Chapter 1
Clustering Multiple Graphs – an NMF approach

1.1 The Graph Clustering Problem

A collection of weighted graphs on n vertices often arises as a "histogram" of interaction events between n actors in terms of their interaction frequencies observed over periods of time. In particular, for each $t = 1, \ldots, T$, the (i, j)th entry $G_{ij,t}$ of G_t encodes the number of times that person i and person j interacted during tth period. This section considers a problem of grouping multiple graphs into few clusters. To give a more precise description of a "graph clustering" problem, consider $(\kappa(1), G_1), \ldots, (\kappa(T), G_T)$ be an (independent) sequence of pairs of a class label $\kappa(t)$ and a (potentially weighted) graph G_t on n vertices. We assume that the class label $\kappa(t)$ takes values in $\{1, \ldots, K\}$ and also that given $\kappa(t) = k$, each G_t is a random graph on n vertices whose distribution depends only on the value of k.

1.2 Theory : NMF, ARI, AIC – what is this?

In non-negative matrix factorization, we begin with a matrix, $A \in \mathbb{R}^{n \times T}$ and seek to approximate A by the product of two non-negative matrices $W \in \mathbb{R}^{n \times r}$ and $H \in \mathbb{R}^{r \times T}$, where r is the rank of the approximation. There are several choices of loss-function that determine the strength of the approximation, and each gives rise to a different factorization. Common choices are divergence:

$$D(A||WH) = \sum_{i,j} A_{i,j} \log \left(\frac{A_{i,j}}{WH_{i,j}} \right) - A_{i,j} + WH_{i,j}$$

and squared loss:

$$\|A - WH\|_F^2,$$

where

$$\|B\|_F^2 = \sum_{i,j} B_{i,j}^2.$$

Note that the divergence equals the Kullback-Liebler divergence when A and WH are doubly stochastic.

Actual minimization of the loss functions can be difficult for various reasons. In the squared loss, the objective is non-convex and has non-unique minimizers. Further, since the problems have constraints which are generally active at the solution, the constraints must be considered while performing the optimization. Lee and Seung propose multiplicative update rules based on the steepest descent direction with a step-size chosen to allow for multiplicative updates that are guaranteed to result in a non-increasing loss function value. This ensures that if a non-negative factorization was chosen initially, the non-negativity constraints would automatically be met [2]. Another popular update rule is "Alternating Least Squares," where the squared loss problem is solved in W and H in alternating steps. These solutions are easy to obtain by standard least-squares procedures [3].

The attractiveness of non-negative factorization can be seen in the applications, where we are able to give an interpretation of W (basis) and H (weights). Here, we present a simple example, where we form a matrix by taking its columns to be the weighted sum of basis vectors.

Here we use the R package "NMF" [1] to showcase how to use non-negative factorization on a toy problem. First, we build a matrix by drawing its columns from one of two distributions:

Listing 1.1 Making a matrix easily factorizable by NMF

```
>basis <- matrix( c(1,2,3,4,
>                   4,3,2,1),4,2)
> A <- matrix(0,4,3)
> A[,1] <- .9*basis[,1]+.1*basis[,2]
> A[,2] <- .5*basis[,1]+.5*basis[,2]
> A[,3] <- .1*basis[,1]+.9*basis[,2]
```

Next, we apply NMF to it and note the near-zero error:

Listing 1.2 Factorizing a matrix by NMF

```
> library('NMF')
> theNmf <- nmf(A, rank=2,method = 'lee')
> sum(abs(basis(theNmf) %*% coef(theNmf) - A))
[1] 1.145542e-08
```

Here we introduce a measure for similarity between clusterings, the adjusted rand index (ARI) [**?**]. Basically, the ARI is calculated by summing up the number of agreements between two clusterings (when pairs of objects are co-clustered in both or are not co-clustered in both) and subtracting off the expected number of agreements if the two clusterings were formed using a generalized hypergeometric distribution and are drawn with fixed number of clusters and cluster sizes. Finally, we divide by the total number of pairs minus the same expected number. The ARI is at most 1, indicating a perfect matching between clusterings. When a clustering

is compared to the truth, an ARI value of 0 indicates that the clustering is no better than chance.

Using our confusion matrix above, we calculate the ARI between the truth and the baseline:

$$ARI = \frac{\binom{9}{2} + \binom{10}{2} - \left(2\binom{10}{2}(\binom{9}{2} + \binom{11}{2})\right)/\binom{20}{2}}{\frac{1}{2}\left(2\binom{10}{2} + \binom{9}{2} + \binom{11}{2}\right) - \left(2\binom{10}{2}(\binom{9}{2} + \binom{11}{2})\right)/\binom{20}{2}} \approx 0.80$$

Our model selection information criteria is defined as follows:

$$AICc = \text{Negative Log Likelihood} + \text{Penalty Term},$$

where the likelihood is specified by Poisson densities and the penalty term is specified by quantities that grows with the number of parameters. More specifically,

$$\texttt{-loglike} = \sum_{t=1}^{T}\sum_{ij}(X_{ij,t}/N_t)\log(X_{ij,t}/Nt) \tag{1.1}$$

$$\texttt{penalty} = \sum_{k=1}^{r}\frac{1}{\widehat{N}_k}\left(\sum_{ij}\mathbf{1}\{\widehat{W}_{ij,k} > 0\} - 1\right). \tag{1.2}$$

We make some observation of the penalty term. First, for a graph that is "undersampled" (i.e., N_t is small), it is difficult for it to be a "stand-alone motif" graph. Next, a set of motifs such that $\widehat{W}_{\cdot,k_1}$ and $\widehat{W}_{\cdot,k_2}$ share "many" common support is not favored.

1.3 Examples

1.3.1 Example 1 – Wikipedia in English and French

A tensor is a mathematical term for a multi-way array. For instance, a 3-way array, i.e. $\boldsymbol{X} = (X_{ijk})$, is a tensor, and in particular, a sequence of adjacency matrices has a natural tensor representation. More explicitly, for a pair of graphs, say, $F = (F_{ij})$ and $E = (E_{ij})$ on n vertices, one can associate it with the tensor X, where for $k = 1$, $X_{ij1} = F_{ij}$ and for $k = 2$, $X_{ij2} = E_{ij}$. Then, the mode ℓ matrization of the tensor is a matrix version M of the tensor, where the ℓth column of M is a vectorization of the matrix obtained by fixing the ℓth index to be a particular value.

For example, let us consider a problem of deciding if two matrices F and E are similar or not, where n vertices are a page in the Wikipedia and F_{ij} and E_{ij} indicate whether or not page i and page j are linked. In the listing below, `FrWiki` and `EnWiki` corresponds, respectively, to F and E in our text, and henceforth, we take F and E to be as such.

Listing 1.3 Building a data matrix from the Wikigraphs

```
> data(FrenchWiki)
> data(EngishWiki)
> require(abind)
> X = aBind(FrWiki, EnWiki, along=3)
> M = cbind(as.vector(FrWiki), as.vector(EnWiki))
```

Wikipedia is an open-source Encyclopedia that is written by a large community of users (everyone who wants to, basically). There are versions in over 200 languages, with various amounts of content. The full data for Wikipedia are freely available for download. A Wikipedia document has one or more of: title, unique ID number, text the content of the document, images, internal links links to other Wikipedia documents, external links links to other content elsewhere on the web, and language links links to the same document in other languages. The multilingual Wikipedia provide a good testbed for developing methods for analysis of text, translation, and fusion of text and graph information.

Naturally, there are plenty of similarities between E and F since the connectivity between a pair of pages is driven by the relationship between topics on the pages. Nevertheless, E and F are different, i.e., $\|E-F\|_F^2 > 0$, since the pages in Wikipedia are grown "organically", i.e., there is no explicit coordination between English Wikipedia community and French Wikipedia community that try to enforce the similarity between E and F. To answer the question, we fit the model using the inner dimension to be 1 and then 2 using `gclust.rsvt`, and then compute their information criteria using `getAICc`:

Listing 1.4 Computing AICc for comparing the Wikigraphs

```
> gfit1 = gclust.rsvt(M,1)
> gfit2 = gclust.rsvt(M,2)
> gic1 = getAICc(gfit1)
> gic2 = getAICc(gfit2)
```

The numerical results relevant to answering our decision problem are reported in Table 1.1. In particular, because the AIC value `gic1` for $r = 1$ is lower than the AIC value `gic2` for $r = 2$, our analysis suggests that E and F have the same connectivity structure.

Now, it is fair to ask the meaning of the claim that E and F have the same connectivity structure. This is very much connected with the formulation of `getAICc`, and without going into the full details, we will say that E and F are the same provided that for each i and j,

$$\frac{\mathbf{E}[E_{ij}|s(E)]}{s(E)} = \frac{\mathbf{E}[F_{ij}|s(F)]}{s(F)}, \tag{1.3}$$

where $s(E)$ and $s(F)$ denotes the total edge weight of E and F respectively, i.e., $s(E) = \mathbf{1}^\top E\mathbf{1}$ and $s(F) = \mathbf{1}^\top F\mathbf{1}$. Alternatively, the equality in (1.3) can be restated as a hypothetical question about the probability, conditioning that a new link

is formed, of the link being from page i to page j, and about whether or not the probability being the same for English Wikipedia and French Wikipedia.

A notable consequence of our framework for comparing two graphs is that even for the case where $s(E)$ is much bigger that $s(F)$, our analysis framework still permit us to conclude that E and F are the same. In words, if we have concluded that E and F are different, then one could say that not only that the cross-reference *intensity* in two language is different but also that the connectivity *structure* is different in two languages. In our earlier example, since $r = 1$ is more favorable, one can say that (despite the fact that $s(E) > s(F)$), E and F have the same connectivity structure.

Table 1.1 Are French and English Wikigraphs similar? The answer using `gclust.rsvt` provides an evidence supporting the conclusion *Yes*.

nclust	negloglik	penalty	AIC
1	43.05	1.525	44.58
2	41.65	4	45.65

1.3.2 Example 2 – Wearable RFID Sensors

Wearable RFID sensors are used to detect close-range interactions between individuals in the geriatric unit of a university hospital. The study involves 46 health care workers and 29 patients over the span of 4 days and 4 nights (from Monday through Thursday). It is reported there that "the contact patterns were qualitatively similar from one day to the next", and in this example, we reexamine this statement through our procedure.

For simplicity, individuals were grouped in four classes according to their role in the ward: patients (PAT), medical doctors (physicians and interns, MED), paramedical staff (nurses and nurses' aides, NUR) and administrative staff (ADM). As such, the data naturally lends itself to a block structured graph.

Using the vertex contraction that groups the actors by their role (i.e., PAT, ADM, MED, NUR), we construct a collection of 4×4 weighted adjacency matrices. Dividing the entire duration to four intervals (i.e., by day), we arrive at a collection of four weighted adjacency matrices, each accounting for a 24-hour period.

From Figure 1.2, we make some observations that help us clustering of four graphs. First, only during the first day, NUR and MED interaction rate is high. Only during the third and fourth days, there is zero incident of PAT detecting MED. All four days, NUR-¿NUR interaction remains intense. Except on the first day, NUR-PAT interaction remains intense. These observations suggest that there are three clusters, where each of the first and the second graph constitute its own cluster, and the third and four graphs form the third cluster.

Listing 1.5 Computing AICc for the four-period version

```
> data(RFIDsensors)
```

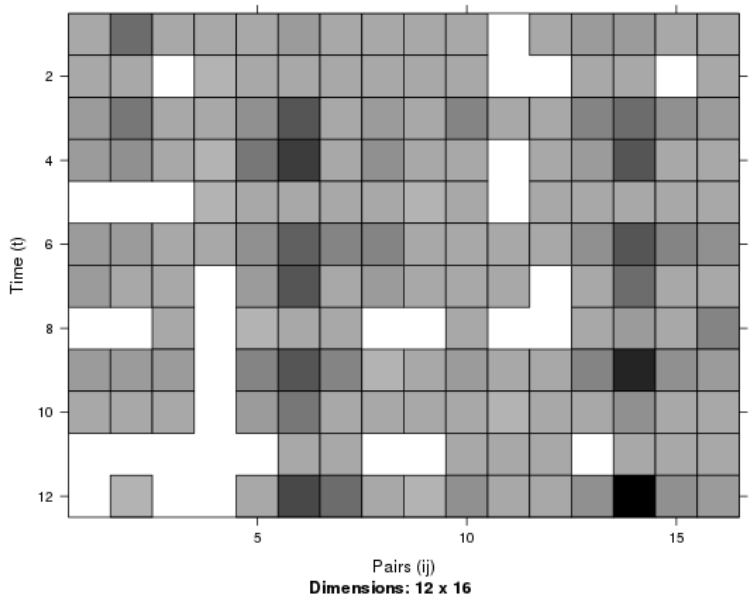

Fig. 1.1 Wearable Sensor Data as a 12×16 matrix, i.e., $\texttt{M12}^\top$. Each row of $\texttt{M12}^\top$ for each of the twelve periods, and each column of $\texttt{M12}^\top$ for each entry of 4×4 matrix.

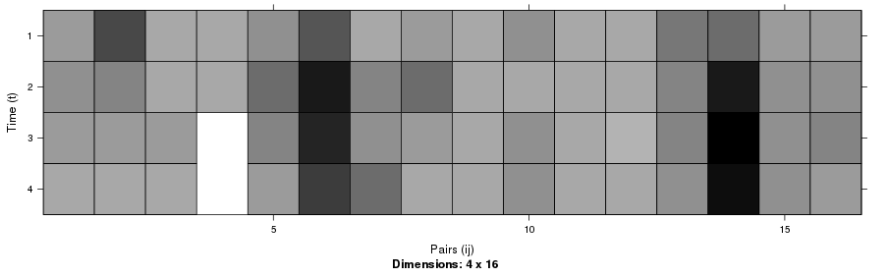

Fig. 1.2 Wearable Sensor Data as a 4×16 matrix, i.e., $\texttt{M4}^\top$. The tth row of $\texttt{M4}^\top$ corresponds to the aggregation of the rows from the $(1+3\times(t-1))$th row to $3\times t$th row of $\texttt{M4}^\top$. For example, the first row of $\texttt{M4}^\top$ corresponds to the first three rows of $\texttt{M12}^\top$

```
> M4 = sapply(RFIDsensors[[1]], as.vector)
> gic1 = getAICc(gclust.rsvt(M4,1))
> gic2 = getAICc(gclust.rsvt(M4,2))
> gic3 = getAICc(gclust.rsvt(M4,3))
> gic4 = getAICc(gclust.rsvt(M4,4))
```

As shown in Table 1.2, the value of `gic3` is the smallest among the four possible choices, and four and twelve respectively (c.f. and Table 1.3). The general conclusion that one can draw from this observations is that there does not seem to exist any recurring pattern.

Table 1.2 Does the contact pattern differ from day to day?

$\widehat{d}$	NegLogLik	Penalty	AICc
1	18.1841	0.0010	18.1851
2	17.7843	0.0042	17.7885
3	**17.6532**	**0.0095**	**17.6627**
4	17.6720	0.0168	17.6887

Table 1.3 Does the contact pattern changes every eight hours? (Top four choices)

$\widehat{d}$	NegLogLik	Penalty	AICc
9	48.46202	0.1018	48.56385
10	48.51386	0.1386	48.65249
11	48.23954	0.1526	48.39221
12	**48.10388**	**0.1876**	**48.29151**

Listing 1.6 Fitting a three-cluster model on the four-period and on the 12-period version

```
> M12 = sapply(RFIDsensors[[2]],as.vector)
> gfit.M4 = gclust(M4,3)
> gfit.M12 = gclust(M12,3)
> matplot(t(gfit.M4$H), type='b',cex=c('A','B','C'))
> matplot(t(gfit.M12$H),type='b',cex=c('a','b','c','d'))
```

One can ask how the 12 period analysis compares with the 4 period version. For this, one can force on a 12 period the three motif clustering, and search therein for a pattern consistent with the result from 4 period analysis. As reported in Table 1.4, the two clustering is consistent with each other. In summary, the suggested conclusion is that the only pair that has the similar interaction pattern is the Wednesday-Thursday and Thursday-Friday pair, and this is indicated by 'C' appearing twice in Table 1.4. While the 12 period analysis yields the best clustering is each graph being its own cluster, the best 3 clustering of the 12 graphs shows that C is represented by the sequence (a,b,a).

1.3.3 Example 3 – C. elegans' chemical and electric pathways

Consider graphs based on n neurons, where each edge weight is associated with the functional connectivity between neurons. The area of studying such a graph for

Table 1.4 How do fitting a three-cluster model on the daily version and on the 8-hour version correspond to each other? The letters A, B and C code three clusters for the daily version and the letters a, b and c code three clusters for the 8-hour version.

Num. of Graphs	Day 1	Day 2	Day 3	Day 4
4	A	B	C	C
12	(c,c,a)	(a,a,a)	(a,b,a)	(a,b,a)

further expanding our knowledge of biology is called "connectome". In this section, we introduce such graphs, first being the chemical path way connectivity, and the second being the functional path way connectivity, and consider how we can answer a simple connectom question using `gclust` and `getAICc`.

Listing 1.7 C. elegans' chemical and eletrical pathways

```
> load('celegan')
> AeAc = sapply(list(Ac,Ae) ,as.vector)
> AeAc.gic1=getAICc(gclust.rsvt(AeAc,1,method='lee'))
> AeAc.gic2=getAICc(gclust.rsvt(AeAc,2,method='lee'))
>
> vcmat = t(apply(diag(9),2,rep,times=c(rep(30,8),39)))
> Ae.vc = vcmat %*% Ae %*% t(vcmat)
> Ac.vc = vcmat %*% Ac %*% t(vcmat)
> AeAc.vc = sapply(list(Ae.vc,Ac.vc),as.vector)
>
> AeAc.vc.gic1=getAICc(gclust.rsvt(AeAc,1,method='lee'))
> AeAc.vc.gic2=getAICc(gclust.rsvt(AeAc,2,method='lee'))
```

The AIC values for $\hat{d} = 1$ and $\hat{d} = 2$ are respectively 30.12 and 33.31, whence this leads us to a suggestion that the connectivity structure of `Ae` and `Ag` could be the same (up to some random noise). A more careful statement of this can be stated as follows:

$$\mathbf{E}[(\texttt{Ae})_{ij}] = \mathbf{E}[(\texttt{Ac})_{ij}] \qquad \text{for each pair } ij. \tag{1.4}$$

A way to cross-check the claim that $\hat{d} = 1$ is to utilize the concept of *vertex contraction* that we have seen earlier. The main idea is simple, and we motivate the main idea by way of a overly simplified example. Consider two vectors p and q of positive integers. Then, if p and q are identical, then $\mathbf{1}^\top p = \mathbf{1}^\top q$. On the other hand, even if p and q are different, it is possible that $\mathbf{1}^\top p = \mathbf{1}^\top q$, e.g., consider

$$p = \begin{pmatrix} 1 \\ 2 \end{pmatrix} \text{ and } q = \begin{pmatrix} 2 \\ 1 \end{pmatrix}. \tag{1.5}$$

In other words, if $\boldsymbol{\mu}_e = \mathbf{E}[\texttt{Ae}]$ and $\boldsymbol{\mu}_c := \mathbf{E}[\texttt{Ac}]$ are the same, then it follows that for any matrix Q,

$$Q\boldsymbol{\mu}_c Q^\top = Q\boldsymbol{\mu}_e Q^\top \tag{1.6}$$

Listing 1.8 C. elegans' chemical and electrical pathways - Vertex Contraction

```
> Q = t(apply(diag(9),2,rep,times=c(rep(30,8),39)))
> Ae.vc = Q %*% Ae %*% t(Q)
> Ac.vc = Q %*% Ac %*% t(Q)
> AeAc.vc = sapply(list(Ae.vc,Ac.vc),as.vector)
```

In Listing 1.8, the same vertex contraction is performed on both `Ae` and `Ac`, where the vertex contraction matrix Q is such that each of the first eight groups of thirty vertices is aggregated (collapsed) to a single vertex, and for the last thirty-nine vertices is aggregated to a single vertex. Performing our procedure to the collapsed graphs yields that the AIC values for $\hat{d} = 1$ and $\hat{d} = 2$ are 15.84 and 15.61, suggesting that there is two patterns. Performing a Monte Carlo experiment using 100 random choices for such partitions, we obtain Table

Table 1.5 Monte Carlo experiments involving 100 random vertex contraction

$\widehat{d}$	1	2
Frequency	3	97

Next, let us consider two alternative vertex contraction "policies" coded in Listing 1.9 and Listing 1.10. In Listing 1.10, 279 neurons are collapsed according to their types, yielding 3 vertices and our procedure in this case suggests $\hat{d} = 2$. This is consistent with the random vertex contraction results' suggestion that the chemical pathways and electric pathways are sufficiently different with respect to their connectivity structures.

On the other hand, in Listing 1.9, 279 neurons are aggregated/collapsed according to their types, yielding 52 vertices, and our procedure in this case suggests $\hat{d} = 1$. However, our "more-intense" vertex contraction suggests that the "hypothesis" that $\hat{d} = 1$ can not hold true. Note that this does not says that the information criteria mistakenly has chosen $\hat{d} = 1$. Rather, the choice made by `getAICc` says that given the amount of data that we have, the most frugal way to represent the data is to use a single mean in the spirit of the principle known in a machine learning community as the *bias-variance trade-off*. Roughly speaking, in our particular case, for $n = 279$, it amounts to a saying that even if $\boldsymbol{\mu}_e$ and $\boldsymbol{\mu}_a$ are different, with the amount of data given, it is not frugal to model both with the same mean matrix. On the other hand, upon vertex contraction to a much smaller matrices, i.e., each entry in the matrices has a bigger number, such is no longer a hindrance to making a precise decision.

Listing 1.9 Vertex Contraction to 52 vertices

```
> n.types = data.Celegans$Types
> n.types.unique = unique(n.types)
> Q = matrix(0,52,278)
> for(itr in 1:3) {
>     Q[itr,which(n.types==n.types.unique[itr])] = 1
> }
```

```
> Ae.Q = Q %*% Ae %*% t(Q)
> Ac.Q = Q %*% Ac %*% t(Q)
> AeAc.Q = sapply(list(Ae.Q,Ac.Q),as.vector)
> AeAc.Q.gic = foreach(itr=1:2,.combine='rbind') %do% {
>     getAICc(gclust.rsvt(AeAc.Q,itr,method='lee'))
> }
```

Listing 1.10 Vertex Contraction to 3 vertices

```
> n.Vcols = data.Celegans$Vcols
> n.Vcols.unique = unique(n.Vcols)
> Q = matrix(0,3,278)
> for(itr in 1:3) {
>     Q[itr,which(n.Vcols==n.Vcols.unique[itr])] = 1
> }
> Ae.Q = Q %*% Ae %*% t(Q)
> Ac.Q = Q %*% Ac %*% t(Q)
> AeAc.Q = sapply(list(Ae.Q,Ac.Q),as.vector)
> AeAc.Q.gic = foreach(itr=1:2,.combine='rbind') %do% {
>     getAICc(gclust.rsvt(AeAc.Q,itr,method='lee'))
> }
```

1.3.4 Example 4 – Simulation experiment motivated by real data

We now consider a biologically motivated simulation example, with model parameters extracted from Izhikevich & Edelman (2008). This example is significantly simplified from the (necessarily incompletely understood) biology; nonetheless, it is (loosely) based on biological understanding and serves as a challenging illustrative test case.

We consider a sequence of (random) graphs from a stochastic block model with two motifs specified by $\overline{B}^{(1)}$ and $\overline{B}^{(2)}$, where

$$\overline{B}^{(1)} := \begin{pmatrix} 0.1 & 0.045 & 0.015 & 0.19 & 0.001 \\ 0.045 & 0.05 & 0.035 & 0.14 & 0.03 \\ 0.015 & 0.035 & 0.08 & 0.105 & 0.04 \\ 0.19 & 0.14 & 0.105 & 0.29 & 0.13 \\ 0.001 & 0.03 & 0.04 & 0.13 & 0.09 \end{pmatrix}, \tag{1.7}$$

$$\overline{B}^{(2)} := \begin{pmatrix} 0.19 & 0.14 & 0.29 & 0.105 & 0.13 \\ 0.001 & 0.03 & 0.13 & 0.04 & 0.09 \\ 0.015 & 0.035 & 0.105 & 0.080 & 0.04 \\ 0.045 & 0.05 & 0.14 & 0.035 & 0.03 \\ 0.1 & 0.045 & 0.19 & 0.015 & 0.001 \end{pmatrix}. \tag{1.8}$$

Note in particular that $\overline{B}^{(2)}$ is obtained by $\overline{B}^{(1)}$ by permuting the rows of $\overline{B}^{(1)}$ and then permuting the columns of $\overline{B}^{(1)}$.

Listing 1.11 Parameter Set-Up

```
> B1 <- rbind(
>   c(.1,    .045, .015, .19,   .001),
>   c(.045, .05,   .035, .14,   .03),
>   c(.015, .035, .08,   .105, .04),
>   c(.19,   .14,   .105, .29,   .13),
>   c(.001, .03,   .04,   .13,   .09))
> P1<- t(matrix(c(
>   0,0,0,1,0,
>   0,0,0,0,1,
>   0,0,1,0,0,
>   0,1,0,0,0,
>   1,0,0,0,0),5,5))
> P2<- t(matrix(c(
>   1,0,0,0,0,
>   0,1,0,0,0,
>   0,0,0,1,0,
>   0,0,1,0,0,
>   0,0,0,0,1),5,5))
> B2 <- P1 %*% B1 %*% P2
```

Then, for each $t = 1,\ldots,10$, we take $G(t) \sim SBM(B^{(\kappa(t))}, \nu)$ where for each ij, if $\kappa(t) = r$, then $G_{ij}(t)$ is a Poisson random variable with its success probability $B^{(r)}_{\nu(i),\nu(j)}$, where $\nu : \{1,\ldots,100\} \to \{1,\ldots,5\}$ and the cardinality of $\{k : \nu(t) = k\}$ is 20 for each $k = 1,\ldots,5$.

In short, we have $T = 10$ graphs, each of them has $n = 100$ vertices and follows a 5-block model. There are $m = 20$ vertices in each block. We are using $\overline{B}^{(1)}$ to generate the first 5 graphs while we are using $\overline{B}^{(2)}$ to generate the rest 5 graphs. Moreover, note that the sequence $\{N(t)\}_{t=1}^{T}$ form an i.i.d. sequence of random variables. where $N(t) := \mathbf{1}^{\top} G(t)\mathbf{1}$. In other words, there are no class-differentiating signal in the edge weights. As such, the only class-differentiating signal present is from the structural difference between $\overline{B}^{(1)}$ and $\overline{B}^{(2)}$.

Listing 1.12 Data Generation

```
> W = cbind(as.vector(B1),as.vector(B2))
> H = rbind(
>         rep((c(0,1)),times=c(5,5)),
>         rep((c(1,0)),times=c(5,5)))
> X0 = apply(W %*% H,2,function(x) Matrix(x,5,5))
> Mvec = lapply(X0,
>         function(x) {
>           retval = apply(x,2,rep,times=rep(20,5))
```

```
>           retval = apply(retval,1,rep,times=rep(20,5))
>         })
> Mvec = sapply(Mvec, as.vector)
> Xvec = apply(Mvec, c(1,2), function(x) rpois(1,x))
```

In Table 1.6. we report the AICc values for performing clustering on a sequence of graphs whose expected values are

$$(\overline{B}^{(1)},\ldots,\overline{B}^{(1)},\overline{B}^{(2)},\ldots,\overline{B}^{(2)}).$$

From the result, we can see that with repeated SVT steps (i.e., `gclust.rsvt`), we obtain the correct number of clusters, namely, 2. But without any SVT step (i.e., `gclust.app`), the algorithm's choice rank 6 is far away from the true rank. So we can conclude that the SVT step improves the model selection performance by getAICc.

Table 1.6 AIC for unperturbed $\overline{B}^{(k)}$

	Rank 1	Rank 2	Rank 3	Rank 4	Rank 5	Rank 6
`gclust.rsvt`	169.34	**167.16**	171.72	176.77	184.00	191.99
`gclust.app`	169.36	163.25	161.23	158.60	156.59	**155.21**

By perturbing the $\overline{B}^{(k)}$ to become sparser or denser. we compare the performance of the algorithm with SVT and without SVT. For each ε, we write

$$B^{(k)}(\varepsilon) = (\varepsilon\overline{B}^{(k)}) \wedge \mathbf{1}\mathbf{1}^{\top},$$

where $\wedge$ denote the operation that takes component-wise minimum. Note that it remains that $B^{(2)}(\varepsilon)$ can be derived from $B^{(1)}(\varepsilon)$ by permutations. We use the parameter ε to control the sparsity of the matrix while keeping the overall structure of B . When ε approaches 0, $B^{(1)}(\varepsilon)$ is close to a zero matrix; when ε is large enough, $B^{(1)}(\varepsilon)$ approaches J. In both cases, the difference between $B^{(1)}(\varepsilon)$ and $B^{(2)}(\varepsilon)$ are decreasing, which means the matrix is transforming from rank-2 to rank-1.

From Figure 1.3, we can see that no matter the matrix becomes sparser or denser, the algorithm with SVT always gives us the true rank except for the extreme case. The algorithm with SVT successfully captures this transformation. As to the algorithm without SVT, we can see that it performs as good as the one with SVT. But when the matrix becomes sparser, it performs consistently bad and do not capture the transformation of the rank.

For another perturbation scheme using

$$B^{(k)}(\varepsilon) = (\overline{B}^{(k)} + \varepsilon\mathbf{1}\mathbf{1}^{\top}) \wedge \mathbf{1}\mathbf{1}^{\top},$$

one can also see, from Figure 1.4, that a similar pattern persists.

We will consider the following clustering procedure as the baseline: Form the matrix Λ with $\Lambda_{ij} = \|G(i) - G(j)\|_F$. Use function `pamk` from the `fpc` package on

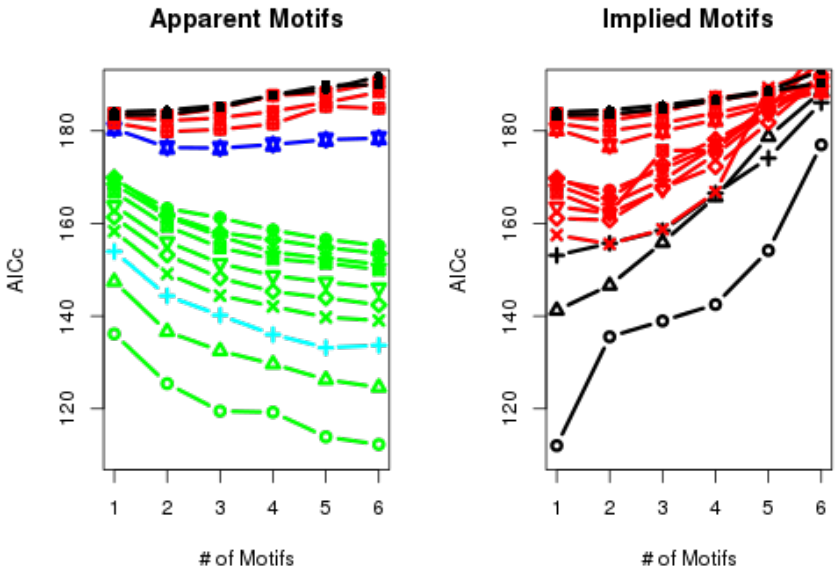

Fig. 1.3 AIC for $(\varepsilon B) \wedge \left(\mathbf{1}\mathbf{1}^{\top}\right)$ as ε changes

Λ and allow it to consider between 2 and $\min(\texttt{maxRank}, m-1)$ clusters. With our SBM parameters $B^{\kappa(t)}$ specified in 1.7 and 1.8, we generate $T = 20$ graphs, with $\kappa(t) = 1$ for $t = 1,2,\ldots,9$, and $\kappa(t) = 2$ for $t = 10,11,12,\ldots,20$. That is, the first nine graphs are from cluster 1 and the rest are from cluster 2. We let $m = 10$, so that $n = 50$.

Listing 1.13 Clustering Method #1 – `pamk` o `dist`

```
> mcout.pamk
>   = foreach(mcitr=1:100,.combine='rbind') %dopar% {
>       Xvec.r = Xvec100.r.tmp[[mcitr]]
>       diss = dist(t(Xvec.r))
>       sim.cl.pamk = pamk(diss,krange=1:9,diss=TRUE)
>       rbind(pamk=sim.cl.pamk$pamobject$clustering)}
> mcout.pamk.meanari = mean(apply(mcout.pamk,1,
>       function(x) adjustedRandIndex(x,c(rep(1,5),rep(2,5)))))
```

For one realization of the data, we see that gclust outperforms the baseline, as gclust correctly clusters all graphs and the baseline clusters one graph incorrectly. Both methods correctly identified 2 clusters.

After repeating the above experiment 1000 times, we obtain a range of ARI values for each method. We've summarized them using a boxplot as shown below. Basically, gclust never has a sub-perfect ARI, while the baseline method sometimes

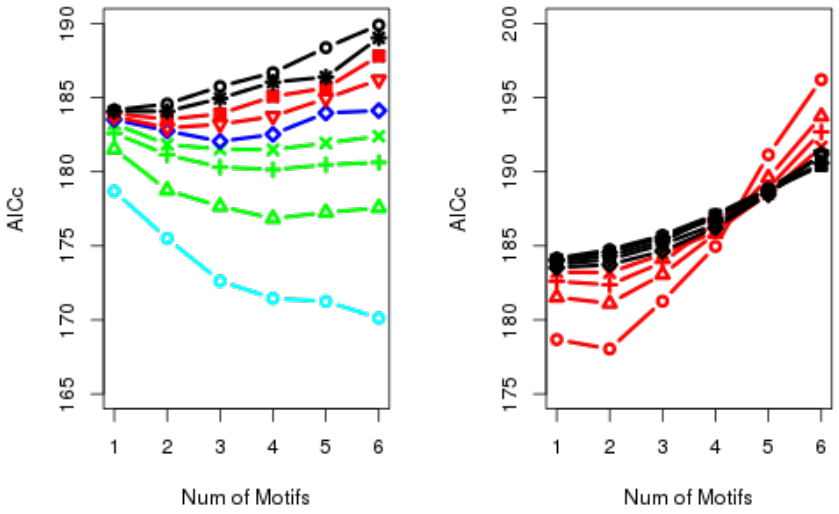

Fig. 1.4 AIC for $(\varepsilon B + \varepsilon \mathbf{1}\mathbf{1}^\top) \wedge \left(\mathbf{1}\mathbf{1}^\top\right)$ as ε changes

Listing 1.14 Clustering Method #2 (`gclust`)

```
> mcout.gclust =
>   foreach(mcitr=1:100,
>   .combine='rbind',
>   .errorhandling='remove',
>   .inorder=FALSE) %dopar% {
>       Xvec.r = Matrix(Xvec100.r.tmp[[mcitr]])
>       aic.rsvt =
>           foreach(x=1:3,
>       .combine='rbind',
>       .errorhandling='remove',
>       .inorder=FALSE) %dopar% {
>               getAICc.dev(gclust.rsvt(Xvec.r, x, method='lee'))}
>       opt.mod = which.min(aic.rsvt[,4])
>       gout = gclust.rsvt(Xvec.r,aic.rsvt[opt.mod,1])
>       rbind(gclust = apply(gout$H,2,which.max))
> }
> mcout.gclust.meanari = mean(apply(mcout.gclust,1,
>               function(x) adjustedRandIndex(x,c(rep(1,5),rep(2,5)))))
```

Table 1.7 Confusion matrices for `gclust` and `baseline`

	True Cluster 1	True Cluster 2
`gclust` Cluster 1	9	0
`gclust` Cluster 2	0	11

	True Cluster 1	True Cluster 2
`pamk` Cluster 1	9	1
`pamk` Cluster 2	0	10

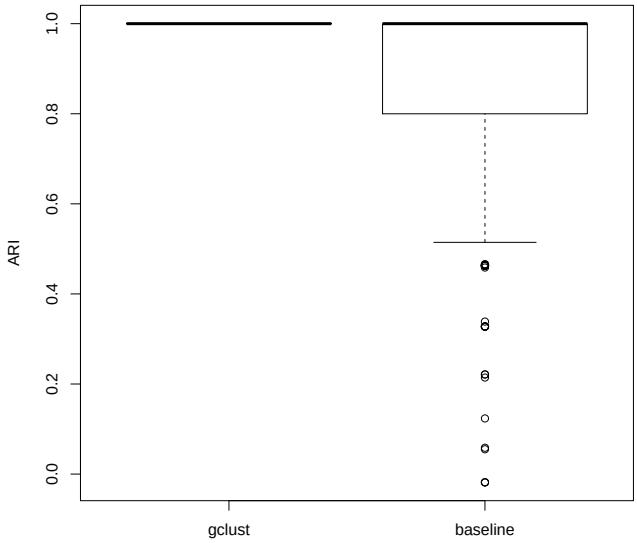

Fig. 1.5 Performance Comparison Between Two Clustering Approaches

does. After repeating the above experiment 1000 times, we obtain a range of ARI values for each method.

1.3.5 Example 6 – Simulated experiment with an configuration antagonistic to "lee" and "brunet" options.

In this example, we illustrate a typical situation under which (getAICc o gclust) does not perform well, and provide a heuristic that sometime remedy the situation.

Listing 1.15 Building a data matrix with an antagonistic configuration

```
> nvertex = 10
> maxT = 10
> distr <- c(rep(10,maxT/2),rep(10,maxT/2))
> A = foreach(itr=1:maxT,.combine='cbind') %do% {
>     rpois(nvertex^2,lambda=distr[itr])
> }
> gic = foreach(itr=1:10,.combine='rbind') %do% {
>     getAICc(gclust.rsvt(A,itr,method='lee'))
> }
```

Listing 1.16 Using `pe-nmf` for sparsity

```
> betagrid = seq(0,1,by=0.25)
> gic = foreach(beta=betagrid) %do% {
>           foreach(itr=1:10),.combine='rbind') %do% {
>               retval = gclust.rsvt(Alist, itr,
>               nmfmethod='pe-nmf',
>               alpha=0, beta=theta);
>               retval = getAICc(retval) }}
> sapply(gic,function(x) which.min(x[,4]))
```

The reason that Listing 1.15 pose a significant challenge to (getAICc o gclust) is that although $G(t)/s(G(t))$ yields a consistent estimate of its expected value $\mathbf{E}[G(t)|s(G(t))]/s(G(t))$, the first half has a significantly larger variance than the last half.

We now return to our first example using Wikipedia data.

Table 1.8 Cluster assignment based on taking $\widehat{d} = 6$ for clustering 12 Wikigraphs shows the perfect agreement – based on 6×6 graphs

–	People	Places	Dates	Things	Math	Categories
English Wikipages	$*$	$+$	$\times$	$\bullet$	$\diamond$	$\circ$
French Wikipages	$*$	$+$	$\times$	$\bullet$	$\diamond$	$\circ$

Table 1.9 Are French and English Wikigraphs similar? The answer using `gclust.app` is *No*.

nclust	negloglik	penalty	AIC
1	43.11	1.525	44.63
2	41.65	4	45.65

Listing 1.17 Vertex Contraction on Wikipedia data

```
> vcpol = c(rep(33,10),34,
>         rep(25,10),11,rep(18,10), 11,
>         rep(19,10), 12,
>         rep(19,10),19,
>         rep(9,10),6)
>
> vcmat = t(apply(diag(length(mybrks)),2,rep,times=vcpol));
> rWikiFr = vcmat %*% WikiFr %*% t(vcmat)
> rWikiEn = vcmat %*% WikiEn %*% t(vcmat)
>
> tWikiEn= vector('list',6)
> indx = 1
> for(itr in seq(1,66,by=11)) {
>     LHS = itr;RHS = itr + 10;
>     tWikiEn[[indx]] = rWikiEn[LHS:RHS,LHS:RHS]
>     indx = indx + 1
> }
>
> tWikiFr = vector('list',6)
> indx = 1
> for(itr in seq(1,66,by=11)) {
>     LHS = itr;RHS = itr + 10;
>     tWikiFr[[indx]] = tWikiFr[LHS:RHS,LHS:RHS]
>     indx = indx + 1
> }
> tWiki = c(tWikiEn,tWikiFr)
>
> aic.rsvt = t(sapply(1:12,
>     function(x) getAICc(gdclust.rsvt(tWikiFr,x))));
```

1.3.6 Shooting patterns of NBA players

To find out 9 NBA player's shooting patterns using statistical learning methods(especially Non-Negative Matrix Factorization) with R.

Paper "Factorized Point Process Intensities: A Spatial Analysis of Professional Basketball"(Harvard University) Data Explanation: Our simulation data is based on raw data about made and missed field goal attempt locations from roughly half of the games in the 2012-2013 NBA regular seasons. These data were collected by optical sensors as part of a program to introduce spatio-temporal information to basketball analytics. We discretize the basketball court into V tiles and compute X such that $X_{n,v} = \{x_{n,i} : x_{n,i} \in v\}$, the number of shots by player n in tile v. The following table (See Table 1.10) is the data used in this paper:

There are 9 players and 10 patterns in the data set. Each cell x_{ij} in this table represents the probability of *ith* player shooting in tile *j*.For instance, see row "Tony Parker" column "Tile 3", the number 0.17 means that there is a 0.17 possibility that Tony Parker shoots in Tile 3.

Table 1.10 Multinomial probability for 9 NBA players' shooting location

Player	Tile 1	Tile 2	Tile 3	Tile 4	Tile 5	Tile 6	Tile 7	Tile 8	Tile 9	Tile 10
LeBron James	0.21	0.16	0.12	0.09	0.04	0.07	0.00	0.07	0.08	0.17
Brook Lopez	0.06	0.27	0.43	0.09	0.01	0.03	0.08	0.03	0.00	0.01
Tyson Chandler	0.26	0.65	0.03	0.00	0.01	0.02	0.01	0.01	0.02	0.01
Marc Gasol	0.19	0.02	0.17	0.01	0.33	0.25	0.00	0.01	0.00	0.03
Tony Parker	0.12	0.22	0.17	0.07	0.21	0.07	0.08	0.06	0.00	0.00
Kyrie Irving	0.13	0.10	0.09	0.13	0.16	0.02	0.13	0.00	0.10	0.14
Stephen Curry	0.08	0.03	0.07	0.01	0.10	0.08	0.22	0.05	0.10	0.24
James Harden	0.34	0.00	0.11	0.00	0.03	0.02	0.13	0.00	0.11	0.26
Steve Novak	0.00	0.01	0.00	0.02	0.00	0.00	0.01	0.27	0.35	0.34

Procedures After introducing background knowledge, we discuss how we can use the techniques above to research NBA players' shooting patterns.

1.3.6.1 How to get "mydata"

Notation: Let x_{ij} represents the number of shoots for ith player in jth tile.
Assumptions: Assuming that the number of shoots follows a multinomial distribution.
We have a probability vector of each player and the total number of shoots each player tried, we can get the following data set (See Table 1.11) by parametric bootstrap method.

Table 1.11 Number of Shoots

Pattern	LeBron	Brook	Tyson	Marc	Tony	Kyrie	Stephen	James	Steve
1	147	27	165	138	55	108	52	306	0
2	10	146	373	19	89	79	15	0	8
3	92	221	16	109	69	69	31	110	0
4	51	47	0	4	30	104	14	0	17
5	22	3	4	268	76	134	71	24	0
6	62	9	16	198	27	21	38	25	0
7	0	32	5	0	33	98	130	136	6
8	46	12	0	10	21	0	16	0	200
9	45	0	13	0	0	69	60	100	232
10	125	3	8	24	0	118	121	199	217

References

1. Renaud Gaujoux and Cathal Seoighe. A flexible r package for nonnegative matrix factorization. *BMC Bioinformatics*, 11(1):367, 2010.

2. Daniel D. Lee and H. Sebastian Seung. Algorithms for non-negative matrix factorization. *Advances in neural information processing systems*, 2000.
3. P. Paatero and U. Tapper. Positive matrix factorization: a non-negative factor model with optimal utilization of error estimates of data values. *Environmetrics*, 1994.