], the authors use a probabilistic latent space model to model users' preferences as convex combinations of preference factors, and employ approximate EM algorithm to learn the model.

Some of the filtering methods have a “geometric” flavor. [Pennock et al., 2000] propose a collaborative filter based on personality diagnosis. They associate each user with a vector R^{true}, indicating the true rating of this user for every item in the system. The actual rating is assumed to be a random variable drawn from a Gaussian distribution with mean equals to the corresponding element of that item in the user's R^{true} vector. If there is a missing rating, the corresponding R^{true} element is a uniform random variable.

In [Pavlov and Pennock, 2002] the authors developed a maximum entropy approach for collaborative filtering. The algorithm is especially suitable for dynamic stream collaborative data. They view the items as clusters based on the user access patterns, and make predictions that minimizes the probability of crossing cluster boundaries.

[Melville et al., 2002] presented content-based collaborative filtering that incorporates components from both content-based methods for recommendation systems and collaborative filtering algorithms. They introduce an effective framework for performing high quality recommendations. They first use a content-based predictor to enhance the user data, and then provide recommendations through other collaborative filtering algorithms.

[Hofmann, 2004] introduced a model-based algorithm which relies on a statistical mixture model involving latent class variables. They gave a constant time prediction approach once the model for discovering user communities and prototypical interest profiles is learned.

In [Miller et al., 2004], the authors presented PocketLens, a distributed collaborative filtering algorithm that overcomes the limitation of traditional algorithms such as non-portable and user privacy issues. PocketLens is designed to run in a peer-to-peer environment.

One major problem for collaborative filtering is the sparsity in the data. Often the ratings in the collaborative data are insufficient to identify similarities in users and items. [Huang et al., 2004] proposed an algorithm that makes use of an associative retrieval framework to overcome the sparsity limitation. The basic idea of associative retrieval is to build a graph or network model of users and items for the collaborative data, by exploring the transitive associations among the users and items, information of improved quality can be retrieved through this graph model.

Similar to principal component analysis (PCA) we discussed in the previous section, maximum margin matrix factorization (MMMF) algorithms are also widely used as a dimensionality reduction algorithm, and it can be extended for collaborative filtering problems. Traditional MMMF algorithms use a semi-definite programming (SDP) solvers to handle problems on matrices of dimensionality up to a few hundred. In [Rennie and Srebro, 2005] the authors investigated a gradient-based optimization method for MMMF and applied the algorithm on large (sparse) collaborative data and achieved good empirical accuracy.

[Leung et al., 2006] presented a collaborative filtering framework based on fuzzy association rules and multiple-level similarity (FARAMS). FARAMS takes advantages of product similarities in taxonomies to handle the sparsity of the collaborative data, and use fuzzy association rule mining to improve competitive prediction quality. In [Leung et al., 2007] they presented cross-level association rules (CLARE) to address the cold-start problem (new user has to make many ratings before filtering is effective) in collaborative filtering.

In [Banerjee et al., 2007], the authors presented a partition co-clustering[1] formulation for matrix approximation. [George and Merugu, 2005] applied the same framework to the problem of collaborative filtering. They designed incremental and parallel versions of the co-clustering algorithm and applied on users and items in the collaborative data. The algorithm they presented is efficient and suitable for real-time dynamic evolving data, with comparable accuracy but lower computational cost.

In summary, as a problem with wide applications, collaborative filtering has been studied for long. However, none of the listed algorithm directly use the same Bayesian network structure to model the entire data. We propose our framework with experiments to show empirical results in Chapter 3.

[1]Co-clustering is the technique of clustering both rows and columns of two-dimensional data matrices.

Chapter 3

Collaborative Filtering

In this chapter, we present an approach to the well studied collaborative filtering problem.

We use a large scale Bayesian network [Pearl, 1988], a popular graphical model framework, to model the collaborative data. In our framework, users and items in the collaborative data are represented by latent characteristic values, which correspond to two sets of nodes in the graphical model. Ratings are viewed as observed evidence.

This chapter is organized as following. We first show how to model collaborative data as a complex Bayesian network in Section 3.1. We then demonstrate how to estimate the parameters using approximate inference algorithms in Section 3.3. In Section 3.2 we show how to perform collaborative filtering once we have the graphical model and the learned parameters. We show how this model can be further extended with additional information in Section 3.5.

3.1 Modeling Collaborative Data

A Bayesian network is one of the most popular graphical models to describe a joint distribution over a set of random variables. The conditional independencies between the variables are compactly encoded by the structure of the network, and the joint distribution can be factored, most of the time, compactly into the product of the set of factors associated with the variables. We give more introduction on Bayesian networks in Appendix A.

We have m users $\mathbf{U} = \{U_1, U_2, \ldots, U_m\}$ and n items $\mathbf{G} = \{G_1, G_2, \ldots, G_n\}$. We set $\delta_{ij} = 1$ if there is a rating r_{ij} from U_i to G_j, otherwise set $\delta_{ij} = 0$. We assume that ratings are discrete. We further assume that there is a hidden characteristic variable representing the personal preferences of each user, and similarly a hidden property variable representing the properties of each item. Let u_i be the corresponding variable for user U_i, and likewise g_j is the property variable for item G_j. We further assume that the rating r_{ij} is solely dependent on the latent variables u_i and g_j: once u_i and g_j are given, r_{ij} is independent of other ratings in the data.

It is natural to model the collaborative data as a Bayesian network. For each user U_i, we add it as a node in the Bayesian network structure $\mathcal{G}$, and similarly for each item G_j. For each existing rating r_{ij}, *i.e.* $\delta_{ij} = 1$, we add a node R_{ij} into $\mathbf{G}$, and make U_i and G_j be the parents of R_{ij}. A sample Bayesian network structure for 3 users and 2 items with 5 ratings is shown in Figure 3.1.

Eventually we get a very large Bayesian network structure with all the users, items and

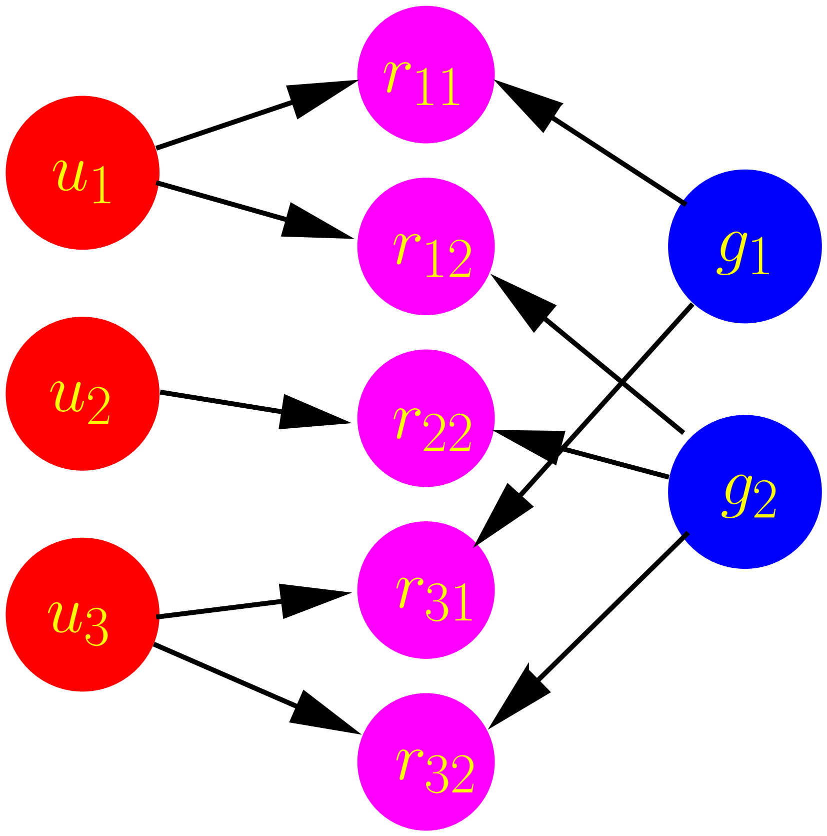


Figure 3.1: Sample Bayesian network structure for collaborative filtering.

ratings as nodes, where each rating node has exactly two parents indicating the corresponding user and item for this rating.

3.1.1 Model Explanations

The proposed model structure assumes the users and items are independent before the observing the ratings. We also assume that the posterior joint distribution of users and items given all the ratings can be factored into production of marginal posteriors of users and items. We explain this in detail in Section 3.3. The distribution of a rating should only depend on the latent properties of the corresponding item and the preference of the user. The entire structure correlates the distribution of these latent user or item properties via the set of rating observations.

To take a concrete example, let $\mathbf{U}$ be a set of game players, and $\mathbf{G}$ be a set of games. Assume that the latent variable u and g are both binary. Assume that $u_i = 0$ means the user U_i likes two-player games and $u = 1$ means the user likes multi-player games. Similarly, assume that $g_j = 0$ means the game G_j is a two-player game, and $g_j = 1$ means it's a multi-player game. Assume that the ratings are also binary variables: 0 means "dislike", and 1 means "like". It is often the case that u_i will tend to rate g_j higher if they "match". Hence one reasonable conditional probability distribution (CPD) for this family is shown in Table 3.1.

U_i	G_j	$R_{ij} = 0$	$R_{ij} = 1$
0	0	0.1	0.9
0	1	0.9	0.1
1	0	0.8	0.2
1	1	0.05	0.95

Table 3.1: Sample CPD for collaborative filtering problem.

3.1.2 Parameter Tying

It is reasonable to make the assumption that the CPD for the family of R_{ij} is the same for all i and j. In another words, all the CPDs for $P(R_{ij} \mid U_i, G_j)$ share the same parameters throughout the entire Bayesian network. For example, as long as someone likes multi-player games, and some game is a two-player one, then he will probably not like the game. There is no need to distinguish who someone is, or which game it is by analogy. Similarly, we make the assumptions that the priors $P(U_i)$ for all users are also shared with the same parameters, and the same for $P(G_j)$.

The characteristics of parameter tying is essential for estimating in this type of models, with complex but well formed structures. The entire data is essentially one instance for all the rating variables for this huge Bayesian network. There is no way to learn each individual CPDs of ratings or prior distribution of user (item) preferences. Once we tie all the corresponding parameters, we have a more compact way of deriving sufficient statistics of individual *user-rating-item* families and estimate the global CPD as an averaged conditional probability of all the rating families. We give more explanation on this in Section 3.3.

Given the Bayesian network structure constructed in this way, the only parameters we

Parameter	Meaning
θ_u	Prior for all u_i
θ_g	Prior for all g_j
$\theta_{r\|u,g}$	Conditional probability of rating r_{ij} given u_i and g_j

Table 3.2: Parameters for the Bayesian network structure.

need consist of parameters in Table 3.2. We denote them by $\mathbf{\Theta}$.

3.2 Collaborative Filtering

One of the key problems in collaborative filtering is predicting missing ratings. We denote $\mathbf{R} = \{R_{ij} \mid \delta_{ij} = 1\}$ to be the set of all existing ratings. The filtering problem in our framework is essentially the query problem:

$$P(r_{ij} \mid \mathbf{R}), \text{where } \delta_{ij} = 0. \tag{3.1}$$

Once we have the parameters $\mathbf{\Theta}$, we can use inference algorithms to obtain the posteriors $P(u_i \mid \mathbf{R})$ and $P(g_j \mid \mathbf{R})$ for all users and items. Additionally, from the structure of the Bayesian network model, r_{ij} is conditionally independent of all other variables given u_i and g_j. Equation 3.1 can thus be rewritten as

$$P(r_{ij} \mid \mathbf{R}) = \sum_{u_i,g_j} P(r_{ij} \mid u_i, g_j) P(u_i, g_j \mid \mathbf{R}) \approx \sum_{u_i,g_j} \theta_{r|u_i,g_j} P(u_i \mid \mathbf{R}) P(g_j \mid \mathbf{R}) \ .$$

The joint posterior $P(u_i, g_j \mid \mathbf{R})$ is very difficult to compute, and we use loopy belief propagation (LBP) [Murphy et al., 1999] to compute its approximation represented by the last approximation in Equation 3.2. Detailed explanation of this is in Section 3.3.2. Any prediction task becomes simply a look-up in the posteriors of users and items, and a multiplication with the CPD $\theta_{r|u,g}$.

3.3 Parameter Estimation

The structure of the Bayesian network is given by our problem setting, and the parameter estimation problem is to find

$$\Theta^* = \arg\max_{\Theta} P(\Theta \mid \mathbf{R}).$$

In the case that the parameters have uniform priors, this estimation is the same as maximum likelihood. The maximum likelihood estimation is given as

$$\Theta^* = \arg\max_{\Theta} P(\mathbf{R} \mid \Theta). \tag{3.2}$$

The likelihood function in the above equation can further factor into the product of individual $\theta_{r|u,g}$ parameters and the posteriors $P(u_i \mid \mathbf{R}; \Theta)$ and $P(g_j \mid \mathbf{R}; \Theta)$. We will omit Θ in the future equations.

$$P(\mathbf{R}) = \sum_i \sum_j \prod_i P(u_i \mid \mathbf{R}) \prod_j P(g_j \mid \mathbf{R}) \prod_{i,j:\delta_{ij}=1} P(r_{ij} \mid u_i, g_j). \tag{3.3}$$

ESS	meaning
$\bar{M}_u[x]$	expected number of times that $u_i = x$
$\bar{M}_g[y]$	expected number of times that $g_j = y$
$\bar{M}_{r\|u,g}[z, x, y]$	expected number of times that $R_{ij} = z$ when $u_i = x$ and $g_j = y$

Table 3.3: Expected sufficient statistics for collaborative data.

3.3.1 Expectation Maximization for Collaborative Data

In the case of collaborative data, we only observe ratings, but never the latent user or item properties. As a standard way of learning parameters from partially observed data, we use the expectation maximization (EM) algorithm [Dempster et al., 1977] to estimate the parameters.

In our problem, we will use loopy belief propagation as the inference method to be used in the "E step". We will discuss more about this in the next section. To estimate the parameters θ_u, θ_g and $\theta_{r|u,g}$, the expected sufficient statistics (ESS) we need are listed in Table 3.3.

$\bar{M}_u$ is a vector of components $\bar{M}_u[x]$ of all latent user characteristic x. $\bar{M}_u[x]$ is the total expected number of times in all users whose characteristic is x. Similar definitions hold for $\bar{M}_g[y]$ and $\bar{M}_{r|u,g}[z, x, y]$.

$$\begin{aligned}
\bar{M}_u[x] &= \sum_i P(u_i = x \mid \mathbf{R}) \\
\bar{M}_g[y] &= \sum_j P(g_j = y \mid \mathbf{R}) \\
\bar{M}_{r|u,g}[z, x, y] &= \sum_i \sum_j P(r = z, u = x, g = y \mid \mathbf{R})
\end{aligned}$$

The overall learning algorithm is shown in Algorithm 1.

Algorithm 1 $\Theta \Leftarrow$EM($\mathbf{R}, EMiter, LBPiter$)

```
**Inputs:** $\mathbf{R}$: collaborative data, $EMiter$: EM iterations, $LBPiter$: LBP iterations
**Outputs** $\Theta$: learned parameters

    // Set initial parameters.
For all $u_i$, $g_j$ and $i, j$ s.t. $\delta_{ij} = 1$:
Set $\theta_u = Dirichlet(1, 1, \dots)$
Set $\theta_g = Dirichlet(1, 1, \dots)$
for $z \in \{$ all possible ratings $\}$ do
    Set $\theta_{r|u,g}[r = z] \sim Dirichlet(1, 1, \dots)$
end for
for $k = 1$ to $EMiter$ do
        // E step
    $[\mathbf{U}, \mathbf{G}] \Leftarrow$ LBP($\mathbf{R}, \Theta, LBPiter$)
    $\bar{M}_u[x] \Leftarrow \sum_i u_i[x]$
    $\bar{M}_g[y] \Leftarrow \sum_j g_j[y]$
        // This depends on the loopy assumption that $P(u, g \mid \mathbf{R}) \approx P(u \mid \mathbf{R}) P(g \mid \mathbf{R})$
    $\bar{M}_{r|u,g}[z, x, y] \Leftarrow \sum_{i,j:r_{ij}=z} u_i[x] \cdot g_j[y]$
        // M step
    Update $\theta_u[x] \Leftarrow \bar{M}_u[x] / \sum_k \bar{M}_u[k]$ for all $x$
    Update $\theta_g[y] \Leftarrow \bar{M}_g[y] / \sum_k \bar{M}_g[k]$ for all $y$
    Update $\theta_{r|u,g} \Leftarrow \bar{M}_{r|u,g}[z, u, g] / \sum_k \bar{M}_{r|u,g}[k, u, g]$ for all $z, u, g$
end for
Return $\Theta = \{\theta_u, \theta_g, \theta_{r|u,g}\}$.
```

3.3.2 Loopy Belief Propagation

The core algorithm in the expectation step is inference. We need to compute the posterior $P(u_i \mid \mathbf{R})$ and $P(g_j \mid \mathbf{R})$ for all u_i and g_j.

Exact inference algorithms are intractable here due to the complex structure. We choose loopy belief propagation (LBP) [Murphy et al., 1999] here to perform the inference for several reasons: first, when LBP has converged, we can get posterior distribution for all users and items together. Second, LBP is faster in comparison with sampling methods for our problem. Detailed information on LBP and related factor manipulation algorithms is shown in Appendix A.

We construct the cluster graph as following: keep the original Bayesian network structure, and turn the directed arcs into undirected, and then change the R_{ij} nodes into U_i, G_j compound clusters. The initial potentials for the U_i and G_j clusters are all set to θ_u and θ_g, respectively. The potential for U_i, G_j compound cluster is set to $P(u_i, g_j \mid r_{ij}) \propto P(r_{ij} \mid P(u_i, g_j)P(u_i)P(g_j)$. Message passing follows the directions of either "converging" or "dispatching". The detailed algorithm is shown in Algorithm 2.

3.4 Experiments

We test our algorithm on some real world collaborative data, and compare the prediction accuracy and regression errors with other popular methods.

Algorithm 2 $[\mathbf{U},\mathbf{G}] \Leftarrow$ LBP($\mathbf{R},\Theta,LBPiter$)

```
Inputs: $\mathbf{R}$: collaborative data, $\Theta$: model parameters, $LBPiter$: maximum number of iterations
Outputs $\mathbf{U},\mathbf{G}$: the posterior of $P(u_i \mid \mathbf{R})$ and $P(g_j \mid \mathbf{R})$ for all $u_i$ and $g_j$

    // Construct the cluster graph.
For all $u_i$, $g_j$ and $i,j$ s.t. $\delta_{ij} = 1$:
Set factor $\pi_{u_i} = \theta_u$, $\mu_{u_i} = 1$
Set factor $\pi_{g_j} = \theta_g$, $\mu_{g_j} = 1$
Set factor $\pi_{u_i,g_j} = \frac{1}{Z}\theta_{r|u,g}[r = r_{ij}]\theta_u\theta_g$, where $Z = \sum_u\sum_g \theta_{r|u,g}[r = r_{ij}]\theta_u\theta_g$
    // Belief propagation.
for $k = 1$ to $LBPiter$ do
  For all $u_i$, $g_j$ and $i,j$ s.t. $\delta_{ij} = 1$:
      // Converging direction.
  Compute $\psi_{u_i} = \pi_{u_i}/\mu_{u_i}$
  Compute $\psi_{g_j} = \pi_{g_j}/\mu_{g_j}$
  Update $\mu_{u_i} = \pi_{u_i}$, $\mu_{g_j} = \pi_{g_j}$
  Update $\pi_{u_i,g_j} = \pi_{u_i,g_j} \cdot \psi_{u_i} \cdot \psi_{g_j}$
      // Dispatching direction.
  Compute $\psi_{u_i} = \sum_g \pi_{u_i,g_j}/\mu_{u_i}$
  Compute $\psi_{g_j} = \sum_u \pi_{u_i,g_j}/\mu_{g_j}$
  Update $\mu_{u_i} = \sum_g \pi_{u_i,g_j}$, $\mu_{g_j} = \sum_u \pi_{u_i,g_j}$
  Update $\pi_{u_i} = \pi_{u_i} \cdot \psi_{u_i}$
  Update $\pi_{g_j} = \pi_{g_j} \cdot \psi_{g_j}$
end for
Return $\mathbf{U} = \{\pi_{u_i}\}$ for all $u_i$, $\mathbf{G} = \{\pi_{g_j}\}$ for all $g_j$.
```

3.4.1 Experiments Setup

To test the performance of our graphical model representation on collaborative data, we use the real world data set BGG from *http://BoardGameGeek.com/*. The BGG data contains 10538 users and 14333 board games. Ratings are sparse, only 642847 exist (0.45%). All the ratings are integers from 0 to 10.

We use cross validation to test the prediction accuracy. For collaborative filtering, it is not a trivial problem to split the data for training and testing. Unlike traditional tasks, for collaborative filtering, predicting a missing rating for any user inevitably involves retrieving other users' ratings as well as his or her own ratings. Simply splitting entire data into training and testing parts based on user and item IDs will not work for filtering: we will have no information to predict ratings for new items in the testing set.

Instead, we use the testing structure as shown in Figure 3.2. We need a subset of the data for learning the parameters for the model. In our settings, we choose a subset of users $\tilde{U}$ and games $\tilde{G}$ to hide from training (*i.e.* we train on all the ratings except those from $\tilde{U}$ and to $\tilde{G}$). For testing, we need further information from some of the hided ratings. In Figure 3.2, we learn $\mathbf{\Theta}$ from C. We then construct a new Bayesian network over the ratings in B with the parameters learned. Note that the ratings in A are always hidden. After running LBP on the newly constructed Bayesian network, we have the marginal distribution of all the users and items. We then use these quantities along with the learned $\theta_{r|u,g}$ parameters learned from C to perform filtering tasks for the ratings in A.

We first test with binary predictions. All ratings are converted to either 0 or 1, based on

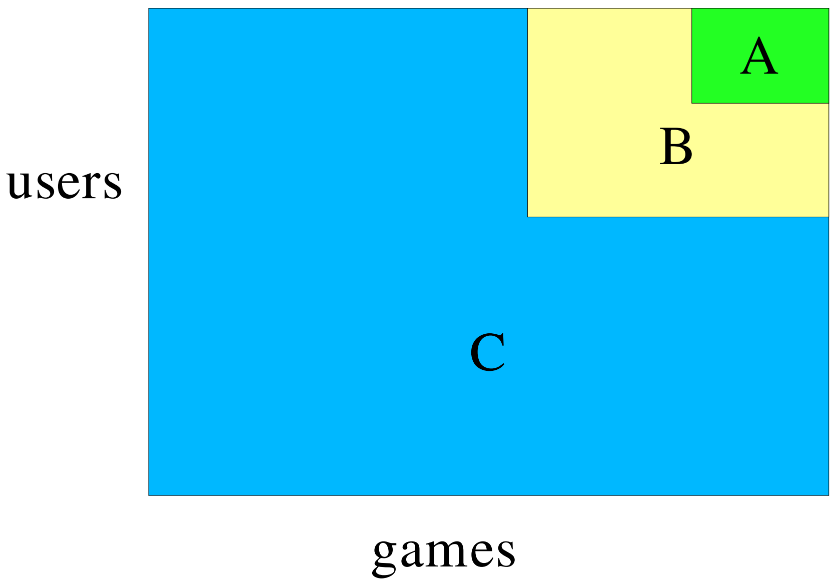


Figure 3.2: Learning and testing structure.

the average rating of each user. For each user, we compute the mean of all his ratings, and convert any rating above or equal to that value to 1, and others to 0. Prediction accuracy is used to evaluate the performance.

Root Mean Square Error

We use root mean square error (RMSE) as the metric to evaluate the prediction quality.

$$RMSE = \sqrt{\frac{\sum_{i,j:u_i,g_j \in \text{Part} A} \sum_k P(r_{ij} = k \mid \mathbf{R_B}; \Theta)(k - R_{ij})^2}{\sum_{i,j:u_i,g_j \in \text{Part} A} 1}} \tag{3.4}$$

Experimental Data Sets

For part C, where we learn parameters $\mathbf{\Theta}$ from, we construct a subset of data with 500 users and 500 games with highest rating densities. We randomly pick 50 users and 50 items to form part B for the testing phase. Among which we further randomly pick 20 users and 20 items to form part C, the ratings of which are used for the final cross validation accuracy.

3.4.2 Results

Figure 3.3 shows the prediction accuracy of part B and A together as a function of the number of EM iterations. The accuracy tends to converge after around iteration 12, so we do not show the entire figure.

Figure 3.4 shows the testing accuracy against linear regression results and a constant prediction base line. The results are averaged across 10 independent experiments with the same settings[1].

The constant predicting base line for each experiment is just to predict all 0 or all 1 whichever is more frequent in the data. The linear regression method is shown in detail in the next section.

Linear Regression

We first fill each missing rating r_{ij} with average rating of u_i. This results in full rating matrix. To predict the missing rating r_{ij} using linear regression, for each item g_k, let $\mathbf{x}_k = [r_{1k}, \ldots, r_{(i-1)k}, r_{(i+1)k}, \ldots, r_{mk}]^\top$, *i.e.* $\mathbf{x}_k$ is the vector containing all the ratings for g_k except

[1]With different randomly chosen users and items in part B and C.

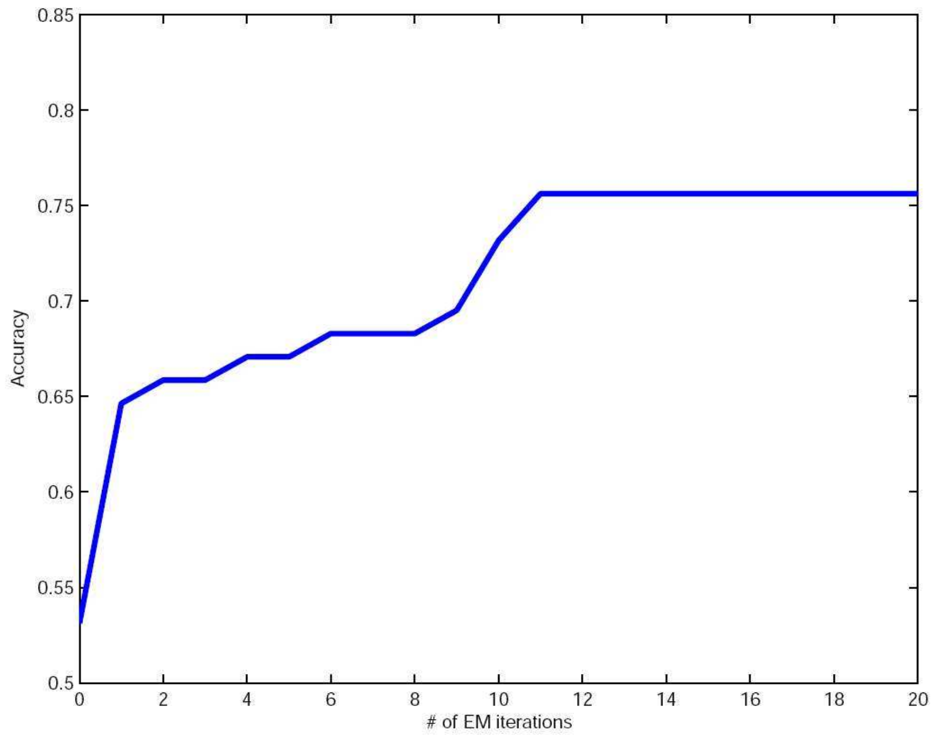


Figure 3.3: Binary prediction accuracy of B+A against EM iterations.

from u_i. We then use linear regression to find

$$\left[\hat{\mathbf{w}}_i, \hat{b}_i\right] = \arg\min_{\mathbf{w},b} \sum_k (\mathbf{w}^\top \mathbf{x}_k + b - r_{ik})^2$$

The predicted rating is then given by

$$\hat{r}_{ij} = \hat{\mathbf{w}}_i^\top \mathbf{x}_j + \hat{b}_i \ .$$

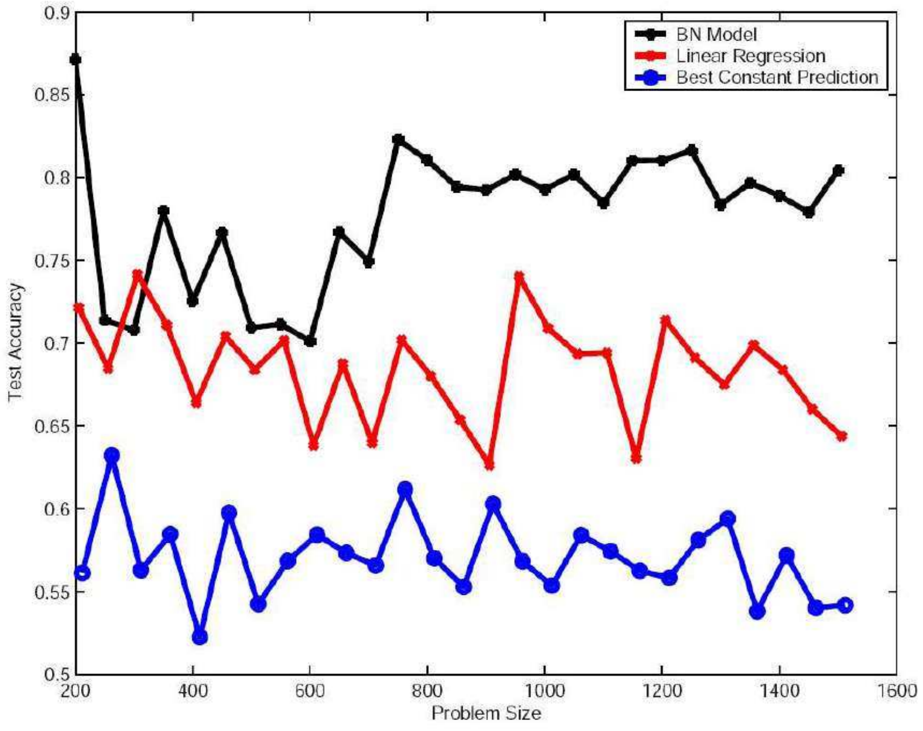


Figure 3.4: Testing accuracy of Bayesian network, linear regression, and constant prediction.

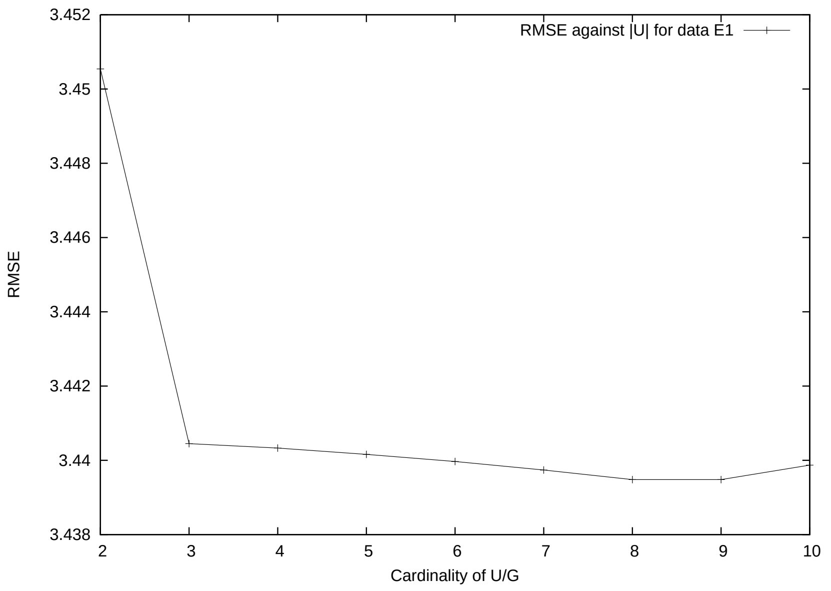


Figure 3.5: RMSE of testing Part A against cardinality of U and G.

Cardinality of the Latent Vectors

For binary prediction tasks, it turns out that the cardinality of u and g is not very important in terms of the final prediction results. Here we set both of them to 2.

We use the parameters learned from part C at EM iteration 100. We randomly pick 20 user IDs and 20 game IDs within the covered part B region to form part A. We then predict all the ratings for part A using the learned parameters and other ratings given in part B. For each of the experiments, we ran 10 independent experiments and report the average errors. We set the cardinality of both u and g from 2 to 10.

We then tested the RMSE as we change the cardinality of U and G from 2 to 10. Figure 3.5 shows the result. It is clear from the results that as long as the cardinality of U and G exceeds 2, it does not affect much to the final testing RMSE.

3.5 Model Extension

It is often the case that additional information, such as text description of the items, is available along with the ratings. The BGG data set comes with game descriptions for most of the games. It will be helpful if we can make use of the information to boost the filtering accuracy. With the Bayesian network model, we can easily extend this model to incorporate additional information. We make this idea concrete through the example of text usage.

3.5.1 Model Setup

To incorporate text information into our Bayesian network model, we use the standard "bag-of-words" model [Lewis, 1998] from text retrieval.

We choose the K most useful words to use in the model: $W_1, W_2, \ldots, W_K$. We will discuss how to choose these words later. We then define binary random variables w_{jk} for all G_j and W_k such that $w_{jk} = 1$ iff W_k appears in the description of G_j. These are sub-units of the Bayesian network shown as following in Figure 3.6. The entire Bayesian network looks like Figure 3.7.

We add additional parameters θ_w. Denote $\theta_w[g] = [P(w_1 \mid g), P(w_2 \mid g), \ldots, P(w_K \mid g)]$. All existing learning and inference algorithms still work in a similar way. For example, during loopy belief propagation, we need to additional pass the messages between word indicators and the game nodes. In comparison to the LBP method discussed in previous section, we just add an additional message passing procudure between the w nodes and their

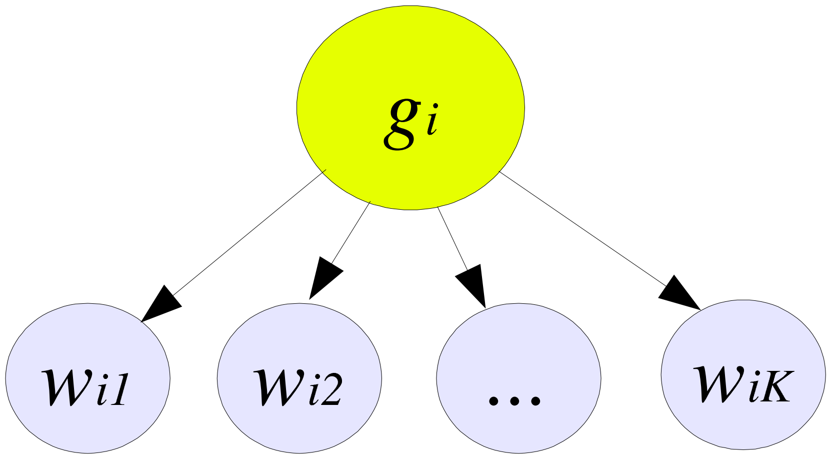


Figure 3.6: Bayesian network with text nodes.

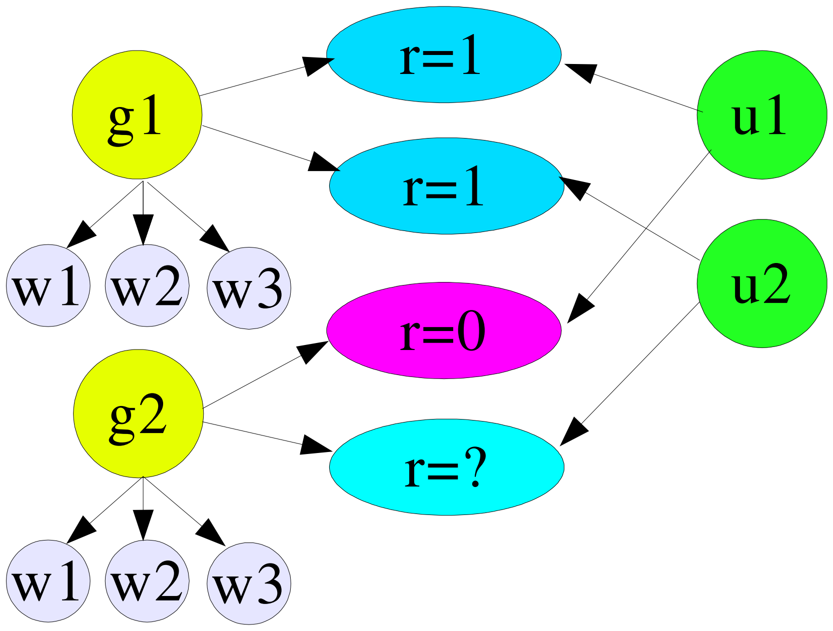


Figure 3.7: Bayesian network with text nodes.

corresponding g nodes in the Bayesian network structure.

3.5.2 Word Selections

There are too many distinct words in the whole text corpus for the Bayesian network model. In the BGG data, the number of different words is around 20000 even after applying the Porter stemming algorithm [Porter, 1980]. It is not only computationally expensive to incorporate all of them into the Bayesian network model, but may also lead to over-fitting.

Neither the most frequent words nor the least frequent words are representative enough to be chosen as key words. We chose the words with the highest mutual information (MI) [Shannon, 1948] with the game type g:

$$MI(w_k, g) = -E[\log P(w_k)] - E[\log P(g)] + E[\log P(w_k, g)].$$

The mutual information between two distributions P and Q indicates how much we know about Q once we observe a sample from P. Here in this problem, intuitively we want to choose the words that give us maximal information about the games.

Once we have the belief propagation results without the text, we have access to an initial posterior distribution of the games $P(g \mid \mathbf{R})$. Because the word indicators are also given as evidence, it is straightforward to compute the expected entropy between $P(g \mid \mathbf{R})$ and $P(w_k)$. Figure 3.8 shows the most frequent words (left) and words with the largest mutual information (right). It is clear that the words with higher mutual information are more meaningful for distinguishing game types.

3.5.3 Results

We ran the same binary prediction experiments as described in the previous section, except that now we added in the game description and extended the Bayesian network model accordingly. The results are shown in Figure 3.9.

The result shows that with the text information incorporated into the Bayesian network model, we achieve higher prediction accuracy.

Word Frequncy Order	Mutual Information Order
the	game
win	player
have	ar
most	thi
will	each
all	card
but	their
elect	can
point	which
after	board
of	your
in	us
differ	first
region	all
player	other
ar	have
a	move
game	two
about	win
seven	turn
polit	ha
sequenti	up
is	point

Figure 3.8: Most frequent words in BGG (left) and words with highest mutual information (right).

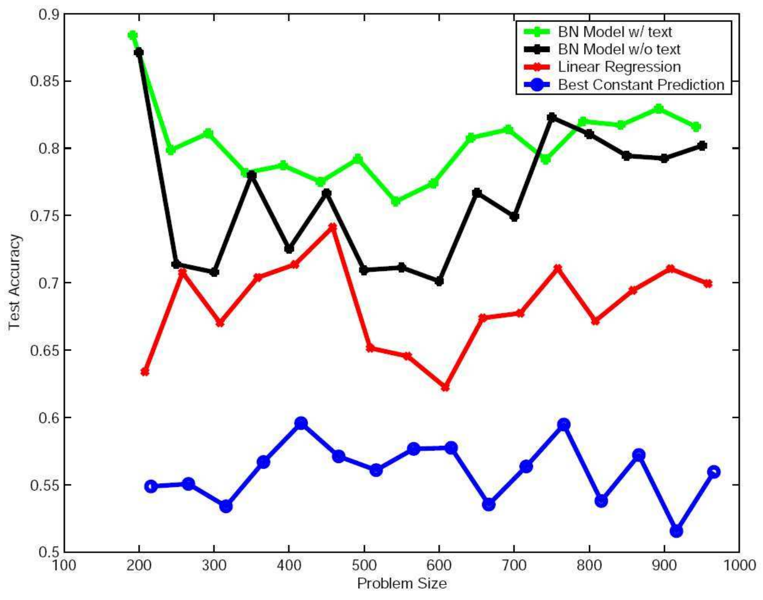


Figure 3.9: Binary prediction accuracy of Bayesian network model with and without text information, along with linear regression and constant prediction results.

Chapter 4

Visualization of Collaborative Data

The filtering problem, as discussed in Chapter 3, obviously plays an important roll in manipulating collaborative data. Much of the prior work has been in the area of collaborative filtering. However, no previous algorithms have approached the problem of visualizing collaborative information. Here we initiate this problem and propose an approach.

4.1 Problem Formulation

We follow some of the notation conventions of the previous chapter. Let $U = \{u_1, u_2, \ldots, u_m\}$ and $G = \{g_1, g_2, \ldots, g_n\}$ be the sets of all users and items, respectively. Without ambiguity, we will use u_i to refer both to the i-th user and to the corresponding point in the embedded space for that user. We use the same notation for g_j. We define δ to indicate whether a user has rated an item.

$$\delta_{ij} = \begin{cases} 1 & \text{if } u_i \text{ rated } g_j \\ 0 & \text{the rating of } u_i \text{ of } g_j \text{ is not available} \end{cases}$$

Let r_{ij} be the rating of u_i of g_j if $\delta_{ij} = 1$. Here we normalize all the ratings to the range $[0, 1]$ (*i.e.* 1 is the highest rating, and 0 is the lowest). Let $R = \{r_{ij} \mid \delta_{ij} = 1\}$. We further denote $G^i = \{g_j \mid \delta_{ij} = 1\}$ and $U^j = \{u_i \mid \delta_{ij} = 1\}$. G^i is the set of all the items that u_i rated, and U^j is the set of all the users that rated g_j.

The visualization problem is to find an embedding of all the points U and G in a Euclidean space we call the *embedded space*. The embedding should be one in which the distance between a user and an item is related to the corresponding rating.

We construct a Bayesian network for the collaborative with the same structure as we used in Chapter 3: all the users, items and existing ratings are represented as nodes in the network, and each rating node has the corresponding user and item nodes as its parents in the structure of the Bayesian network. The difference is that, for the task of visualization, each user or item node corresponds to the embedded position of the point, and hence is a real-value multivariate random variable. Also, the conditional probability distribution, in analogy to $\theta_{r|u,g}$ in the filtering tasks, is no longer just a simple table. It becomes a function mapping the position of user and item nodes into the distribution of target rating. We will give detailed introduction to our framework and solution in the following text.

In the embedded space, we assume each u_i and g_j are random variables drawn independently from prior distributions $P_u(u_i)$ and $P_g(g_j)$. We introduce a rating function $f : \Re_0^+ \mapsto [0, 1]$, which maps the distance between two points (a user and an item) in the embedded space to a real value on $[0, 1]$: the expected rating for the two points. $f(x)$ is a monotonically non-increasing function, $f(0) = 1$, and $f(\infty) = 0$. Intuitively, two points with a smaller mutual distance should have a higher expected rating. At this point, we will assume that the rating function $f(x)$ is given. Later, we will show how this function can be learned from data. The actual rating r_{ij} between two points u_i and g_j in the embedded space is a random variable drawn from a distribution $P_f(r_{ij} \mid u_i, g_j)$ with mean $f(\|u_i - g_j\|)$.

Given all the ratings, R, as evidence and the rating function, f, our task is to put all the user points U and item points G into the embedded space so that the likelihood of observed ratings R is maximized. That is, we want to find the U and G points that maximize the posterior:

$$\begin{aligned}[U^*, G^*] &= \arg\max_{U,G} P(U, G \mid R) \qquad (4.1)\\ &= \arg\max_{U,G} \prod_{i,j|\delta_{ij}=1} P_f(r_{ij} \mid u_i, g_j) \prod_i P_u(u_i) \prod_j P_g(g_j)\ .\end{aligned}$$

$P(U, G, R)$ is a real-valued Bayesian network in which each user and item variable has no parents and each rating variable has two parents (one user and one item). The ratings are given as evidence and the task is to determine the most probable joint assignment to the user and item variables given the ratings.

4.1.1 Gaussian Assumptions

We assume that all the distributions are from the Gaussian family. To be specific,

$$P_u = \mathcal{N}(\mathbf{0}, \mathbf{\Sigma_u})$$

$$P_g = \mathcal{N}(\mathbf{0}, \mathbf{\Sigma_g})$$

$$P_f(r_{ij} \mid u_i, g_j) = \mathcal{N}(f(\|u_i - g_j\|), \sigma_r) \ .$$

Here $\mathcal{N}(\mu, \mathbf{\Sigma})$ is a Gaussian distribution with mean μ and covariance matrix $\mathbf{\Sigma}$. Note that while these distributions are all normal, the function f is non-linear and therefore the resulting joint distribution is not a Gaussian.

4.2 Parameter Estimation

It is intractable to compute the posterior in Equation 4.1 directly. We use Markov chain Monte Carlo sampling.

4.2.1 Metropolis-Hastings Algorithm

In particular, we use the Metropolis-Hastings (MH) algorithm [Metropolis et al., 1953], which was extended to graphical models [Jordan and Weiss, 2002]. Given a graphical model over the random variables $X = \{x_1, x_2, \ldots, x_N\}$, assume a target distribution π over X. For each variable x_i, there is an associated proposal distribution Q_{x_i}: the distribution of new samples for that variable.

Given a current assignment to X, MH randomly picks a variable x_i and tries to replace its value with a new sample x_i' drawn from the proposal Q_{x_i}. Let $Y = X - \{x_i\}$.

The *transition gain ratio* for changing the sample x_i to x_i' is defined as

$$\mathcal{T}^Q(x_i \to x_i') = \frac{Q_{x_i'}(x_i)\pi(Y, x_i')}{Q_{x_i}(x_i')\pi(Y, x_i)} \ . \tag{4.2}$$

The probability of accepting this new sample x_i' is

$$\mathcal{A}(x_i \to x_i') = \min\left\{1, \mathcal{T}^Q(x_i \to x_i')\right\} \ . \tag{4.3}$$

Using the local independencies of the graph, this can be decomposed into a set of local probabilities, shown in the following text.

MH Algorithm for Embedding

In our visualization problem, π is the posterior distribution of U and G given R (Equation 4.1). Initially, we sample from P_u for every u_i and sample from P_g for every g_j. This jointly form a single starting sample (a joint assignment to U and G) for our MCMC method.

We use a a special form of sampler for the proposal procedure. Each time we only sample a change for one existing node. The proposal distribution is again set to Gaussian to ease the computation.

We use proposal distributions Q_{u_i} for the node u_i and Q_{g_j} for the node g_j. We set the proposal distributions to be Gaussians with means at the previous embedded position:

$$Q_{u_i} = \mathcal{N}(u_i, \mathbf{\Sigma}'_{\mathbf{u}})$$
$$Q_{g_j} = \mathcal{N}(g_j, \mathbf{\Sigma}'_{\mathbf{g}}) \ .$$

If we choose the node u_i to be sampled, we draw u'_i from Q_{u_i}, and then compute the accept ratio for this change according to Equation 4.3. Denote $U_{-i} = \{u_1, \ldots, u_{i-1}, u_{i+1}, \ldots, u_m\}$ Using the local independence properties, the transition gain ratio with respect to the rating function f is given by

$$\begin{aligned}\mathcal{T}_f^Q(u_i \to u'_i) &= \frac{Q_{u'_i}(u_i)\pi(U_{-i}, u'_i)}{Q_{u_i}(u'_i)\pi(U_{-i}, u_i)} \\ &= \frac{Q_{u'_i}(u_i)P_u(u'_i)\prod_{j\in G^i} P_f(r_{ij} \mid u'_i, g_j)\prod_{k\neq i}\prod_{j\in G^k} P_f(r_{kj} \mid u_k, g_j)}{Q_{u_i}(u'_i)P_u(u_i)\prod_{j\in G^i} P_f(r_{ij} \mid u_i, g_j)\prod_{k\neq i}\prod_{j\in G^k} P_f(r_{kj} \mid u_k, g_j)} \\ &= \frac{Q_{u'_i}(u_i)P_u(u'_i)\displaystyle\prod_{j\in G^i} P_f(r_{ij} \mid u'_i, g_j)}{Q_{u_i}(u'_i)P_u(u_i)\displaystyle\prod_{j\in G^i} P_f(r_{ij} \mid u_i, g_j)}\end{aligned} \tag{4.4}$$

Similarly, the transition gain ratio for an item node, g_j is

$$\mathcal{T}_f^Q(g_j \to g'_j) = \frac{Q_{g'_j}(g_j)P_g(g'_j)\displaystyle\prod_{i\in U^j} P_f(r_{ij} \mid u_i, g'_j)}{Q_{g_j}(g'_j)P_g(g_j)\displaystyle\prod_{i\in U^j} P_f(r_{ij} \mid u_i, g_j)} \ . \tag{4.5}$$

We repeat the above resampling phase until the process has mixed. The stationary distribution of this procedure is the true posterior $P(U, G \mid R)$.

4.2.2 Simulated Annealing

Recall that we want the $\arg\max_{U,G} P(U, G \mid R)$. The Metropolis-Hastings algorithm will give us joint samples of U and G, drawn from that posterior $P(U, G \mid R)$. To get the samples that maximize the posterior, we modify the standard MH algorithm along the lines of the simulated annealing (SA) algorithm [Kirkpatrick et al., 1983].

In particular, we modify Equation 4.2 to add an annealing temperature, β:

$$\mathcal{T}^Q(x_i \to x_i') = \frac{Q_{x_i'}(x_i)\pi(Y, x_i')^\beta}{Q_{x_i}(x_i')\pi(Y, x_i)^\beta}$$

The transition gain ratios of equations 4.4 and 4.5 are then

$$\mathcal{T}_f^Q(u_i \to u_i') = \frac{Q_{u_i'}(u_i)\left[P_u(u_i') \prod_{j \in G^i} P_f(r_{ij} \mid u_i', g_j)\right]^\beta}{Q_{u_i}(u_i')\left[P_u(u_i) \prod_{j \in G^i} P_f(r_{ij} \mid u_i, g_j)\right]^\beta}$$

$$\mathcal{T}_f^Q(g_j \to g_j') = \frac{Q_{g_j'}(g_j)\left[P_g(g_j') \prod_{i \in U^j} P_f(r_{ij} \mid u_i, g_j')\right]^\beta}{Q_{g_j}(g_j')\left[P_g(g_j) \prod_{i \in U^j} P_f(r_{ij} \mid u_i, g_j)\right]^\beta}$$

The added temperature factor β grows gradually from 1 to ∞. Initially $\beta = 1$, and this method is the same as the standard Metropolis-Hastings algorithm. As β grows, the simulated annealing algorithm penalizes changes resulting in lower likelihoods; the algorithm tends to only climb uphill in the posterior distribution.

4.2.3 Learn the Rating Function

Until now, we have assumed that the rating function f was known. However, we would like this function to adapt to the collaborative data.

It would be straight-forward to select f from a family (for example the exponential, $f(x) = e^{-\lambda x}$). However, the actual rating function may have a very different shape. Instead we note that all collaborative datasets of which we are aware have a finite number of values for the ratings. Many are binary ("like" or "do not like") and others are based on a five- or ten-point scale. Continuous, real-valued ratings are seldom used. We therefore let f be a step function with discrete quantizations. A sample example of such a rating function is shown in Figure 4.1.

We discretize f into K quantizations. Let $\Theta = \{\theta_i \mid i = 1, \ldots, K\}$ be the set of K splitting points in sorted order, with $\theta_K = \infty$. Given the set of splitting points Θ, the rating function is[1]:

$$f(x;\Theta) = 1 - \frac{i}{K}, \quad \text{if } \theta_i \leq x < \theta_{i+1} \ .$$

From Equation 4.1 and our Gaussian assumptions, we have

$$P(U, G \mid R) \propto \prod_{i,j|\delta_{ij}=1} P_f(r_{ij} \mid u_i, g_j) \propto \exp(-\frac{1}{2} \sum_{i,j|\delta_{ij}=1} (r_{ij} - f(\|u_i - g_j\|))^2)$$

The proportionality is satisfied because during the estimation of the rating function, we fix all the user and item points.

[1]Assume $\theta_0 = 0$.

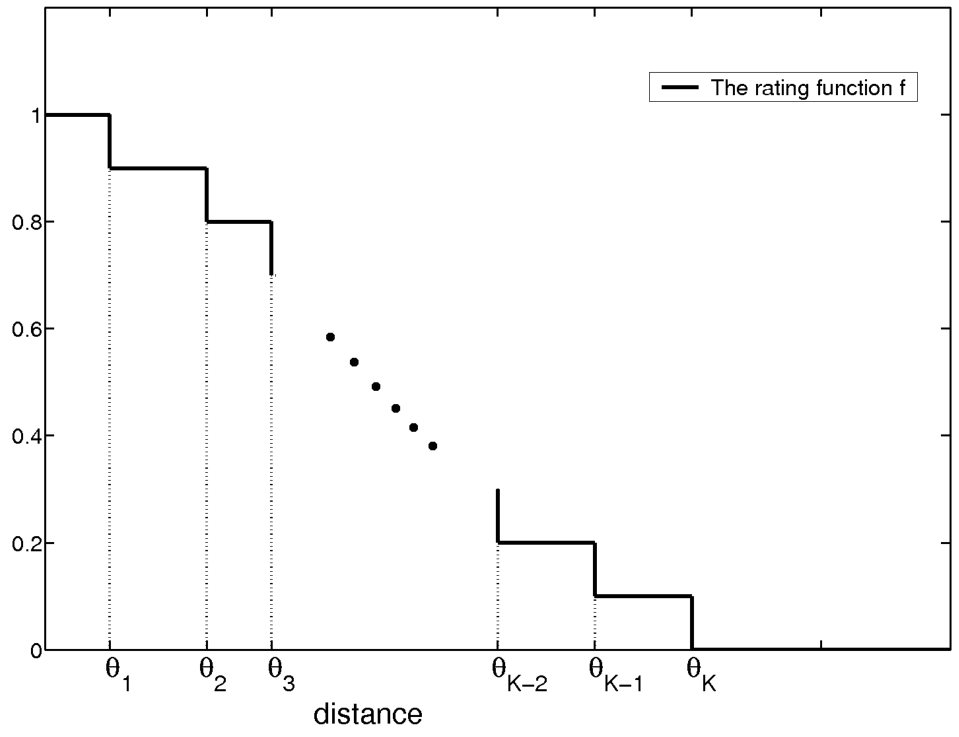


Figure 4.1: Discretization of rating function.

To maximize the above posterior distribution $P(U, G \mid R)$ with respect to f (represented by Θ, the problem is not transformed to:

$$\Theta^* = \arg\min_{\Theta} \sum_{i,j|\delta_{ij}=1} E[(f(\|u_i - g_j\|; \Theta) - r_{ij})^2] \ , \tag{4.6}$$

where the expectation is with respect to the posterior distribution over U and G. This formulation is equivalent to maximizing the probability of the ratings; the squared error in the above equation comes directly from the Gaussian assumption regarding the distribution P_f.

We use the expectation maximization (EM) algorithm [Dempster et al., 1977] to learn the rating function. We initially set $\Theta = \Theta^0$, a random starting point that meets our requirements for f.

The E-step employs MH to sample from the expectations in Equation 4.6 using the rating function $f^k = f(\cdot; \Theta^k)$ at the k-th iteration.

The M-step updates Θ^{k+1} based on the generated sample configurations of the embedded space (which approximate the expectations of Equation 4.6). Using all the u_i and g_j points, the optimal rating function is updated according to Equation 4.6. Let N be the number of terms in the summation of Equation 4.6 (one for each rating for each sample). The M-step optimization can be done efficiently (and exactly) in $O(NK)$ time using dynamic programming.

Before updating the rating function in the M-step, we renormalize all the points in the embedding space. Due to our assumptions that P_u and P_g are fixed Gaussian distributions with zero mean and that we have the freedom to change f, if the above procedure were run without modification, all the points would collapse together toward the origin. Consequently, the learned rating function would have splitting points with smaller and smaller values. We fix this by a simple normalization step that scales and translates the points to reset the mean of all of the points to the origin and the variance of their positions to one. Note that this is not a general "whitening" step in that we only multiply the points by a scalar, not a matrix.

4.2.4 The Algorithm

To put everything together, the overall algorithm in our approach is listed in Algorithm 3. The parameters of this algorithm are l_s (the number of samples used for estimating the expectation), l_b (the number of samples necessary for the MCMC process to converge), ϵ (the amount by which to increase β), and the variances of the Gaussian distributions.

4.3 Experiments and Results

We discuss our three datasets, our methodology for comparison, and then compare our algorithm to three others.

4.3.1 Algorithm Initialization

Because we are learning the rating function f, the absolute positions of the embedded points will not affect our approach directly. Rather, the relative positions of the points matter. There-

Algorithm 3 $[U, G]$ $\Leftarrow$Embed-Graph(R,D)

```
Inputs: $R$: rating matrix, $D$: embedding dimensionality
Outputs: $U$ and $G$: embedded points

$\beta \Leftarrow 1$
$P_u \Leftarrow \mathcal{N}(\mathbf{0}, \mathbf{\Sigma_u}), P_g \Leftarrow \mathcal{N}(\mathbf{0}, \mathbf{\Sigma_g})$
Sample $\{u_i \sim P_u\}_{i=1}^{m}$
Sample $\{g_j \sim P_g\}_{j=1}^{n}$
$Q_{u_i} \Leftarrow \mathcal{N}(u_i, \mathbf{\Sigma'_u}), Q_{g_j} \Leftarrow \mathcal{N}(g_j, \mathbf{\Sigma'_g})$
$P_f(r_{ij} \mid u_i, g_j) \Leftarrow \mathcal{N}(f(\|u_i - g_j\|), \sigma_r)$
$f \Leftarrow f(\cdot; \Theta^0)$
repeat
      // E-Step:
   $\mathcal{S} \Leftarrow \emptyset$
   for $k = 1$ to $l_b + l_s$ do
      Randomly pick a point $x_i$ from samples in $[U, G]$
      if $x_i$ is a user point $u_i$ then
         Sample $u'_i \sim Q_{u_i}$
         $u_i \Leftarrow u'_i$ with probability $\mathcal{A}_f(u_i \rightarrow u'_i)$
      else if $x_i$ is an item point $g_j$ then
         Sample $g'_j \sim Q_{g_j}$
         $g_j \Leftarrow g'_j$ with probability $\mathcal{A}_f(g_j \rightarrow g'_j)$
      end if
      if $k > l_b$ then      // burn in for $l_b$ iterations
         Add $(U, G)$ to $\mathcal{S}$      // Save last $l_s$ iterations
      end if
      $[U, G]$ $\Leftarrow$normalize($[U, G]$)
   end for
   $\beta \Leftarrow (1 + \epsilon)\beta$
      // M-Step:
   $f(\cdot; \Theta)$ $\Leftarrow$learned rating function using $\mathcal{S}$
until The current sample $[U, G]$ is stable
```

fore, the overall scale of $\mathbf{\Sigma_u}$ and $\mathbf{\Sigma_g}$ do not affect the result.

In particular, for all the three datasets, we set $\mathbf{\Sigma_u}, \mathbf{\Sigma_g}, \mathbf{\Sigma'_u}$ and $\mathbf{\Sigma'_g}$ each to be the identity matrix. We set $\sigma_r = 0.25$ for SAT, $\sigma_r = 0.1$ for MovieLens, and $\sigma_r = 0.05$ for BGG. These values directly reflect the discretization of the rating scores. Our informal tests show that the algorithm is not sensitive to these particular numbers and we have made no effort to tune them.

For the rating function f, we choose to set Θ^0 directly using an M-step from the samples drawn from their priors. For our experiments, we set $l_s = 2000$, $l_b = 1000$, and $\epsilon = 0.02$ for all three datasets.

4.3.2 Experiment Datasets

We test our visualization algorithm on the following data sets.

The SAT dataset contains SAT II subject examination scores for 40 questions chosen from a study guide of historic questions and 296 users. SAT II is a standard exam taken by high school seniors applying to colleges in the United States. All the scores are either 0 or 1 (indicating whether the student got the question correct), and there are no missing values. The 40 questions are from the subjects French, Mathematics, History and Biology. The exam was administered on-line over the course of one week.

The BGG dataset is the same as the one used in Chapter 3.

Finally, the MovieLens dataset contains ratings from users on a variety of movies. All the ratings are integers from 1 to 5. We picked 400 users and 50 movies, maximizing the rating density. The rating matrix density is 41.0% on this subset. This dataset is publicly available from *movielens.umn.edu*.

4.3.3 Implementation Issues

We compare our results with locally linear embedding (LLE) [Saul and Roweis, 2003], Eigentaste (ET) [Goldberg et al., 2001], and co-occurrence data embedding (CODE) [Globerson et al., 2005]. None of the algorithms is exactly suited to our problem, so we discuss our adaptations in this section.

If we consider the rating matrix as a set of points in the high dimensional space, we can use LLE to embed them into a lower dimensional space. The LLE algorithm requires a full rating matrix R. This is not available for the MovieLens and BGG datasets. We use linear regression to fill the missing ratings. This method is discussed in Section 3.4.2 in the previous chapter.

Both LLE and ET can embed either users or items into an Euclidean space. Yet, neither of them can embed both in the same space. We tried several ways to extend them and to make them comparable. One straight-forward way is to embed all the user points first into the space. Then for every item, find all the users who gave it its highest rating, and place this item at the mean of those users points.

For our results, we used an alternative method, which performed better than the one above. Let $\hat{R}$ be the full rating matrix filled in using linear regression. We introduce a correlation matrix C among all n items. The diagonal C_{ii} is set to 1. Let R_i be the i-th column of $\hat{R}$,

$$C_{ij} = \frac{R_i^\top R_j}{\|R_i\| \cdot \|R_j\|} \ .$$

We then let $X = [C \ \hat{R}^\top]$ and use LLE or ET to embed X into the target Euclidean space. The first n points correspond to the items and the last m points to the users.

ET only works if there is a *gauge set* of items which all users have rated. However, in the MovieLens and BGG datasets, no such gauge set exists. Using the above regression technique to fill in a gauge set results in bad (and misleading) results, so we omitted them and have only included ET results for the SAT dataset.

The CODE algorithm requires co-occurrence statistics between users and items. The relationship between co-occurrences and ratings is not clear. However, it is natural to assume r_{ij} is proportional to the probability of the co-occurrence of u_i and g_j. Intuitively, a higher rating indicates it is more likely that the user and item "occur" at the same time. We set the empirical distribution of (u, g) to be proportional to the rating matrix (filled by linear regression if there are missing ratings). We initialize the mappings uniformly and randomly from the set $[-0.5, 0.5]^D$ as the starting point for the optimization.

Our linear regression method for filling in missing values has proven reasonable on the prediction task, but admittedly it is not the most sophisticated algorithm possible. Therefore, to distinguish embedding factors from data completion factors, we also ran our MCMC algorithm on the completed rating matrix from linear regression.

Both our method and CODE have variable running times (number of EM iterations in our case, number of random restarts for CODE). For the results reported here, we gave each 30 seconds of CPU time on a 2.8 GHz processor.

4.3.4 Sample Results

The SAT data was selected because of our ability to extract a “ground truth.” In particular, we expect that when embedded, the questions from the same subjects should be grouped together. Figure 4.2 shows the embedding for dimension 2 using the simulated annealing approach (along with the three other approaches).

There are ten questions in each category. We can clearly see that our method clusters all the French questions tightly together. The same happens for the Math questions. (There are eight Math questions that overlap in a small area.) The other methods do not produce as tight clusters.

The History and Biology questions do not cluster as well. Further data analysis has shown that there is very little predictability in the History and Biology questions, so this result is perhaps not surprising. The French and Math questions tended to test a body of knowledge that is often retained as a coherent block, where as the History and Biology questions on this exam tended to test more isolated blocks of knowledge.

Figure 4.2 also shows that the user points and the item points intermix more evenly with our approach. This meets our expectation that for any user, we can always find things they like or dislike (questions on which they perform well or poorly). In the embedding results of LLE, ET, and CODE, a large number of user points lie in parts of the graph outside the

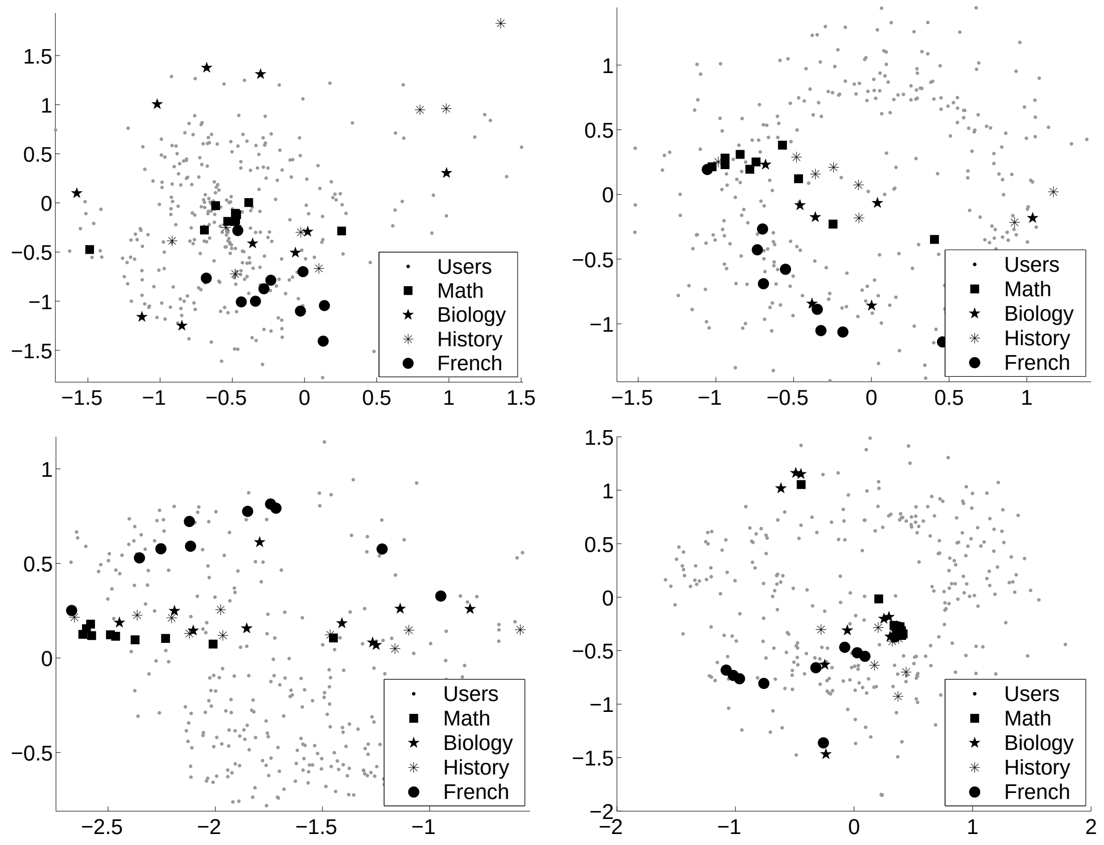


Figure 4.2: 2-Dimensional embeddings for the SAT questions using a simulated annealing version of our MCMC algorithm (top left), local linear embedding (top right), Eigentaste (bottom left), and CODE (bottom right).

convex hull of the the question points. This makes it impractical to make recommendations to those users based on the visualizations.

4.3.5 Evaluation Criteria

There are no prior standard metrics for evaluating the quality of the embedded graph. We introduce *Kendall's tau* [Kendall, 1955] as a suitable evaluation criterion. Kendall's tau is used to compute the correlation in ordering between two sequences X and Y. It is especially useful for evaluating the correlation between two sequences that may have many ties.

Given two sequences X and Y of the same length, a pair $(i, j), i \neq j$ is called *concordant* if the ordering of X_i and X_j is the same as the ordering of Y_i and Y_j. By contrast, if the relative ordering is different, this pair (i, j) is called *discordant*. If $X_i = X_j$ or $Y_i = Y_j$, then (i, j) is neither concordant or discordant, and it is called an *extra x* pair or *extra y* pair, respectively.

Kendall's tau is defined as

$$\tau = \frac{C - D}{\sqrt{C + D + E_y}\sqrt{C + D + E_x}} \tag{4.7}$$

where C is the number of all concordant pairs, and D is the number of all discordant pairs. E_x and E_y are the numbers of extra x pairs and extra y pairs.

It is easy to verify that τ is always between -1 and 1. $\tau = 1$ indicates the two sequences have perfect positive correlation, and $\tau = -1$ indicates perfect negative correlation. $\tau = 0$

indicates their orderings are independent.

To evaluate the quality of the graph in the embedded space, for each experiment we randomly selected a set of users, $\bar{U}$, and items, $\bar{G}$, as testing users and items. All the ratings between users in $\bar{U}$ and items in $\bar{G}$ were held out for testing and were not used in generating the embedding.

The embedding algorithms will produce the embedded points for those nodes in $\bar{U}$ and $\bar{G}$. In order to evaluate the embedding quality, we generate two sequences and compute Kendall's tau between them: sequence X contains the actual ratings between all the pairs $u_i \in \bar{U}$ and $g_j \in \bar{G}$ such that $\delta_{ij} = 1$, and sequence Y contains the distances between the corresponding u_i and g_j in the embedded graph.

A good embedding will place u_i far from g_j if r_{ij} is small, and close if r_{ij} is large. Kendall's tau for the above two sequences exactly evaluate this correlation. Denote τ to be the Kendall's tau for the sequences X and Y. Negative values of τ indicate good embeddings, and we expect the values from embedding algorithms to be smaller than 0 (that of a random embedding).

4.3.6 Experimental Results

We ran our MCMC algorithm both with and without simulated annealing, along with LLE, ET and CODE. We randomly selected one quarter of the users and one quarter of the items for testing ($\bar{U}$ and $\bar{G}$ from above). We randomly selected other users and items to form a training set ($\tilde{U}$ and $\tilde{G}$). All ratings between members of $\tilde{G}$ and $\tilde{U}$, $\tilde{G}$ and $\bar{U}$, and $\bar{G}$ and $\tilde{U}$ are used for training. As stated previously, the ratings between members of $\bar{G}$ and $\bar{U}$ are used for testing. It is necessary to include the ratings between $\tilde{G}$ and $\bar{U}$ (and likewise between $\bar{G}$ and $\tilde{U}$) in order to connect the test users and items with the training users and items.

Because the existence of the test set, there are always missing ratings in the rating matrices used. We use linear regression to fill those ratings. We also ran the MCMC algorithm on the same filled data as LLE, ET and CODE used (MCMC-REG in the graphs).

For each dataset size (number of items in $\tilde{G}$), we ran 25 independent experiments and recorded the means and standard deviations across the experiments for all algorithms. Every algorithm was run on the same set of training and testing sets.

For each of these datasets, Figure 4.3 shows the comparison of our methods to LLE, ET, CODE, and a random embedding, as a function of the size of the training set. We also computed an "ideal embedding" value for τ. Because of ties, Kendall's tau cannot always reach -1, so we calculate the lowest possible value for τ on the random dataset drawn. This takes nothing into account except ties and it is *highly* optimistic and probably not obtainable at such low dimensions. The optimal values for SAT, BGG, and MovieLens datasets are approximately -0.63, -0.87 and -0.90 respectively. We ran LLE algorithm with training size starting at 22 for SAT and 24 for MovieLens because of matrix inversion problems for smaller training sizes.

Figure 4.4 shows another experiment with the same evaluation criteria. In this experiment, we fix the testing data as usual, and use all the remaining data for training. The plot shows Kendall's tau as a function of the number of dimensions of the embedding space.

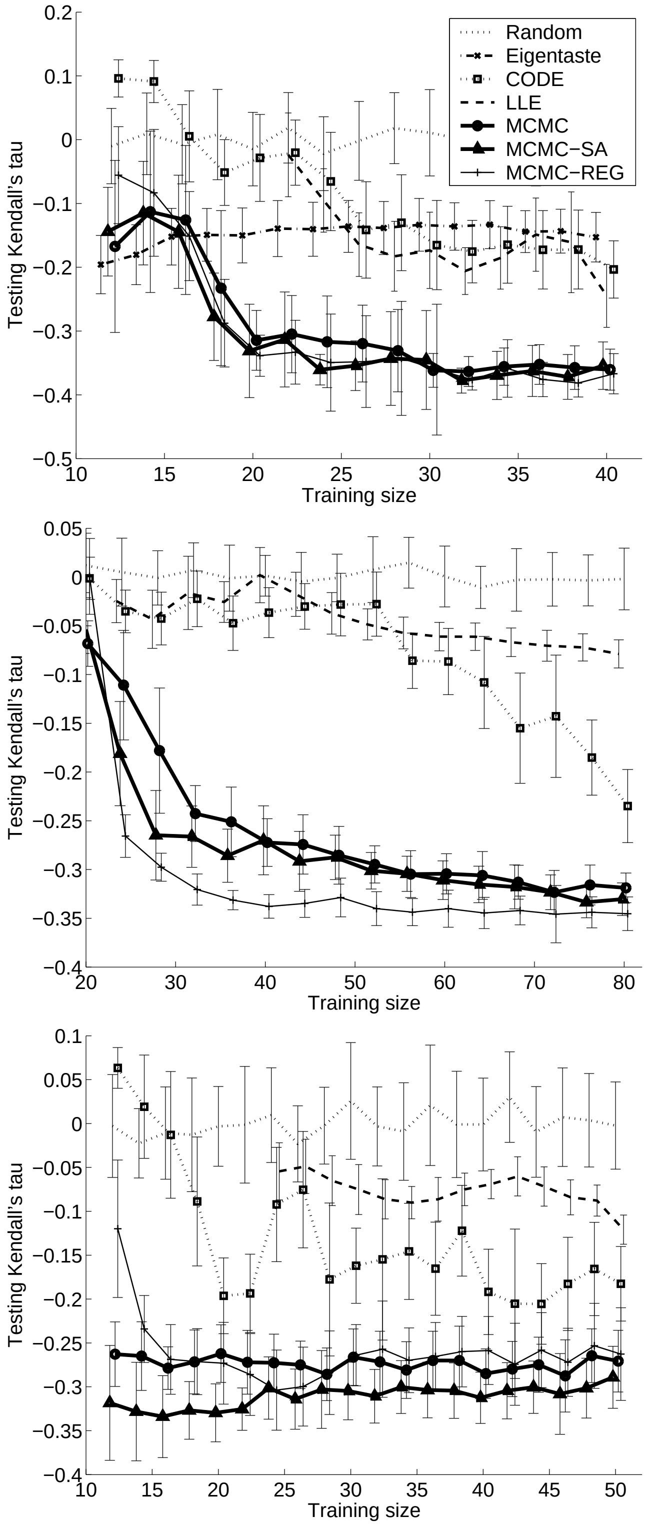


Figure 4.3: Performance of embedding algorithms on the three datasets as a function of training set size. Results are for the data sets SAT, BGG and MovieLens from top to bottom.

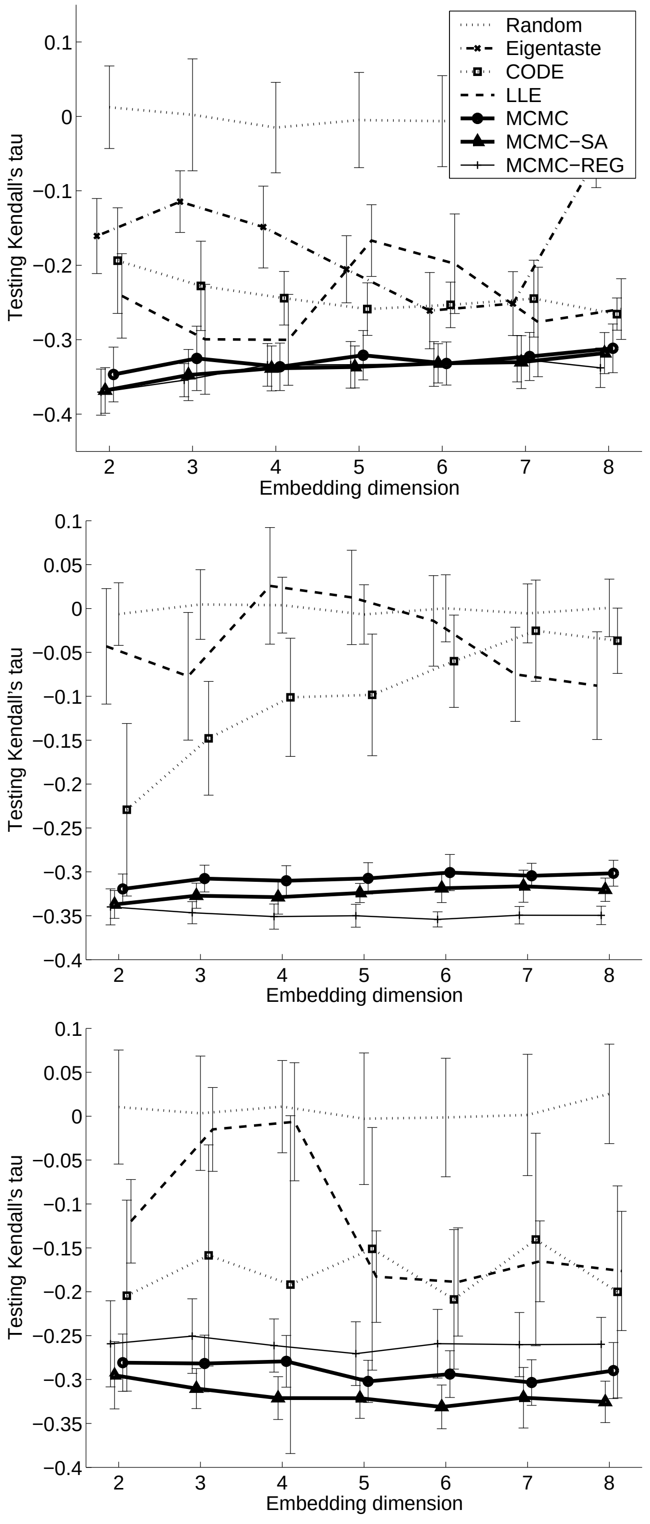


Figure 4.4: Performance of embedding algorithms on the three datasets as a function of embedding dimensions. Results are for the data sets SAT, BGG and MovieLens from top to bottom.

4.3.7 Analysis of the Results

From the experiments above, we can see that when the rating matrix is denser, the embedding algorithm achieve better results. Our sampling method, with (MCMC-SA) and without (MCMC) simulated annealing, outperformed LLE, ET and CODE. None of them were designed with this type of data in mind, so we do not present these results to disparage those methods, but there were no other methods available to test against. Note that our MCMC algorithm on linear regression filled data (MCMC-REG) has similar performance to directly applying our MCMC method on data with missing ratings. This implies that it is not our regression that is causing the poor results from the other algorithms, but rather their misfit to this problem. We would also note that our method also seems more stable (smaller variance) than the other algorithms compared.

On the SAT dataset, which contains full density of ratings, our algorithms show strong negative Kendall's τ which indicates good visualization results. In most cases, using simulated annealing helps improve the quality of embedding (compared to "normal" MCMC). As the training size grows, we have more information on the relations between all the user and item points, and that leads to better performance for all the algorithms.

The BGG and MovieLens datasets have many missing ratings and the ratings values are more subjective and therefore noisier. Our algorithm is not as competitive with the "ideal" value for Kendall's tau, but we feel that this ideal value is wildly optimistic in these settings. Our algorithm does perform better than random embeddings, LLE, and CODE.

Chapter 5

Embedding Images

In extension to the visualization of collaborative data, we can also view this problem as a dimensionality reduction problem. The rating matrix can be any raw input data matrix $\mathbf{X}$ and our framework discussed in the previous chapter can be used to efficiently embed the columns of $\mathbf{X}$ into some low dimensional space. It is natural to think about images. They, too, are high dimensional data. If we can come up with a reasonable processing to transform the images into collaborative data, we can also embed images.

As is previously discussed in Chapter 1, image embedding problem is to place images into a Euclidean space or other layout so that similar images lie close to each other in the embedding. Most of the time there is additional information about the images also available. In the case of search engines, historic user click counts can be used for collaborative filtering. The search engine can record the number of clicks by users, along with their query words. For personal image collections, time stamps are often useful for correlating similar photographs.

We listed some dimensionality reduction algorithms in Chapter 2. In general, they work for image embedding tasks. However, they treat each dimension independently and do not preserve the image structure. Thus, they produce the same result if the pixels' locations were permuted (the same way in each image). We would like to exploit our knowledge of image structure to boost the embedding quality. There are well developed algorithms for image feature extraction, in particular the scale invariant feature transform (SIFT) introduced in [Lowe, 2003]. The SIFT features extracted from images well preserve the image's "key points." Our similarity metric is directly based on the SIFT features.

In Section 5.1 we formulate the image embedding problem formally, and then in Section 5.2 we give a detailed explanation of our solution. Additionally, as we show in Section 5.2.3, our method can be adapted to non-Euclidean spaces. In particular, we can embed the images into a grid or table, suitable for visualization. In Section 5.3 we compare the results of our algorithm with other competing ones with popular image data sets.

5.1 Image Embedding Problem

In previous Chapter 4, we proposed an algorithm that can visualize collaborative data. We used a real-valued Bayesian network to model the embedded positions of the users and items, along with the ratings that relating them. In addition, we proposed to use the nonparametric statistic Kendall's τ [Kendall, 1955] as a criterion to evaluate the embedding quality.

In this chapter, we adopt the overall structure of the visualization of collaborative data.

We do not have “users” that have rated the images, but we employ SIFT features in a similar role (see Section 5.2). We also simplify the optimization method. Instead of trying to maximize the posterior distribution of a complex graphical model, we propose to directly minimize Kendall’s τ.

Scale Invariant Feature Transforms

Scale Invariant Feature Transforms (SIFT) [Lowe, 2003] transform raw images into scale-invariant local features. Many existing algorithms on supervised and unsupervised image processing use the SIFT feature sets as their representation for images, for example [Se et al., 2001] and [Scovanner et al., 2007].

Each image can have a possibly different number of SIFT features, depending on the number of “points of interest” in the image. Each feature is a fixed-length vector describing the local image patch at a location in the image. SIFT features have their advantages in terms of describing characteristics of an image. They are robust to scaling and in-plane rotation, and relative easy to compute and manipulate. Similar images will have a significant number of similar SIFT features, which makes it a suitable representation for image comparison.

Notation

Table 5.1 lists the notation we use in this chapter. Note that we use I and K to denote both images and clusters or their positions in the embedding space if there is no ambiguity.

$\mathbf{I} = \{I_1, I_2, \ldots, I_n\}$	set of n images
$\mathbf{S^i} = \{S^i_1, S^i_2, \ldots, S^i_{n_i}\}$	set of I_i's SIFT features
$\mathbf{K} = \{K_1, K_2, \ldots, K_m\}$	set of m feature clusters (explained below)
$N(S^i, j)$	number of features in S^i that belong to K_j
$r^i_j = \frac{N(S^i,j)}{\lVert S^i \rVert}$	portion of features in S^i that belong to K_j
$d^i_j = \lVert I_i - K_j \rVert$	Distance in the embedding between I_i and K_j

Table 5.1: Notation for image embedding problem.

5.2 Approach

Given a set of images **I**, our task is to embed them into a d-dimensional space. Distances in this embedded space should capture the image similarity: We want to put similar images near each other, and dissimilar ones far apart.

We first extract the SIFT features for all the images, and then cluster all features from all images together into m groups, **K**. We have found that the end results are fairly stable with respect to the number of clusters and the clustering algorithm. We use k-means to do this clustering.

For any image I_i, we then count $N(S^i, j)$, the number of features in S^i that belong to the cluster K_j. If we divide $N(S^i, j)$ by the size of S^i, we have a distribution of "membership" to the cluster K_j for image I_i. So we can consider $r^i_j = \frac{N(S^i,j)}{\lVert S^i \rVert}$ as the fractional "vote" of I_i for K_j.

We view each SIFT cluster as representing a more general feature of an image. In this way, it is similar to a user from collaborative filtering data. Each image is an item and r^i_j

represents how well SIFT cluster K_j “describes” image I_i. So, following Chapter 4, we embed both the images (items) and SIFT clusters (users) in the same space so that SIFT clusters are near images that they like (have many examples of the feature) and are far away from those they do not like (do not have the corresponding SIFT features).

Consider the case where we already have a potential embedding, containing image points and cluster points. Let the images have points $\{I_i\}$ in that space, and the SIFT feature clusters have points $\{K_j\}$. For any particular image I_i, we can compute its distance to all the cluster points, which we denote $d_j^i = \|I_i - K_j\|$. From the SIFT feature clustering, we also have the membership distribution r_j^i. Let τ^i be Kendall’s τ between these two sequences for image i, as computed through Equation 4.7.

Intuitively, the more features from cluster K_j contained in the image (*i.e.* the larger r_j^i) the smaller the image’s distance to K_j should be in the embedding. So we would like a negative correlation between the two sequences, or, equivalently, a negative value of τ^i. The smaller τ is, the better embedding results are. We propose directly minimizing the average Kendall’s τ for all the images.

5.2.1 Problem Formulation

Denote the average Kendall’s τ of an embedding to be

$$T(\mathbf{I}, \mathbf{K}) = \frac{1}{n} \sum_{i=1}^{n} \tau(\{r_j^i\}_{j=1}^{m}, \{d_j^i\}_{j=1}^{m}) \ . \tag{5.1}$$

The image embedding problem can now be formulated as

$$[\mathbf{I}^*, \mathbf{K}^*] = \arg\min_{\mathbf{I},\mathbf{K}} T(\mathbf{I}, \mathbf{K}) \ . \tag{5.2}$$

5.2.2 Simulated Annealing

Exact algorithms to minimize the function T are not possible due to its combinatorial nature. Instead, we use simulated annealing [Kirkpatrick et al., 1983]. Minimizing T is equivalent to maximizing the energy function

$$f(\mathbf{I}, \mathbf{K}) = \exp(-T(\mathbf{I}, \mathbf{K})) \ . \tag{5.3}$$

Simulated annealing is already discussed in Chapter 4. Here we briefly explain how we use this technique for this problem. We begin with a random embedding of $\mathbf{I}$ and $\mathbf{K}$. Samples from a multi-variate Gaussian distribution work fine in practice. At each time step, we randomly choose a point, either an image or a cluster, to resample. For example, if the point I_i is picked, we then propose a new point $\tilde{I}_i$. For this, we also use a multi-variate Gaussian proposal distribution centered at the old position I_i. To calculate the change of the function T, it is not necessary to recompute the Kendall's τ for all images; it suffices to calculate the change for just image i. Let $\tilde{d}_j^i$ be the distance between the proposed new point

$\tilde{I}_i$ and any cluster point K_j, then

$$\Delta T(I_i \to \tilde{I}_i) = \frac{1}{n}\left(\tau(\{r^i_j\}, \{\tilde{d}^i_j\}) - \tau(\{r^i_j\}, \{d^i_j\})\right) \ . \tag{5.4}$$

Hence we accept this new point with probability

$$\mathcal{A}(I_i \to \tilde{I}_i) = \min\{\exp(-\Delta T(I_i \to \tilde{I}_i)), 1\} \ . \tag{5.5}$$

The calculations are similar for moving a cluster point K_j. Although all the Kendall's τ values may change, we can still update the τ values efficiently. Recall from Equation 4.7, for each image I_i, only d^i_j changes, so the values of C (concordance) or D (discordance) can change by at most 1. If we store the previous C and D values along with the Kendall's τ for all the images, we can perform this update efficiently.

We keep iterating this resampling procedure until convergence. Remember that our problem in Equation 5.2 is to find the $\arg\min$ of the T function. As is standard in simulated annealing, we add a temperature parameter β to the system. β is initially 1, and we let it grow toward ∞.

We only need to change the Equation 5.5 to be

$$\mathcal{A}(I_i \to \tilde{I}_i) = \min\{\exp(-\beta\Delta T(I_i \to \tilde{I}_i)), 1\} \ . \tag{5.6}$$

After each resampling, we increase β by a small amount. Intuitively, as β grows larger,

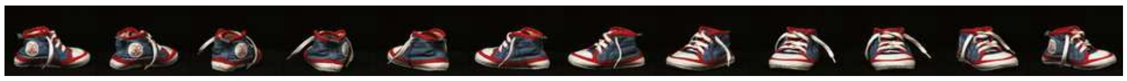


Figure 5.1: One-dimensional embedding for the *shoes* object in the ALOI data set.

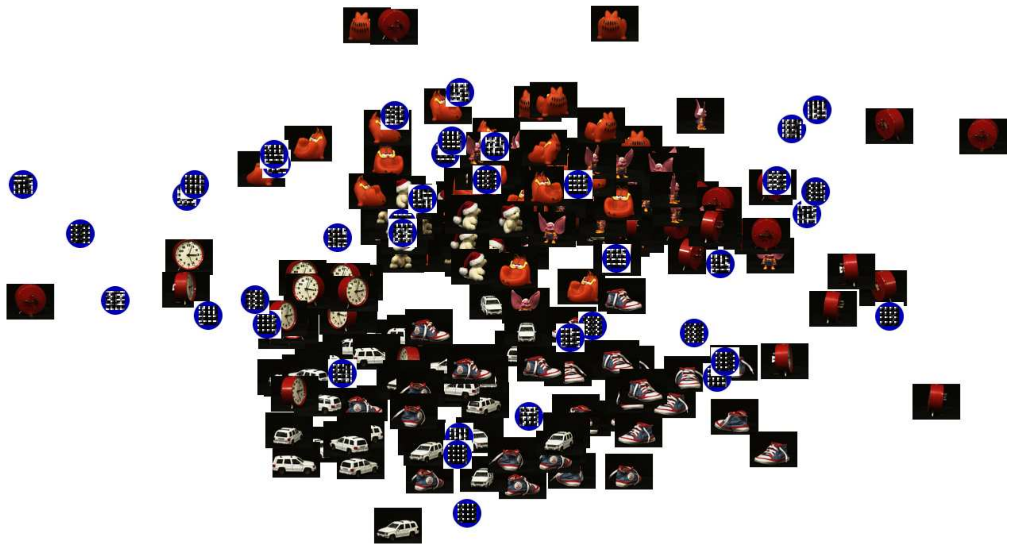


Figure 5.2: Euclidean embedding results for a subset of the ALOI data set.

the system is more reluctant to accept resamples leading to larger T values (worse embeddings).

5.2.3 Grid-based Image Embedding

The final embedding of the images will inevitably involve much overlap if we plot the images in their embedded space. If the embedding is simply for dimension reduction as an initial step of machine learning, this is not a problem. However, it is a problem if visualization is the desired goal.

Unlike many other dimension reduction algorithms, our framework can be easily adapted to a “grid-based” approach. We do this by setting the target embedding space to be a grid, *i.e.* each image can be only placed into one of the embedding grid cells. The proposal distribution for changing an image can be a uniform distribution over all cells, or, more efficiently, a uniform distribution over the neighbors of the image's current cell. If there is already another image that takes the proposed grid position, then the proposed move is to *swap* the two images; otherwise the proposal is to *move* the image position to the new cell.

Everything else in the above algorithm remains the same. We also restrict the SIFT cluster locations to the grid cells. However, we do not require there to be at most one per cell. Rather, the SIFT clusters may coexist on the same cell as we will not be displaying them.

Grid-based image embedding is especially suitable for the applications that are targeted at users instead of machines, for example organizing and visualizing personal photo galleries.

5.3 Experiments and Results

We discuss the data sets we used, show sample embedding results, and compare our algorithm with other related ones.

5.3.1 Data Sets and Accuracy Metric

We tested the embedding algorithms on two real-world data sets. The Amsterdam Library of Object Images (ALOI) [Geusebroek et al., 2005] contains images of a set of small objects. It has images for 1000 objects, each has 72 images taken from different viewing angles. This data set is noise-free and is relatively easy for unsupervised image embeddings. Caltech101

[Fei-Fei et al., 2004] is another well-known data set containing 101 categories of (possibly very different) images. It is a harder data set for image dimension reduction algorithms.

It is usually difficult to numerically evaluate the embedding results. Fortunately we know the ground-truth category for each image in both data sets we used. We use the k-nearest neighbors accuracy (kNNA) to evaluate the embedding:

$$kNNA(I_i) = \frac{\sum_{I_j \in \mathcal{N}_k(I_i)} \text{Category}(I_i)\text{=Category}(I_j)}{k} , \tag{5.7}$$

where $\mathcal{N}_k(I_i)$ is the set of the k nearest neighbors of image I_i in the embedding. The average kNNA for the embedding is just $kNNA = \frac{\sum_i kNNA(I_i)}{\sum_i 1}$.

It is clear that kNNA is always between 0 and 1. Larger average kNNA values indicate better embedding quality. Intuitively, a good embedding should place images from the same category close to each other, and hence boost the kNNA values for all the images. Perfect embeddings will have NN accuracy 1 for small k, while random embeddings will have accuracy around the inverse of the number of categories.

Using kNNA, we compared our simulated annealing embedding (SAE) algorithm with Isomap, LLE, and SDE. In addition to using the raw pixels as input vectors for Isomap, LLE and SDE, we also ran those embedding algorithms using the SIFT distributions as the images' vector representation. This gave the other embedding algorithms the same information as our embedding algorithm and helps to distinguish the advantages of our algorithm from the advantages of our representation. The results for their algorithms on SIFT features are shown

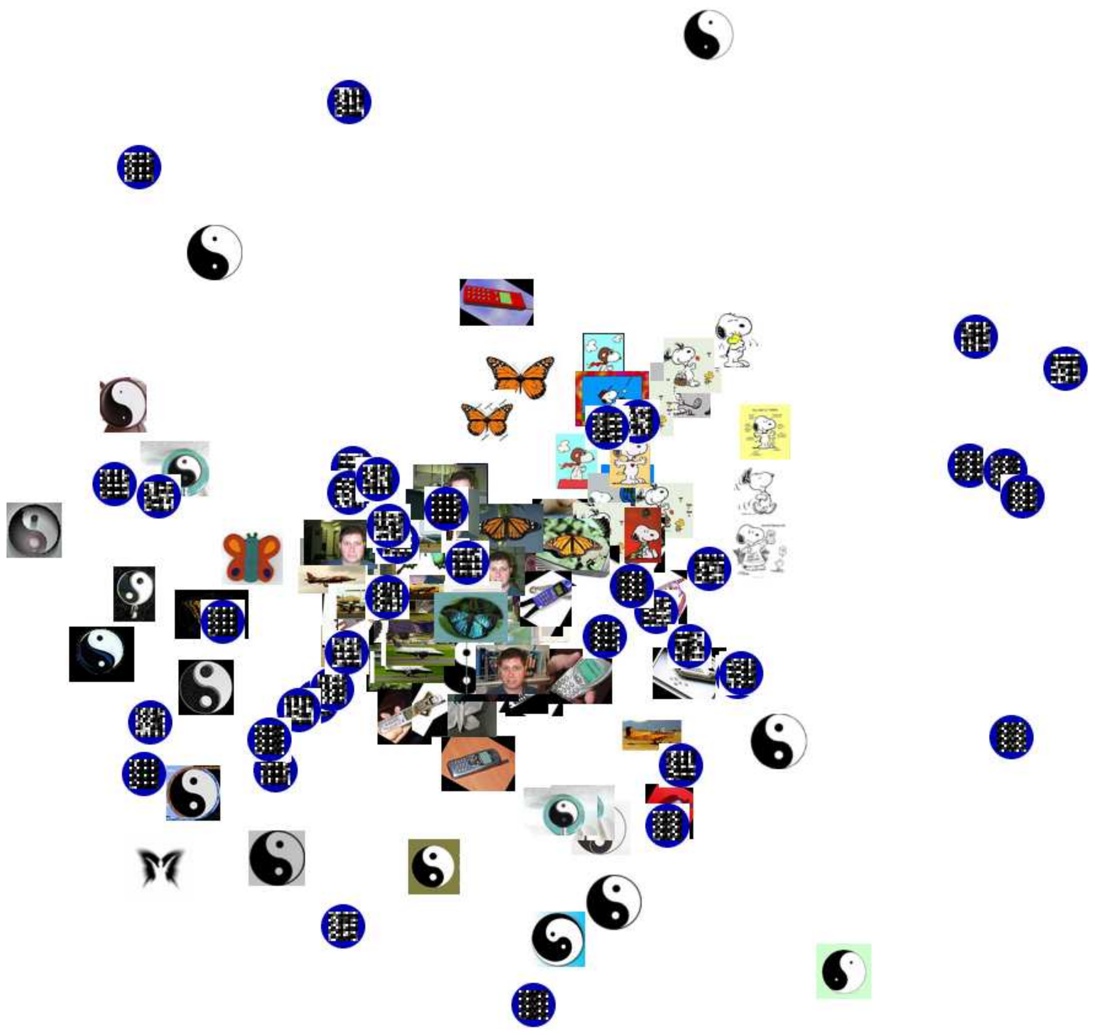


Figure 5.3: Euclidean embedding results for a subset of the Caltech101 data set.

with the label "(SIFT)".

5.3.2 Sample SAE Embeddings

Figure 5.1 shows a sample embedding of shoe images from the ALOI data set, viewed from different angles using SAE. We picked 12 evenly spaced images in the derived embedding from all of the 72 embedded shoe images. It is clear that SAE preserves the pairwise similarity well in terms of the shoe's rotation.

Figure 5.2 shows a sample two-dimensional Euclidean embedding for the ALOI data set. We randomly chose a subset of 6 objects (categories), each with 36 different images. Figure 5.3 shows similar results for the Caltech101 data set. For this case, we randomly picked 6 categories, and 20 images for each of the categories. Note that we also show in the same embedding the positions of the SIFT clusters in circles. The cluster images are generated directly from the gradient intensities specified in the vector. To be specific, the 128-dimensional vector contains 8 gradient values for 16 subwindows of the window of interest. We have plotted these values in roughly corresponding positions in a small image.

Figure 5.4 and Figure 5.5 show the results of using a grid-based embedding on the same data sets. These grid-based SAE embeddings provide a user-friendly image grid. Unlike the unconstrained Euclidean embeddings of Figure 5.2 and Figure 5.3, there is no overlap obscuring some of the images. Yet, the images categories still cluster well into different areas of the space.

5.3.3 kNNA Results

For more quantitative results, we ran SAE, Isomap, LLE and SDE embedding algorithms on 10 randomly chosen objects from the ALOI data set, using 30 images for each object. We set $m = 50$ cluster centers for grouping the SIFT features. We stopped the SAE algorithm when the change in T was below 10^{-6}. The embedding dimension D was set to 2. The simulation annealing temperature parameter β grows with exponent 1.001. We then calculated the

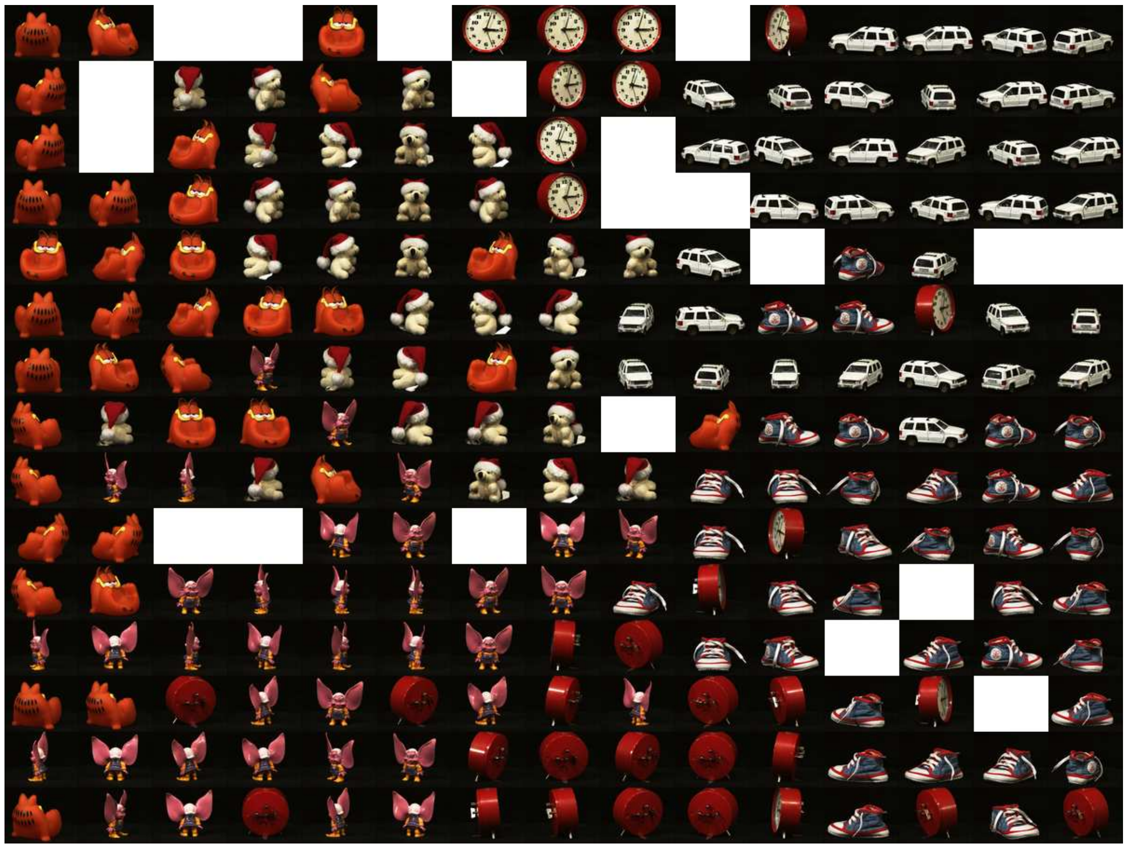


Figure 5.4: Grid embedding for a subset of ALOI.

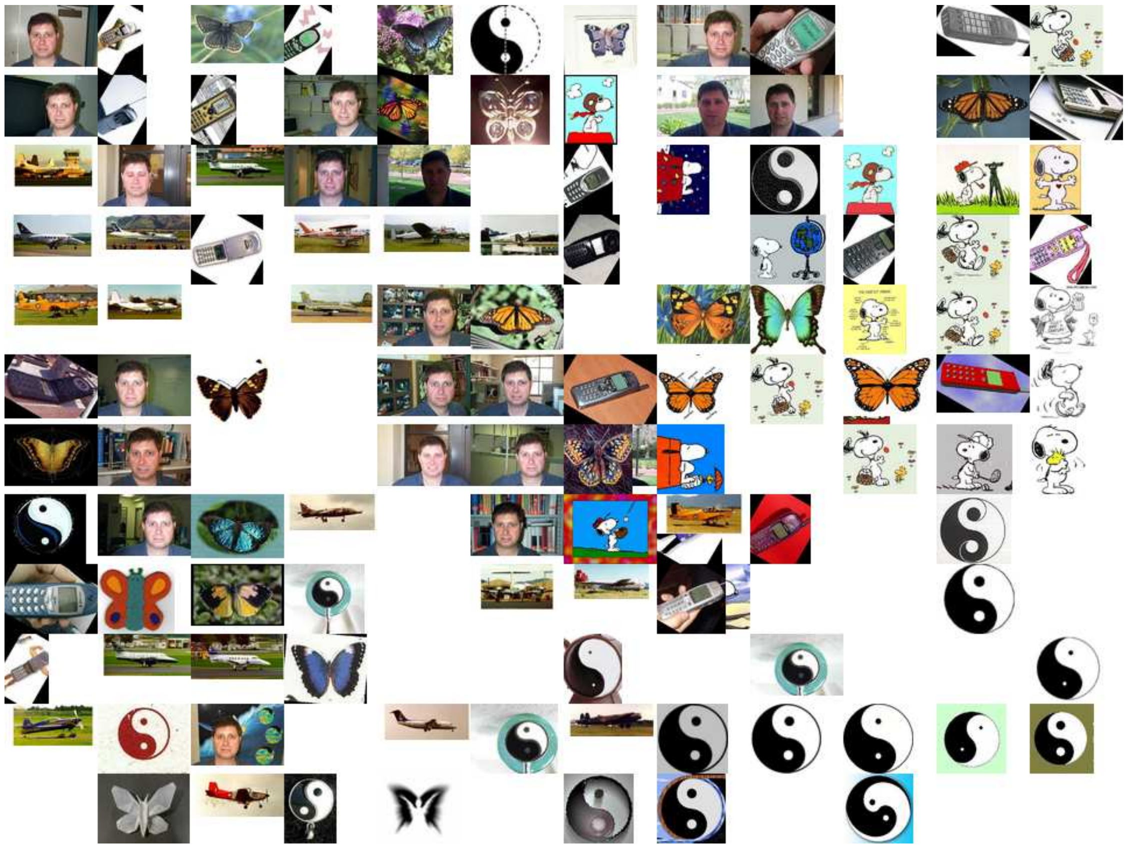


Figure 5.5: Grid embedding for a subset of Caltech101.

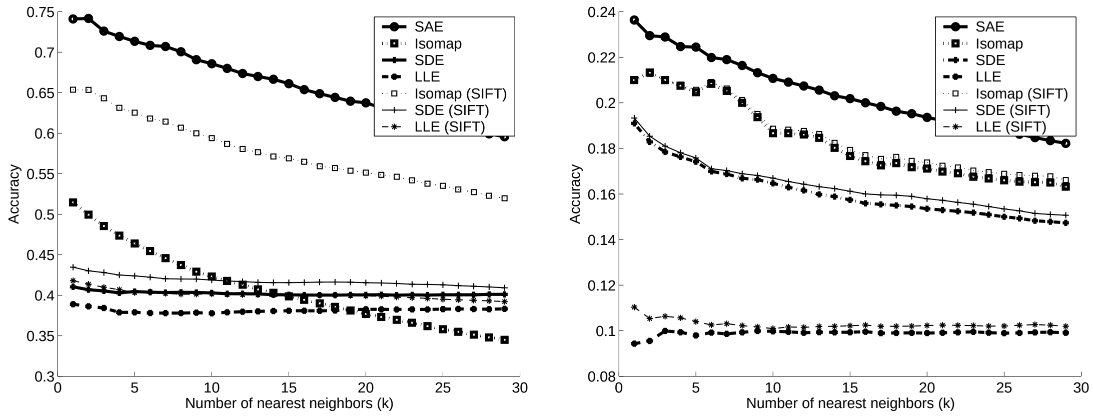


Figure 5.6: kNNA results for ALOI and Caltech101.

kNNA accuracy of the derived embedding for each value of the number of neighbors k. We ran 10 independent experiments (each run randomly drew different objects), and reported the average kNNA values for all the algorithms. The results are shown in Figure 5.6.

With the same settings, we ran similar experiments on the Caltech101 data set with 10 randomly chosen categories, each with 30 images. The kNNA results are shown in Figure 5.6. The images in the Caltech101 data set are of differing sizes. This was not a problem for our algorithm (SAE), but the other algorithms require the images to be of the same size. For the other algorithms, we rescaled all images to a canonical 100-by-100 pixel size.

Our algorithm outperform all the listed competing dimensionality reduction algorithms in terms of kNNA accuracy. Noticeably, although performed on the same SIFT represented data set as our method, the other algorithms still do not perform as well as ours in either data sets.

5.4 Additional Information Sources

Sometimes, there is additional image information available. For example, an image search engine may have the records of user click counts of images for various user queries. Better embeddings can be expected if our algorithm can take into account this information.

Queries are composed of words, so if we treat words in a similar manner as the SIFT-feature clusters in the previous section, we can use the corresponding click counts between images and query words. We get similar “rating” statistics for word-image pairs. We can employ similar embedding algorithm on this rating information instead of using feature-image pairs.

5.4.1 Model Extensions

Our true goal is to use both word-image click counts and feature-image pairs to improve the embedding. In fact, we may have more than two sources of information and more than three types of objects. We thus extend the previous model to allow for more groups of entities beyond the original images and clusters. We use the term “rating” to denote a (possibly incomplete) source of information relating two groups of objects. We allow these ratings to have weights denoting the strength of our belief in the information. More formally, we introduce the following notation.

$\mathbf{I} = \{\mathbf{I}(1), \mathbf{I}(2), \ldots, \mathbf{I}(n)\}$	set of n groups
$\mathbf{I}(i) = \{I_1(i), I_2(i), \ldots, I_{n_i}(i)\}$	all entities in group i
$\pi(i, j)$	weight for the ratings between $I(i)$ and $I(j)$
$r_b^a(i, j)$	rating between $I_a(i)$ and $I_b(j)$
$T_{ij}(\mathbf{I})$	average Kendall's τ for $I(i)$ and $I(j)$

We then define the global Kendall's τ as the weighted average of all possible T_{ij}'s:

$$T(\mathbf{I}) = \sum_{j>i|R(i,j)=1} \pi(i, j) T_{ij}(\mathbf{I}) \ .$$

Note that when $n = 2$, there are two groups of objects, and this is exactly the model we previously presented.

The task if to minimize $T(\mathbf{I})$ with respect to all the image positions $\mathbf{I}$. Similar simulated annealing algorithms can be used to solve this problem.

5.4.2 Click Counts Incorporation

We also tested the value of our model extensions on the click graph (CG) data set. The CG data set is collected from the image search engine of Microsoft. It contains six groups of images: books, mouse, panda, Saturn, shoes, and terrier. There are also click counts available. The click counts record the number of times users selected an image after they issued any particular query to the search engine. We randomly chose 72 images for each of the image groups with 180 queries.

To incorporate the click counts, we make an additional relational group for words. For any click counts between and image I_i and a query Q_q, we add the number of clicks to the image-word count c_{ij} where w_j is a word in Q_q.

Raw click counts are to some extent ill-formed for direct usage. Some image-word pairs may have click counts as large as hundreds, while most of the pairs have small click counts. We use a straight-forward normalization. For each image I_i, we simply normalize all the click counts c_{ij} to be $c_{ij}/\sum_j c_{ij}$ for all the words. This turns out to be effective and useful.

Random Walk Smoothing

In [Craswell and Szummer, 2007], the authors introduced an algorithm that uses a random walk on a Markov chain to model the click counts. We follow their method to smooth the click information.

Let there be n images, and q query words. We form two stochastic matrices: $\mathbf{X}$ and $\mathbf{Y}$ such that $X_{ij} = \frac{c_{ij}}{\sum_j c_{ij}}$, and $Y_{ij} = \frac{c_{ij}}{\sum_i c_{ij}}$. Let the matrix $\mathbf{A}$ be a $n+q$ by $n+q$ transition matrix over both entities (images and query words) such that

$$A_{ij} = \begin{cases} s & \text{if } i = j, \\ (1-s)X_{i,j-n} & \text{if } i \leq n \text{ and } j > n, \\ (1-s)Y_{j,i-n} & \text{if } i > n \text{ and } j \leq n, \\ 0 & \text{otherwise.} \end{cases}$$

Here s is the weight for the Markov chain state's self-transition probability. The smoothed click information for any image I_i is computed using a backward random walk of t steps of this chain,

$$\tilde{\mathbf{c}}_{\mathbf{i}} = \mathbf{A}^t \mathbf{e}_{\mathbf{i}} \ , \tag{5.8}$$

where $\mathbf{e}_{\mathbf{i}}$ is a vector of all zeros except for a one at the ith position. The resulting $\tilde{\mathbf{c}}_{\mathbf{i}}$ is vector of size $n + q$, and the last q component of this vector is the smoothed click information for the image I_i toward all the q query words. We can then use the smoothed click information with the simulated annealing algorithm for the extended model. For our results, we chose $t = 300$ and $s = 0.9$.

Average Accuracy in k Nearest Neighbors

Given an embedding, for any image I_i, since we know the group of the image, we can simply count the number of images k_i from the same group in its k nearest neighbors. The embedding accuracy for I_i is defined as $\frac{k_i}{k}$. The average accuracy of k nearest neighbors (NN accuracy) of the entire system is defined as $\frac{1}{n}\sum_{i=1}^{n}\frac{k_i}{k}$, where n is the total number of images.

Perfect embeddings will have NN accuracy 1 for small k, while random embeddings will have accuracy around the inverse of the number of groups.

Spectral Clustering Accuracy

Another way of evaluating the embeddings is to derive clustering results from them. Let e_1 be the total number of image pairs that are from the same group but clustered in different groups, and let e_2 be the number of image pairs that are from different groups but clustered together. The clustering accuracy is defined as $1.0 - \frac{2(e_1+e_2)}{n(n-1)}$.

We use spectral clustering [Ng et al., 2001] as the basis to perform this task. To be specific, for any embedding, we run spectral clustering based solely on the resulting embedded locations. We tried different possible spectral clustering parameters, and chose the average results for that embedding. Graphs plotting the best possible performance for each run were similar, but more noisy.

Results with Click Counts Information

We set the relative weight between the image-word relation and the image-cluster relation to be 0.2, which means we still concentrate on minimizing the average τ for the image-cluster relationships. Nevertheless, we need to minimize the average τ for image-word relationships as well, since it also contribute a certain weight to the global average Kendall's τ value for the entire system.

Figure 5.7 shows a sample embedding with images from the "panda" category and the related search keywords.

We compare the NN accuracy and spectral clustering for the CG data set: without click information, with smoothed click information, and with un-smoothed click information. The results are shown in Figure 5.8.

Our algorithm achieves higher accuracy results than any of the other three competing algorithms. We believe this is for two reasons.

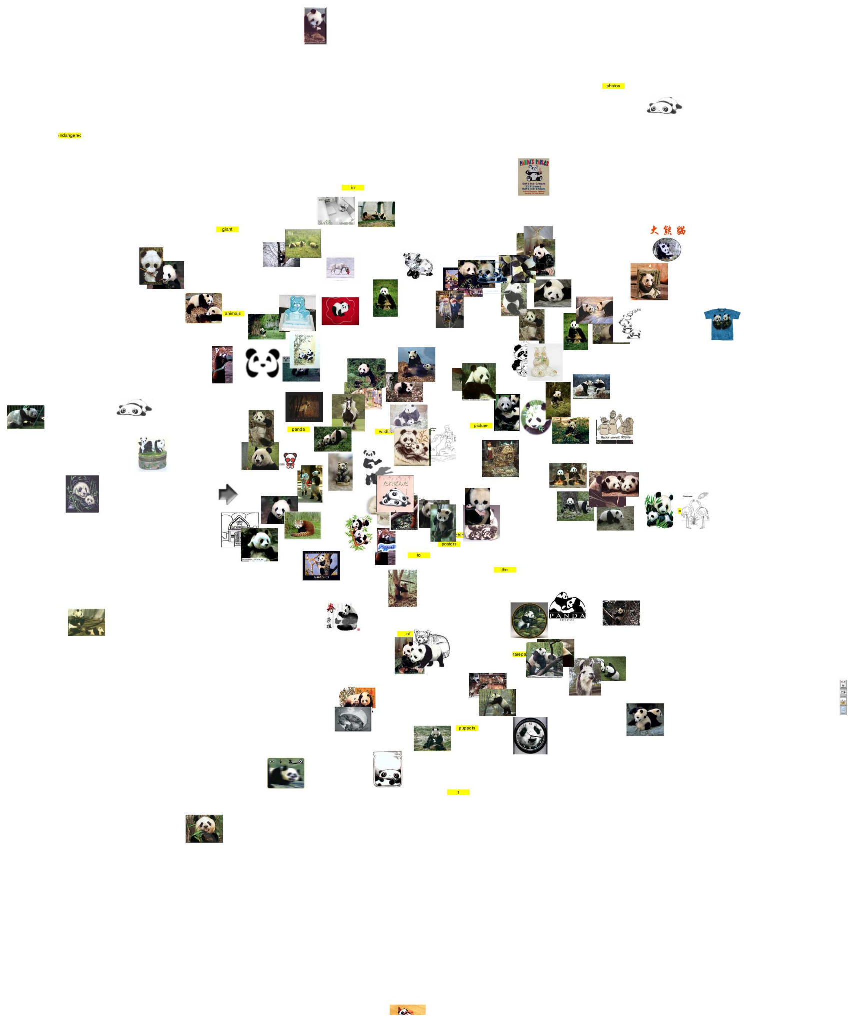
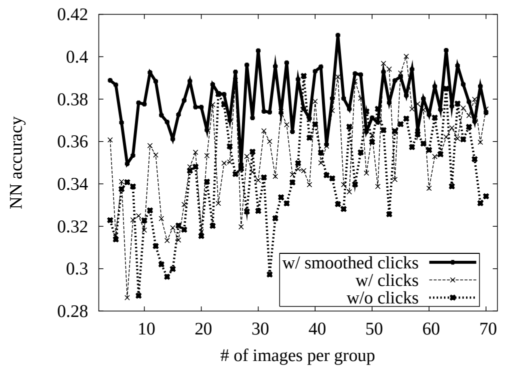


Figure 5.7: Sample embedding for panda images in CG.



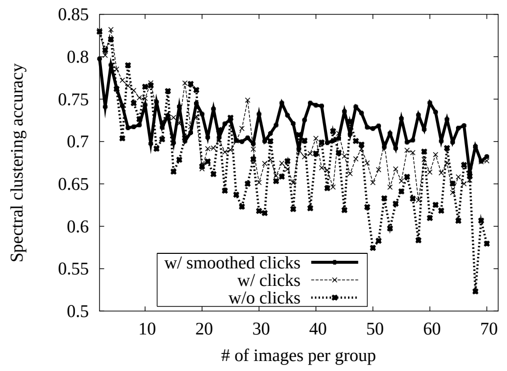


Figure 5.8: NN accuracy (left) and spectral clustering accuracy (right) of the SA results with and without the click information.

First, we exploit the nature of images (by using SIFT features) instead of treating all of the pixels as independent dimensions. When we supply the same SIFT-feature representation to the other embedding algorithms, their performance increases for the ALOI data set. However, we still have better performance, so the use of SIFT features does not entirely explain the results.

Second, our algorithm is designed to cluster images with similar properties (as opposed to find parameters of continuous variation). We feel this is the more important difference. Although our method works for continuous parameter variation (see Figure 5.1) and the dimensionality reduction algorithms have some success in clustering images (see Figure 5.6), our algorithm's strength is in embedding heterogeneous sets of images in which there may not be any continuous path of images leading from one member of the data set to another. For example, there is no "axis of variation" that one can vary to generate a natural smooth set of images from a butterfly image to a cellphone image in the CalTech101 data set. The butterflies and cellphones occupy disconnected regions of the space of images of interest.

For many applications on data sets like CalTech101, we think this strength is particularly important.

Finally, our method has the advantage of being able to generate grid-based embeddings. For some applications, like user interfaces, this is crucial to the utility of the embedding.

It is clear that by incorporating additional click counts information with images, our framework can make use of the information and somewhat boost the testing accuracy.

Chapter 6

Conclusion

We proposed an approach using Bayesian network to perform collaborative filtering. Empirical results show that our filtering algorithm achieve competative accuracy. Our framework can be easily adapted to incorporate additional information other than the ratings (for example game description in the BGG data).

We also formulated a new problem of visualizing collaborative data. This is a potentially very useful problem. Not only are on-line databases of user ratings growing, but personal databases are also becoming more common. We expect the collaborative visualization problem to be useful in organizing personal music or photography collections as well as on-line shopping. We also extended our framework to the domain of image embedding. We gave a complete solution for automatic image embedding problem.

We have not addressed the computational issues nor the stability of the resulting embedding in this work. Both are important problems for on-line deployment in changing databases.

Because of the anytime nature of sampling methods and the ease of introducing constraints, we are hopeful that the solution presented here can be adapted to provide stable and adaptive solutions.

In summary, we presented our solutions to the related problems of collaborative filtering, visualization, and image embedding. They share similar framework of viewing the problems, but each with their own focuses. The methods presented are automatic, without many parameters to tune up with different inputs. Also they are quite efficient in practice, which is essential in some time sensitive application domains.

Bibliography

[Banerjee et al., 2007] Banerjee, A., Dhillon, I., Ghosh, J., Merugu, S., and Modha, D. S. (2007). A generalized maximum entropy approach to bregman co-clustering and matrix approximation. *Journal of Machine Learning Research*, 8:1919–1986.

[Breese et al., 1998] Breese, J., Heckerman, D., and Kadie, C. (1998). Empirical analysis of predictive algorithms for collaborative filtering. In *Uncertainty in Artificial Intelligence*, pages 43–52.

[Chennawasin et al., 1999] Chennawasin, C., Cole, J., and Chen, C. (1999). An emperical study of memory and information retrieval with a spatial user interface. In *21st Annual BCS-IRSG Colloquium on IR*.

[Cox and Cox, 2001] Cox, T. F. and Cox, M. A. A. (2001). *Multidimensional Scaling*. Chapman and Hall/CRC.

[Craswell and Szummer, 2007] Craswell, N. and Szummer, M. (2007). Random walks on the click graph. In *SIGIR*.

[Dagum and Luby, 1993] Dagum, P. and Luby, M. (1993). Approximating probabilistic inference in bayesian belief networks is NPhard. *Artificial Intellige*.

[Dempster et al., 1977] Dempster, A. P., Laird, N. M., and Rubin, D. B. (1977). Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistics Society B*, 39:1–38.

[Donoho and Grimes, 2003] Donoho, D. L. and Grimes, C. (2003). Hessian eigenmaps: New locally linear embedding techniques for high-dimensional data. In *National Academy of Sciences*, volume 100, pages 5591–5596.

[Fei-Fei et al., 2004] Fei-Fei, L., Fergus, R., and Perona, P. (2004). Learning generative visual models from few training examples: an incremental Bayesian approach tested on 101 object categories. In *Computer Vision and Pattern Recognition (CVPR), workshop on Generative-Model Based Vision*.

[Friedman, 1998] Friedman, N. (1998). The Bayesian structural EM algorithm. In *Uncertainty in Artificial Intelligence 14*, pages 129–138.

[Fukumizu et al., 2004] Fukumizu, K., Bach, F. R., and Jordan, M. I. (2004). Kernel dimensionality reduction for supervised learning. In *Advances in Neural Information Processing Systems 16*. MIT Press.

[Fung and Chang, 1989] Fung, R. and Chang, K. (1989). Weighing and integrating evidence for stochastic simulation in Bayesian networks. In *Uncertainty in Artificial Intelligence 5*, pages 209–219. Elsevier Science.

[Geiger et al., 1990] Geiger, D., Verma, T., and Pearl, J. (1990). d-separation: From theorems to algorithms. In *Uncertainty in Artificial Intelligence 5*, pages 139–148.

[Geman and Geman, 1984] Geman, S. and Geman, D. (1984). Stochastic relaxations, Gibbs distributions and the Bayesian restoration of images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 6(6):721–742.

[George and Merugu, 2005] George, T. and Merugu, S. (2005). A scalable collaborative filtering framework based on co-clustering. In *IEEE International Conference on Data Mining 5*, pages 625–628.

[Geusebroek et al., 2005] Geusebroek, J. M., Burghouts, G. J., and Smeulders, A. W. M. (2005). The Amsterdam library of object images. *International Journal of Computer Vision*, 61(1):103–112.

[Gilks et al., 1996] Gilks, W. R., Richardson, S., and Spiegelhalter, D. (1996). *Markov chain Monte Carlo in practice*. Chapman and Hall.

[Globerson et al., 2005] Globerson, A., Chechik, G., Pereira, F., and Tishby, N. (2005). Euclidean embedding of co-occurrence data. In *Advances in Neural Information Processing Systems 17*, pages 497–504.

[Goldberg et al., 2001] Goldberg, K., Roeder, T., Gupta, D., and Perkins, C. (2001). Eigentaste: A constant time collaborative filtering algorithm. *Information Retrieval*, 4(2):133–151.

[Henrion, 1988] Henrion, M. (1988). Propagating uncertainty in Bayesian networks by probabilistic logic sampling. In *Uncertainty in Artificial Intelligence 2*, pages 149–163. Elsevier Science.

[Hofmann, 2004] Hofmann, T. (2004). Latent semantic models for collaborative filtering. *ACM Transactions on Information Systems*, 22(1):89–115.

[Hofmann and Puzicha, 1999] Hofmann, T. and Puzicha, J. (1999). Latent class models for collaborative filtering. In *International Joint Conference in Artificial Intelligence*.

[Huang and Darwiche, 1996] Huang, C. and Darwiche, A. (1996). Inference in belief networks: A procedural guide. 15(3):225–263.

[Huang et al., 2004] Huang, Z., Chen, H., and Zeng, D. (2004). Applying associative retrieval techniques to alleviate the sparsity problem in collaborative filtering. *ACM Transactions on Information Systems*, 22:116–142.

[Ihler et al., 2005] Ihler, A. T., Fischer III, J. W., and Willsky, A. S. (2005). Loopy belief propagation: Convergence and effects of message errors. *Journal of Machine Learning Research*, 6:905–936.

[Jordan and Weiss, 2002] Jordan, M. I. and Weiss, Y. (2002). *Graphical models: probabilistic inference*. MIT Press.

[Kendall, 1955] Kendall, M. G. (1955). *Rank Correlation Methods*. Hafner Publishing Co., New York.

[Kirkpatrick et al., 1983] Kirkpatrick, S., Gelatt, C. D., and Vecchi, M. P. (1983). Optimization by simulated annealing. *Science*, 220(4598):671–680.

[Larraaga et al., 1996] Larraaga, P., Poza, M., Yurramendi, Y., Murga, R. H., and Kuijpers, C. M. H. (1996). Structure learning of Bayesian networks by genetic algorithms: A performance analysis of control parameters. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 18:912–926.

[Leung et al., 2006] Leung, C. W., Chan, S. C., and Chung, F. (2006). A collaborative filtering framework based on fuzzy association rules and multiple-level similarity. *Knowledge and Information Systems*, 10(3):357–381.

[Leung et al., 2007] Leung, C. W., Chan, S. C., and Chung, F. (2007). Applying cross-level association rule mining to cold-start recommendations. In *IEEE/WIC/ACM International Conferences on Web Intelligence and Intelligent Agent Technology - Workshops*, pages 133–136.

[Lewis, 1998] Lewis, D. D. (1998). Naive (bayes) at forty: The independence assumption in information retrieval. In *10th European Conference on Machine Learning*, pages 4–15. Springer Verlag.

[Lowe, 2003] Lowe, D. G. (2003). Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 20:91–110.

[Melville et al., 2002] Melville, P., Mooney, R. J., and Nagarajan, R. (2002). Content-boosted collaborative filtering for improved recommendations. In *Eighteenth National Conference on Artificial Intelligence*, pages 187–192.

[Metropolis et al., 1953] Metropolis, N., Rosenbluth, A. W., Rosenbluth, M. N., Teller, A. H., and Teller, E. (1953). Equation of state calculations by fast computing machines. *Journal of Chemical Physics*, 21:1087–1092.

[Miller et al., 2004] Miller, B. N., Konstan, J. A., and Riedl, J. (2004). Pocketlens: Toward a personal recommender system. *ACM Transactions on Information Systems*, pages 437–476.

[Murphy et al., 1999] Murphy, K. P., Weiss, Y., and Jordan, M. I. (1999). Loopy belief propagation for approximate inference: An empirical study. In *Uncertainty in Artificial Intelligence*.

[Neal and Hinton, 1999] Neal, R. M. and Hinton, G. E. (1999). A view of the EM algorithm that justifies incremental, sparse, and other variants. In Jordan, M. I., editor, *Learning in graphical models*, pages 355–368. MIT Press.

[Ng et al., 2001] Ng, A. Y., Jordan, M. I., and Weiss, Y. (2001). On spectral clustering: Analysis and an algorithm. In *Advances in Neural Information Processing Systems 14*, pages 849–856.

[Pavlov and Pennock, 2002] Pavlov, D. Y. and Pennock, D. M. (2002). A maximum entropy approach to collaborative filtering in dynamic, sparse, high-dimensional domains. In *Advances in Neural Information Processing Systems*, pages 1441–1448. MIT Press.

[Pearl, 1988] Pearl, J. (1988). *Probabilistic Reasoning in Intelligent Systems*. Morgan Kauffman.

[Pearson, 1901] Pearson, K. (1901). On lines and planes of closest fit to systems of points in space. *Philosophical Magazine*, 2:559–572.

[Pennock et al., 2000] Pennock, D., Horvitz, E., Lawrence, S., and Giles, C. L. (2000). Collaborative filtering by personality diagnosis: A hybrid memory- and model-based approach. In *Uncertainty in Artificial Intelligence*, pages 473–480.

[Pernkopf and O'Leary, 2003] Pernkopf, F. and O'Leary, P. (2003). Floating search algorithm for structure learning of Bayesian network classifiers. *Pattern Recognition Letters*, 24:2839–2848.

[Porter, 1980] Porter, M. F. (1980). An algorithm for suffix stripping. *Program*, 14(3):130–137.

[Rennie and Srebro, 2005] Rennie, J. D. M. and Srebro, N. (2005). Fast maximum margin matrix factorization for collaborative prediction. In *International Conference on Machine Learning*, pages 713–719.

[Roweis and Saul, 2000] Roweis, S. and Saul, L. (2000). Nonlinear dimensionality reduction by locally linear embedding. *Science*, 290(5500):2323–2326.

[Sarwar et al., 2001] Sarwar, B., Karypis, G., Konstan, J., and Riedl, J. (2001). Item-based collaborative filtering recommendation algorithms. In *International World Wide Web Conference 10*, pages 285–295.

[Saul and Roweis, 2003] Saul, L. K. and Roweis, S. T. (2003). Think globally, fit locally: Unsupervised learning of low dimensional manifolds. *Journal of Machine Learning Research*, 4:119–155.

[Scholkopf et al., 1998] Scholkopf, B., Smola, E., and Muller, K. (1998). Nonlinear component analysis as a kernel eigenvalue problem. *Neural Computation*.

[Scovanner et al., 2007] Scovanner, P., Ali, S., and Shah, M. (2007). A 3-dimensional SIFT descriptor and its application to action recognition. In *15th international conference on Multimedia*, pages 357–360.

[Se et al., 2001] Se, S., Lowe, D., and Little, J. (2001). Vision-based mobile robot localization and mapping using scale-invariant features. In *IEEE International Conference on Robotics and Automation*.

[Shachter and Peot, 1989] Shachter, R. D. and Peot, M. A. (1989). Simulation approaches to general probabilistic inference on belief networks. In *Uncertainty in Artificial Intelligence 5*, pages 221–231. Elsevier Science.

[Shannon, 1948] Shannon, C. E. (1948). A mathematical theory of communication. *Bell System Technical Journal*, 27:379–423, 623–656.

[Tenenbaum et al., 2000] Tenenbaum, J. B., de Silva, V., and Langford, J. C. (2000). A global geometric framework for nonlinear dimensionality reduction. *Science*, 290:2319–2323.

[Torgerson, 1952] Torgerson, W. (1952). Multidimensional scaling: I. theory and method. *Psychometrika*, 17:401–419.

[Weinberger and Saul, 2006] Weinberger, K. Q. and Saul, L. K. (2006). Unsupervised learning of image manifolds by semidefinite programming. *International Journal of Computer Vision*, 70:77–90.

[Williams, 2001] Williams, C. K. I. (2001). On a connection between kernel PCA and metric multidimensional scaling. In *Advances in Neural Information Processing Systems 13*, pages 675–681. MIT Press.

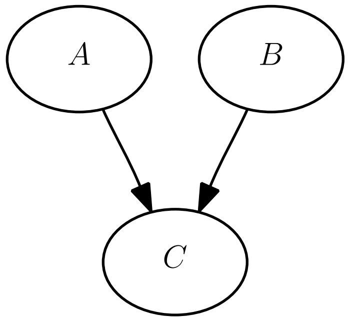
Appendix A

Bayesian Networks

Bayesian networks are the most widely used graphical models to represent factorized probability distribution over multiple random variables. We introduce the basic components of a Bayesian network in Section A.1, and then introduce the two key problems associated with Bayesian networks: inference (Section A.2) and learning (Section A.3).

A.1 Introduction to Bayesian Networks

A Bayesian network $\mathcal{B}$ is composed of its graphical structure $\mathcal{G}$ and its parameter set Θ. $\mathcal{G}$ is always a directed acyclic graph (DAG) over the set of variables $\mathbf{X} = \{X_1, X_2, \ldots, X_n\}$. The variable X_i and all its parents $\mathbf{Pa}(X_i)$ in $\mathcal{G}$ form the scope of family ϕ_{X_i}. A conditional probability distribution (CPD) $P(X_i \mid \mathbf{Pa}(X_i))$ is associated with family ϕ_{X_i}. The joint



$A=0$	$A=1$
0.2	0.8

$B=0$	$B=1$
0.6	0.4

A	B	$C=0$	$C=1$
0	0	0.1	0.9
0	1	0.9	0.1
1	0	0.8	0.2
1	1	0.05	0.95

Table A.1: A simple Bayesian network.

distribution $P(\mathbf{X})$ defined by the Bayesian network factors by each individual family

$$P(X_1, X_2, \ldots, X_n) = \prod_{i=1}^{n} P(X_i \mid \mathbf{Pa}(X_i)) \ , \qquad \text{(A.1)}$$

where if X_i has no parent in $\mathcal{G}$, $P(X_i \mid \mathbf{Pa}(X_i)) \equiv P(X_i)$.

Consider a simple Bayesian network with three random variables A, B, and C in Table A.1. The structure of the network and sample conditional probability distributions (CPDs) are shown in the table. The joint distribution can be thus factored as $P(A, B, C) = P(A)P(B)P(C \mid A, B)$. For example, the probability $P(A = 0, B = 1, C = 1) = P(A = 0)P(B = 1)P(C = 1 \mid A = 0, B = 1) = 0.2 \times 0.4 \times 0.1 = 0.008$.

A.1.1 Independence Assumptions

The structure of a Bayesian network encodes a set of conditional independence assumptions for the random variables. Analyzing the independence properties can help factorize the joint distribution into much simpler marginal distributions, which is the key benefit of using graphical models.

Conditional independence assumptions can be determined by the *d-separation* properties of the graph. Two sets of nodes $\mathbf{X}$ and $\mathbf{Y}$ are conditionally independent given a third set of nodes $\mathbf{Z}$ if they are d-separated in the graph structure. d-separation is further defined with *active trails*. A trail p from node A to node B in the graph is a path from the undirected version of the Bayesian network structure that connects node A and B. The trail is active given $\mathbf{Z}$ if A and B are directly connected, or all triples $U - V - W$ along p meet the following relevant conditions:

- if $U \rightarrow V \rightarrow W$ then $V \notin \mathbf{Z}$
- if $U \leftarrow V \rightarrow W$ then $V \notin \mathbf{Z}$
- if $U \rightarrow V \leftarrow W$ then V or any descendant of V is in $\mathbf{Z}$

$\mathbf{X}$ is d-separated from $\mathbf{Y}$ given $\mathbf{Z}$ if there exists no active trails from any node in $\mathbf{X}$ to any node in $\mathbf{Y}$ when $\mathbf{Z}$ is given. d-separation can be computed in linear time using a depth-first algorithm [Geiger et al., 1990].

We next discuss two key problems for Bayesian networks: inference and learning.

A.2 Inference for Bayesian Networks

Inference for Bayesian networks is the problem of finding the marginal probability of some variables $\mathbf{Y} \subset \mathbf{X}$ given some evidence $\mathbf{e}$: $P(\mathbf{Y} \mid \mathbf{e})$. Evidence is the assignment of particular values to a subset of the variables of $\mathbf{X}$.

In general, the inference problem is NP-hard. The complexity of the problem is determined by the Bayesian network structure, and it is exponential in the tree width of the graph[1] [Dagum and Luby, 1993]. Nevertheless, in practice, exact inference algorithms such as variable elimination and clique tree algorithms [Huang and Darwiche, 1996] are still very useful and efficient for smaller networks or Bayesian networks with simple structures.

Approximate inference algorithms are used when exact inference is intractable or too costly to use. Sampling algorithms such as importance sampling and Markov chain Monte Carlo methods such as Metropolis Hastings [Gilks et al., 1996] and Gibbs sampling [Geman and Geman, 1984] are often used. Direct sampling method is first used in [Henrion, 1988] for Bayesian networks. In [Fung and Chang, 1989], likelihood weighting is used to ensure that no sample is rejected. In [Shachter and Peot, 1989] the authors first presented importance sampling for Bayesian networks.

Loopy belief propagation (LBP) [Murphy et al., 1999] is another widely applicable approximate inference algorithm. In LBP, it first construct clusters of the Bayesian network similar to the clique tree algorithm. One major difference is that the resulting cluster graph need not to be restricted to a tree structure, moreover, the graph can be arbitary and can even

[1]Unless $P = NP$.

have loops. This relaxation on the cluster graph can lead to much simler clusters. The LBP algorithm also use similar message passing schemes as the clique tree algorithm, with the difference in that it may take more than two passes to make the entire graph to converge. In [Ihler et al., 2005], the details about LBP's convergence issues and approximation errors are discussed.

A.3 Learning of Bayesian Networks

Another key problem related with Bayesian networks is learning. More specifically, given a collection of variable instantiations, the learning problem is to find the most suitable Bayesian network structure and corresponding parameters that best explains the evidence.

It is often the case that the structure of the Bayesian network is already given. Only the parameters (CPDs of all the families in the network) need estimation. For fully observed data (data with instantiations of all the variables), maximum likelihood estimation simply reduces to counting the number of co-occurrences (called the *sufficient statistics*) of X_i and $\mathbf{Pa}(X_i)$, and then setting the CPD parameters to be the corresponding empirical ratios. For partially observed data, the expectation maximization (EM) algorithm [Dempster et al., 1977] is usually used to estimate parameters.

For the case when the structure of the Bayesian network also need to be learned, the problem becomes very complicated. First, a score function must be used to evaluate the goodness of a given structure. Second, since there are super-exponential number of possible structures, some structural search algorithms should be employed to search over the possible structures. In [Friedman, 1998], structural expectation maximization (SEM) is proposed to learn both structure and parameters of a Bayesian network when where is missing data. [Pernkopf and O'Leary, 2003] presents a floating search approach for learning the network structure. In [Larraaga et al., 1996], the authors give an approach for structural learning using genetic algorithms.

A.3.1 Sufficient Statistics

Sufficient statistics (SS) are one of the most important concepts in parameter estimation. SS are derived directly from observed data, and are sufficient to estimate the model parameters without the need to refer to the data again.

For example, to estimate the probability of a head when tossing a particular coin, the SS are the number of times that the coin landed head, and total number of times it has been tossed. Once we have these numbers, they are sufficient to estimate the required parameters.

When it comes to a problem with partially observed data, we are no longer able to compute the sufficient statistics directly. Instead, we compute the expected sufficient statistics (ESS) of the partially observed data: the expectation under a particular distribution of SS among all possible data completions.

A.3.2 Expectation Maximization

The expectation maximization (EM) algorithm works by repeatedly iterating the following two phases:

- **E step:** using current estimation of the model Θ^k, compute the expected sufficient statistics (ESS),
- **M step:** update the parameter estimation to Θ^{k+1} by using the ESS derived.

By iteratively repeating the EM steps, the likelihood of the data given the current model parameters is guaranteed to be non-decreasing. Each iteration of EM will result in a better model until convergence [Neal and Hinton, 1999]. Although the EM algorithm does not necessarily fall into global maximum of likelihood, in practice, it is quite stable. We usually start the EM procedure with some randomly chosen initial parameters.

A.3.3 Parameter Estimation for Bayesian networks

For Bayesian networks, the sufficient statistics (SS) needed for estimating its parameters are described as following. In the case of complete data, we have samples (instantiations) of all the variables $\mathbf{X} = \{X_1, \ldots, X_n\}$ in the network. For each variable X_i, to estimate its parameter $\theta_{X_i|Pa(X_i)}$, the SS needed is $M[X_i = x, Pa(X_i) = \mathbf{u}]$: total number of times that X_i is x and its parent takes the values $\mathbf{u}$ in the entire data. The estimation of the parameter is

$$\hat{\theta}_{X_i|Pa(X_i)}(x \mid \mathbf{u}) = \frac{M[X_i = x, Pa(X_i) = \mathbf{u}]}{\sum_y M[X_i = y, Pa(X_i) = \mathbf{u}]} \ . \tag{A.2}$$

For data with missing values, EM algorithm is used. In this case we can no longer derive the SS directly. In the expectation step, we use the current Bayesian network parameters and use inference algorithm to derive the expected sufficient statistics (ESS) $\bar{M}[X_i = x, Pa(X_i) = \mathbf{u}]$. This step can be costly. In the maximization step, the updating rule is the same as in Equation A.2.