SUPERVISED NON-NEGATIVE MATRIX FACTORIZATION FOR ANALYSIS OF MICROBIOME DATA

by

Yun Cai

Submitted in partial fulfillment of the
requirements for the degree of
Master of Science

at

Dalhousie University
Halifax, Nova Scotia
August 2014

Table of Contents

List of Tables

List of Figures

Abstract

Non-Negative Matrix Factorization (NMF) is a very useful tool to reduce dimension of data in machine learning and data mining. But it is an unsupervised learning method. To extract more discriminant information in the training data and improve the performance of classification, we developed a new supervised NMF method. We combine feature matrices from different classes as the feature matrix for the whole data and fit a non-negative Poisson regression to calculate the weight matrix. Our method is tested on the animal dataset and moving picture dataset. The experimental results show that our supervised NMF could greatly enhance the performance of NMF for classification.

List of Abbreviations and Symbols Used

Symbols and Abbr.	Description
X	data matrix
W	Feature matrix
H	Weight matrix
OTU	Operational Taxonomic Unit
NMF	Non-negative matrix factorization
BIC	Bayesian information criterion

Acknowledgements

I would like to express my gratitude to all those who helped me during the writing of this thesis. I gratefully acknowledge the help of my supervisors, Dr. Hong Gu and Dr. Toby Kenney, who have offered me valuable suggestions in the academic studies. In the preparation of the thesis, they spent much time to encourage me and guide me even though they are very busy. They also read through each draft of the thesis and provided me with inspiring advice. Without their patient instruction, insightful criticism and expert guidance, the completion of this thesis would not have been possible.

I also owe a special debt of gratitude to all the professors in this department. From their devoted instructions and enlightening lectures I have benefited greatly and get academically prepared for the thesis.

I would finally like to express my gratitude to my beloved parents and friends who have always been helping me out of difficulties and supporting me no matter when and where.

Chapter 1

Introduction

1.1 Background

Microbes are everywhere, and we are becoming more aware of the many different roles they play. However, we still have limited understanding of the functioning of microbial communities. We are now aware that microbes do not function in isolation, and the community structure of the microbes is the key to understanding the effects of various microbes.

It seems that in many cases, differences between different types of microbial communities (for example, the communities in the guts of healthy and sick people) are not attributable to a single microbe, but rather to the overall structure of the community. It is therefore critical to devise models which take into account this overall structure. The goal of this thesis is to present some first steps towards this type of model.

Usually, the microbes data are counts of different Operational Taxonomic Units (OTUs), species distinction in microbiology (Wooley 2012) [15]. Considering the difficulty and examples of collecting data and the huge number of related OTUs, the data always consists of hundreds or even thousands of variables but only a few observations, which means $p \gg n$ (p is the number of variables and n is the number of observations). We can not apply classical models to this kind of data because of high variance and overfitting. This thesis explores methods to reduce the dimension of data without losing too much important information.

To deal with this kind of problem, unsupervised learning methods like principal components analysis (PCA) and vector quantization (VQ) are now usually used to reduce dimension and pick up main features of the data. But the results from PCA or VQ usually contain negative numbers which are hard to interpret naturally. So in the microbial communities' case, we prefer nonnegative matrix factorization (NMF).

However, like other dimension reduction methods, NMF extracts the most significant features from the data but these sometimes are not what we are interested in.

So to get certain specific information, a supervised learning NMF should be used.

1.2 Review of NMF

1.2.1 NMF

Non-negative matrix factorization (Lee and Seung 1999) [6] is a dimension reduction method for non-negative data (usually count data). The idea is to represent each data point as a mixture of features. Given a non-negative $m \times n$ matrix X, we approximate

$$X \approx W \times H$$

where W is a non-negative $m \times k$ matrix and H is a non-negative $k \times n$ matrix. This means that each column of X is a non-negative linear combination of the features (columns of W). In our analysis, X is the microbes data with counts of OTUs or genes. X_{ij} means of the jth observation, the ith type of OTU or gene appear X_{ij} times. Here k is the number of metavariables, also called the number of types in this thesis. k determines the complexity of the model, thus it is a tuning parameter in this context. Usually k is chosen such that $(m+n)k < nm$, so that we reduce the dimension significantly (Lee and Seung 1999) [6].

For the approximation, Non-negative matrix factorization (NMF) is usually performed to maximize the Poisson log-likelihood of the data,

$$L(W,H) = \sum_{i,j} \left(X_{ij} \log(WH)_{ij} - (WH)_{ij} \right),$$

where $(WH)_{ij}$ is the Poisson mean for X_{ij} (Hastie, Tibshirani and Friedman 2009) [25]. But Euclidean distance can also be used as a criterion in some cases(Lee and Seung 2001) [7].

Figure 1.1 shows an example (Lee and Seung 1999) [6] in which NMF is used in image data analysis, and is compared with the application of other methods such as principal components analysis (PCA) and vector quantization (VQ) in the same data. The database contains 2429 facial images, each face consisting of 19×19 pixels, which means the data is a 361×2429 matrix. So here for the $m \times n$ matrix X, $m = 361, n = 2429$. Each method applies an approximate factorization of the form $X \approx W \times H$. As shown in the 7×7 montages, all three methods have learned a

set of $r = 49$ basis images. Black pixels correspond to positive values and red pixels correspond to negative values. A particular instance of a face shown at the top right is approximately represented by a linear superposition of basis images. The coefficients of the linear superposition is a 7×7 grid shown next to the montages. On the right side are the resulting superpositions.

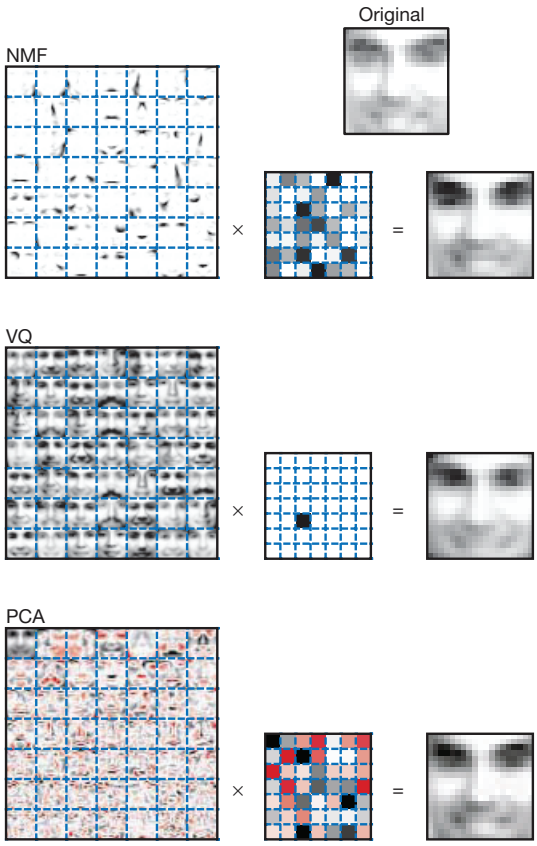


Figure 1.1: Facial image

Lee and Seung (1999) [6] pointed out that NMF has the advantage of interpretability, because it tends to identify sparse features, and describe each data point as a mixture of a small subset of the relevant features. Also, unlike PCA and VQ, it takes account of the non-negative nature of the data.

Then various authors have extended the NMF problem to include additional constraints. For example, smoothness constraints have been used to regularize the computation of spectral features in remote sensing data (Piper 2004) [13] and sparsity

constraints are useful when the degree of sparseness in the H or W matrix needs to be controlled.(Hyunsoo Kim and Haesum Park 2007) [12].

1.2.2 Algorithm for NMF

Lee and Seung (2001) [7] gave two different multiplicative algorithms to compute NMF. One is to minimize the conventional least squares error and the other one is to minimize the generalized Kullback-Leibler divergence. They also proved the monotone convergence of both algorithms using an auxiliary function similar to that used to prove the convergence of the Expectation-Maximization algorithm. This guaranteed that a locally optimal solution for NMF could be found. The second multiplicative algorithm is supplied below.

$$H_{kl} \leftarrow H_{kl} \sum_i W_{ik} \frac{X_{il}}{(WH)_{il}}$$

$$W_{ik} \leftarrow W_{ik} \sum_l \frac{X_{il}}{(WH)_{il}} H_{kl}$$

$$W_{ik} \leftarrow \frac{W_{ik}}{\sum_j W_{jk}}$$

The iteration of the update rule starts with non-negative values of H and W. Those update rules constrain the W and H matrices to be non-negative and also the column-sums of W to be 1. This sum constraint reduces the degeneracy associated with the invariance of WH under the transformation $W \leftarrow W\Lambda$,$H \leftarrow \Lambda^{-1}H$, where Λ is a diagonal matrix.

This multiplicative algorithm, as the first iterative algorithm for NMF, is the most well-known update algorithm and is usually used as a baseline to compare with other new algorithms. However, in practice, its low speed to converge has been shown several times. Also, it needs more iterations and more computation in each iteration than other popular algorithms like gradient descent and alternating least squares algorithms.

To speed up the original multiplicative algorithm, many authors have modified the algorithm. For example, Gonzalez and Zhang (2005) [8] found a way to accelerate this algorithm but they had a problem with convergence. Lin (2005) [4] solved the

convergence problem to guarantee the convergence of iteration, but his method needed more work per iteration than the original one which made his algorithm even slower.

Several alternative strategies have been presented later such as the gradient descent algorithms (Chu 2004) [18]. Actually, the multiplicative update algorithm is a kind of gradient descent method too. The basic gradient descent algorithm for NMF is:

$$H = H - \epsilon_H \frac{\partial f}{\partial H}$$

$$W = W - \epsilon_W \frac{\partial f}{\partial W}.$$

where f is the objective function. In our case, it is the log-likelihood function. ϵ_H and ϵ_W are step size parameters. This algorithm always depends on the step size in the negative gradient direction. The problem is to choose the values for ϵ_H and ϵ_W. Hoyer (2004) [21] proposed choosing 1 as both of their starting values and multiplying them by 1/2 at each iteration. This is simple but does not maintain the non-negative restriction. The W and H matrices may become negative during the iteration.

Then Shahnaz (2006) [9] suggested to set the negative elements to 0 after each iteration. This is commonly used to prevent the W and H matrices' elements becoming negative. But this projection method usually made the analysis more difficult, and sensitive to the starting values of W and H. Without a careful choice of initial values of W and H, it often leads to a poor factorization. Lee and Seung (2001) [7] proposed a better choice for the stepsize but made the algorithm really slow.

The third most popular algorithm is the alternating least squares. This algorithm followed a least squares step. It was first used by Paatero and Tapper (1994) [22]. This algorithm confirms that if one matrix in W and H is given, the other matrix could be found by using a least squares computation. The algorithm is as follows.

$$W = \underset{W \geq 0}{argmin}\, f(W, H),$$

$$H = \underset{H \geq 0}{argmin}\, f(W, H)$$

Here f is the objective function which in our case is the log-likelihood function.

This approach is the "block coordinate descent" method in bound-constrained optimization (Bertsekas 1999) [5]. This algorithm minimizes one block of variables under constraints while fixing the other blocks. NMF is the simplest case with only two block variables which are W and H. Also when we set all negative elements from the iteration to be 0, and remain 0 in the following iterations, we can highly speed up the algorithm. This also makes the algorithms more flexible, and keeps the process from a poor path.

Berry(2006) [20] summarised these three methods and compared them in his paper. No matter which algorithm we choose, one issue we must address is choosing the number of types, k.

1.2.3 Choosing the number of types

The choice of the number of types is an important problem which can significantly impact the results of the analysis. Among the methods available, three methods similar to Bayesian informatin criterion (BIC) methods (G Schwarz 1978) [11] are used most widely (Stoica 2004) [23]. BIC is based on the likelihood function and introduces a penalty term for the number of parameters in the model.

Let $W^{(k)}$ and $H^{(k)}$ be the result of the NMF when k is chosen to be the number of types. So W is $m \times k$ and H is $k \times n$.

$$\hat{X}^{(k)} = W^{(k)} \times H^{(k)}.$$

Under the assumption that the model errors or disturbances are independent and identically distributed according to a normal distribution and that the boundary condition that the derivative of the log-likelihood with respect to the true variance is zero, the BIC could be calculated as (Priestley 1981) [19]:

$$BIC = n \log(||\hat{X}^{(k)} - X||^2) + k \log(n).$$

Then the three similar methods are:

$$BIC_1(k) = \log(||\hat{X}^{(k)} - X||^2) + k\frac{m+n}{mn} \log(\frac{mn}{m+n}),$$

$$BIC_2(k) = \log(||\hat{X}^{(k)} - X||^2) + k\frac{m+n}{mn} \log(c^2),$$

$$BIC_3(k) = \log(||\hat{X}^{(k)} - X||^2) + k\frac{m+n}{mn}\frac{\log(c^2)}{c^2},$$

Here $c = c(m,n) = min(\sqrt{m}, \sqrt{n})$, and $||A|| = (tr(A'A))^{1/2} = ||A||_F$

To find the best k, compute the BIC for a range of values of k and the BIC value usually initially becomes smaller when k turns larger as the WH fits better for larger k. While after a critical value, the BIC become larger as the model becomes overfitting. The k value corresponding to the smallest BIC value is the optimal number of types.

The likelihood part is based on Normal distribution here. It is straight forward to replace Normal assumption by other distributions. We changed the assumption to Poisson distribution and applied the BIC method to our data, but the plot of BIC values does not show a pattern. This makes it difficult to choose a value for the number of types. So we develop a new method to choose the number of types in our thesis.

1.3 Summary of the thesis

In this thesis, we first apply NMF to microbiome data sets, and show that it does extract useful features in some cases (Chapter 2). However, in other cases, there is room for improvement. We therefore proceed to develop a supervised version of NMF for identifying distinguishing features (Chapter 3). In this chapter, we also develop non-parametric method for choosing the number of types. Finally in Chapter 4, we demonstrate the effectiveness of this method on both real and simulation data.

Chapter 2

Applying NMF to microbiome data

2.1 Choosing the number of types

Let k be the number of types. Here to avoid overfitting, we calculate the Poisson log-likelihood for each k value.

$$L(W, H) = \sum_{i=1}^{n} \sum_{j=1}^{m} \left(X_{ij} \log(WH)_{ij} - (WH)_{ij} \right),$$

W is an $n \times k$ matrix and H is a $k \times m$ matrix. As k become larger, the log-likelihood value will become larger. But after a certain value, the log-likelihood will increase more slowly which means the increase in the number of types will not bring more information to the analysis results. It is shown as a graph of log-likelihoods against k. So we choose the value where the curve does not increase significantly as the number of types increase.

The method involves the following steps:

1 Calculate the log-likelihood values for k types for a range of values of k.

2 Plot the log-likelihood values versus k.

3 Find the elbow point at which the curve becomes flat.

4 Set k equal to the value at that point.

2.2 Application of NMF

We apply standard NMF on three datasets: the animal dataset (Muegge 2011) [3], the IBD dataset (Qin 2010) [14]and the moving picture dataset (Caporaso 2011) [16]. We show the results in the form of weight matrix plots.

2.2.1 Results for animal data

The animals dataset (Muegge 2011) [3] contains gut metagenomes extracted from $n = 39$ animals. The metagenomes include 1239 different types of genes (categorized by EC number). The animals can be classified into four types: Carnivore, Foregut fermenting Herbivore, Hindgut fermenting Herbivore and Omnivore. There are 21 herbivores, 11 omnivores and 7 carnivores.

We choose the number of types for the animals data using the above method. We calculate log-likelihood for each k from 1 to 39 and the results are shown in Figure 2.1.

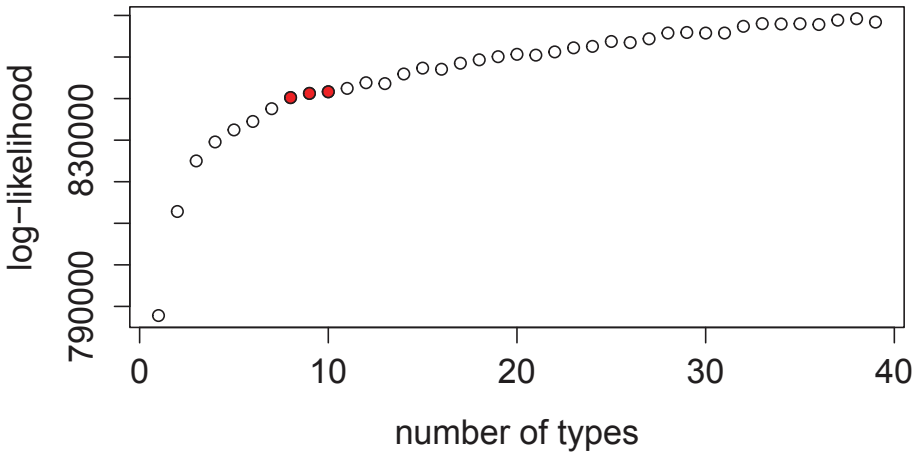


Figure 2.1: Animals data log-likelihood values

We see that here 8, 9 or 10 may be the appropriate value for k. So we perform NMF with types 8, 9 and 10. Then we plot each weight matrix H to see the classification result. We find that the points could be separated best when the number of types is 9.

To display this separation graphically, we project onto a 2-dimensional plane. We use an interactive software package (*SimplePlot*, which will soon be available from Toby Kenney's website `www.mathstat.dal.ca/~tkenney/`) to manually select the best projection by moving the features (represented by crosses on the figure) around

the plane until the data show the best separation. The advantage of using interactive software is that it is easier to identify non-linear separation if that is more appropriate for a particular dataset.

Figure 2.2 shows the weight matrix plot for 9 types.

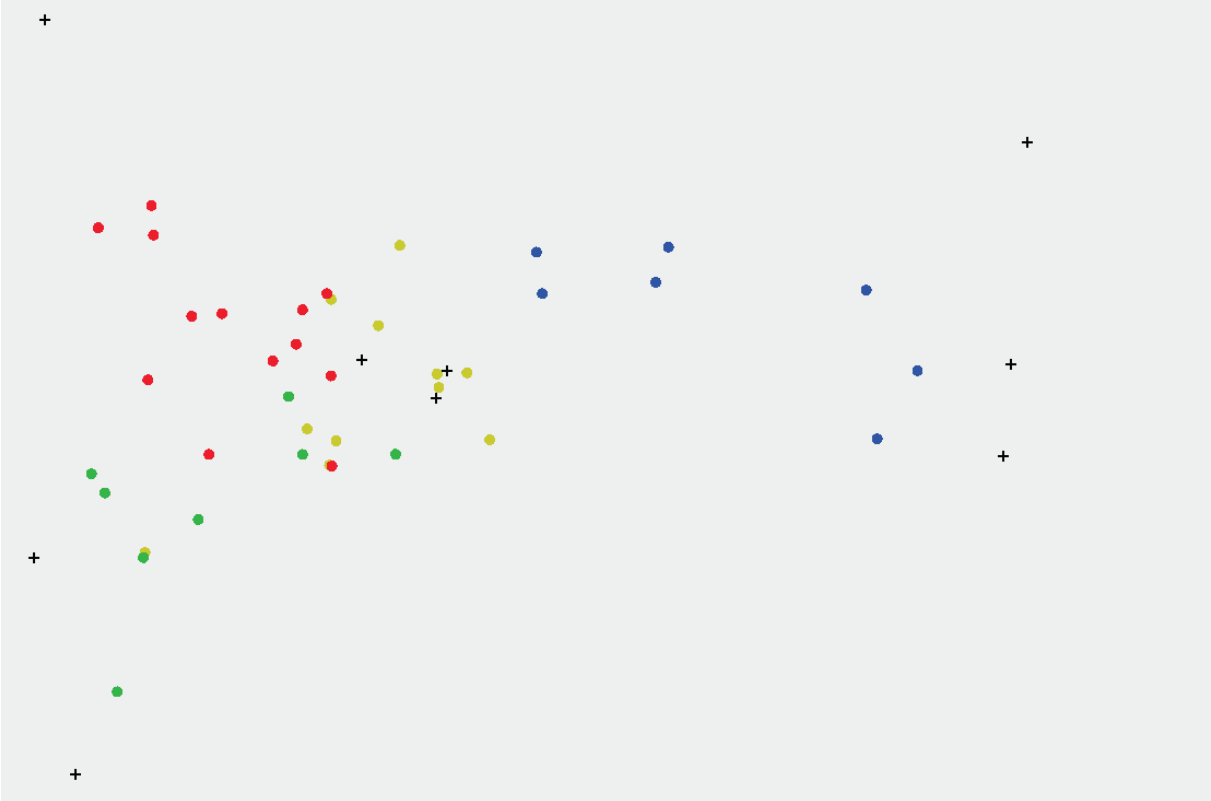


Figure 2.2: animal dataset classification for 9 types. Blue points are Carnivores, red points are Foregut fermenting Herbivores, green points are Hindgut fermenting Herbivores and yellow points are Omnivores.

Figure 2.2 show the Carnivores are totally separated from others and the other three types are separated with a few overlapping points. This means NMF reduces the high dimensional animal data to 9 features and still retain enough information to identify the different classes of animals.

2.2.2 Results for IBD data

In medicine, inflammatory bowel disease (IBD) is a group of inflammatory conditions of the colon and small intestine. It arises as a result of the interaction of environmental

and genetic factors. A number of studies have shown that alterations to enteral bacteria can contribute to inflammatory gut diseases (Coghlan 2012) [2].

We apply NMF to one dataset related to IBD (Qin 2010) [14]. The data consists of 2804 metagenomes sequenced from faecal samples from 124 individuals, 99 of whom are healthy, the others are suffering from IBD. They come from different countries and in different genders and ages. We want to see whether the structure of microbiome is associated with the disease state.

We choose the number of types for the IBD data using the above method. We try k from 1 to 20 and get Figure 2.3.

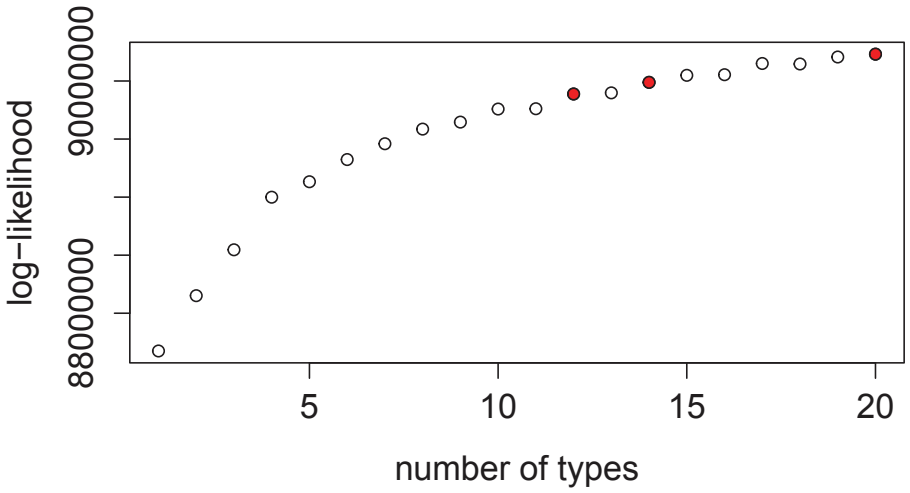


Figure 2.3: IBD data log-likelihood values

From the plot, we see that the curve keeps going up. 14 may be the number of types that works better than others, so we try 12, 14 and 20 to do the NMF and plot the weight matrices to see the results of the analysis. None of the three plots show a pattern of separation. The Figure 2.4 shows the result for 14 types as an example.

The two classes are totally mixed together which means NMF does not identify any distinguishing features here.

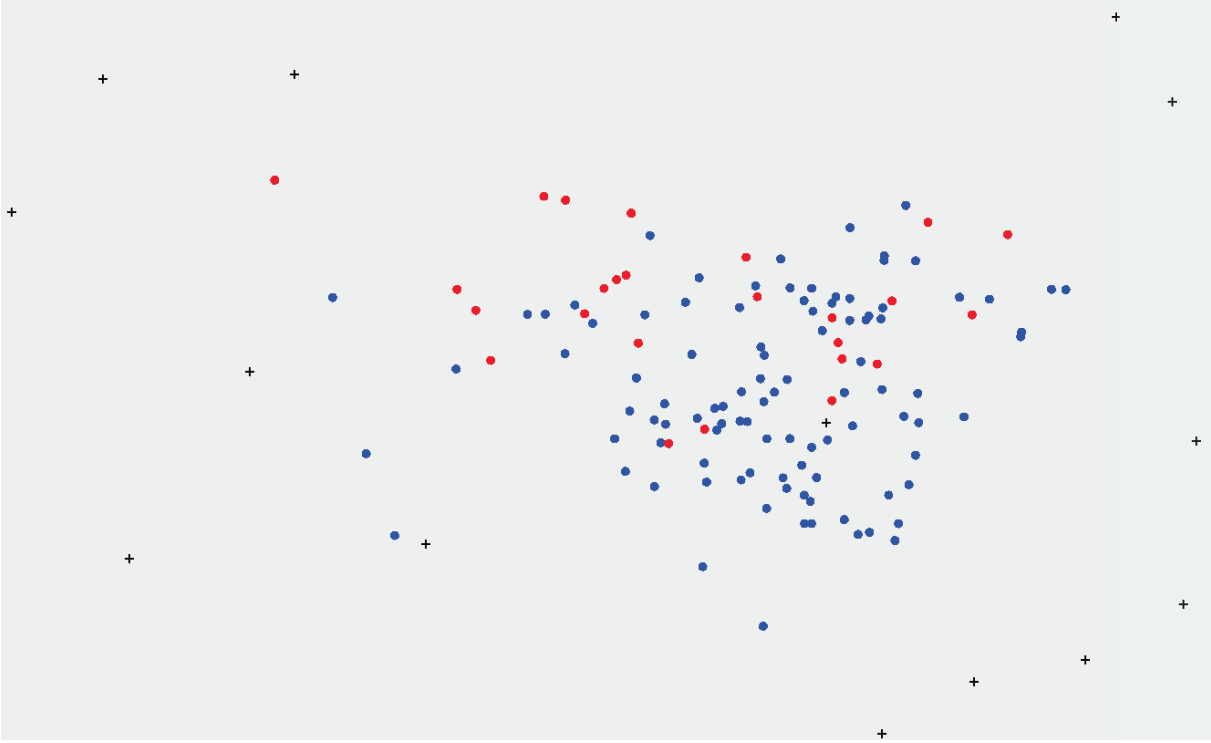


Figure 2.4: IBD dataset for 14 types. The red points here are individuals suffering from IBD and the blue points are healthy individuals.

2.2.3 Results for moving picture data

The moving picture data (Caporaso 2011) [16] is the most detailed investigation of temporal microbiome variation to date. It consists of a long-term sampling series from two human individuals at four body sites; gut, tongue, right and left palm, over 396 time points. Person 1 was measured at 135 time points and person 2 was measured at 261 time points. The total number of variables (OTUs) across all samples was 15685. In spite of this extensive sampling, no temporal core microbiome was detected, with only a small subset of OTUs reoccurring at the same body site across all time points (Caporaso 2011) [16].

We apply NMF to the guts data and tongues data from this moving picture dataset.

Results for tongues data

For the tongues data, there are 135 observations from person 1 and 373 observations from person 2. First we try type values ranging from 1 to 20 on the dataset to choose

an appropriate number of types.

Figure 2.5 shows the log-likelihood values for tongues data.

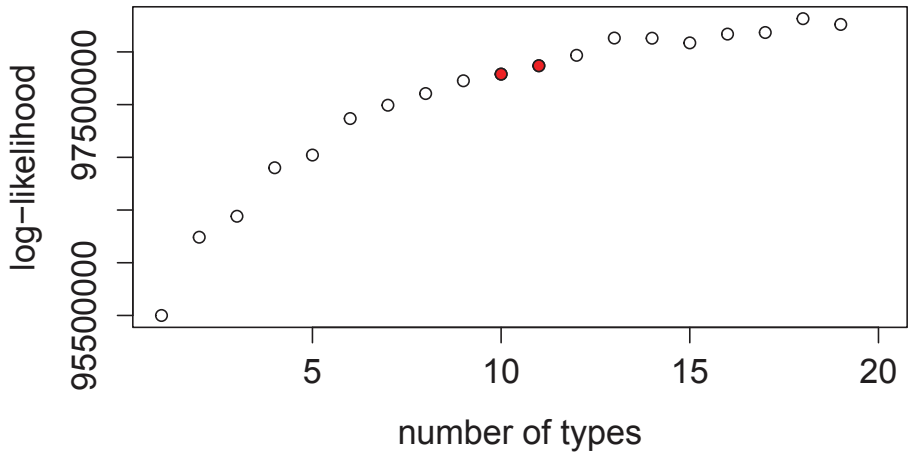


Figure 2.5: Tongues data log-likelihood values

From Figure 2.5, we think 10 or 11 may be appropriate for the tongue dataset. So we try NMF on 10 types and 11 types. Both results do not achieve good separation between samples from the two individuals. The classification results for 10 types are shown in Figure 2.6 as an example.

Here we see that we cannot separate the samples from the two individuals. That means the features given by NMF could not separate the two individuals well.

Results for guts data

The guts data consists of 131 observations from person 1 and 336 observations from person 2. We use the same methodology above to choose the number of types. Figure 2.7 shows log-likelihood values for guts data.

From the plot, we see that 6 or 7 could be chosen as the number of types for guts data. Then we perform a NMF with 6 types and 7 types on the guts data. They both give good separation so we choose the smaller number of types. The results for 6 types are shown below in Figure 2.8.

Figure 2.6: Tongues dataset for 10 types. The blue points are from individual 1 and the yellow points are from individual 2.

We see that the data could be well separated. That means 6 features could distinguish two individuals' guts.

2.3 Summary

Here we can see that although standard NMF works for animals dataset and guts dataset, but it does not perform well on IBD dataset and tongues dataset. The reason is that unsupervised NMF has a weakness that it could only preserve the structure in the data that is the most significant, while sometimes this is not what we want to get. In the next chapter, we modify NMF to identify the different features for different classes. This should allow us to more easily distinguish samples from different classes. In other words, we will develop a new supervised learning method based on NMF.

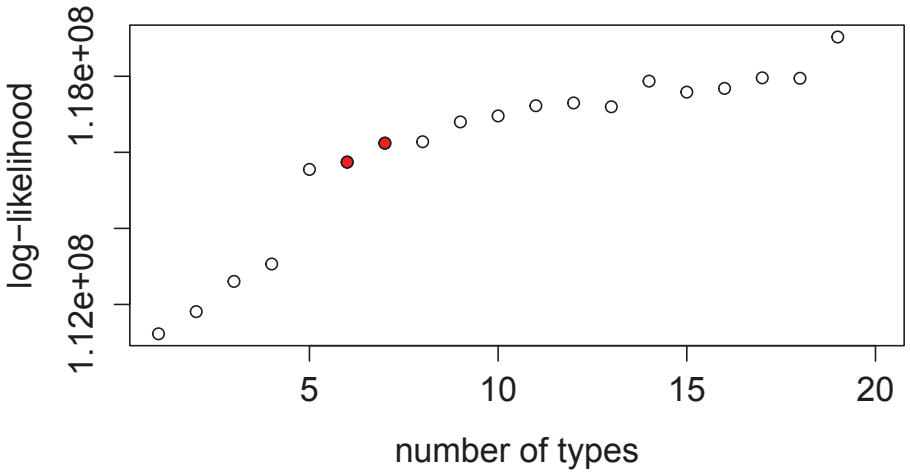


Figure 2.7: Guts data log-likelihood values

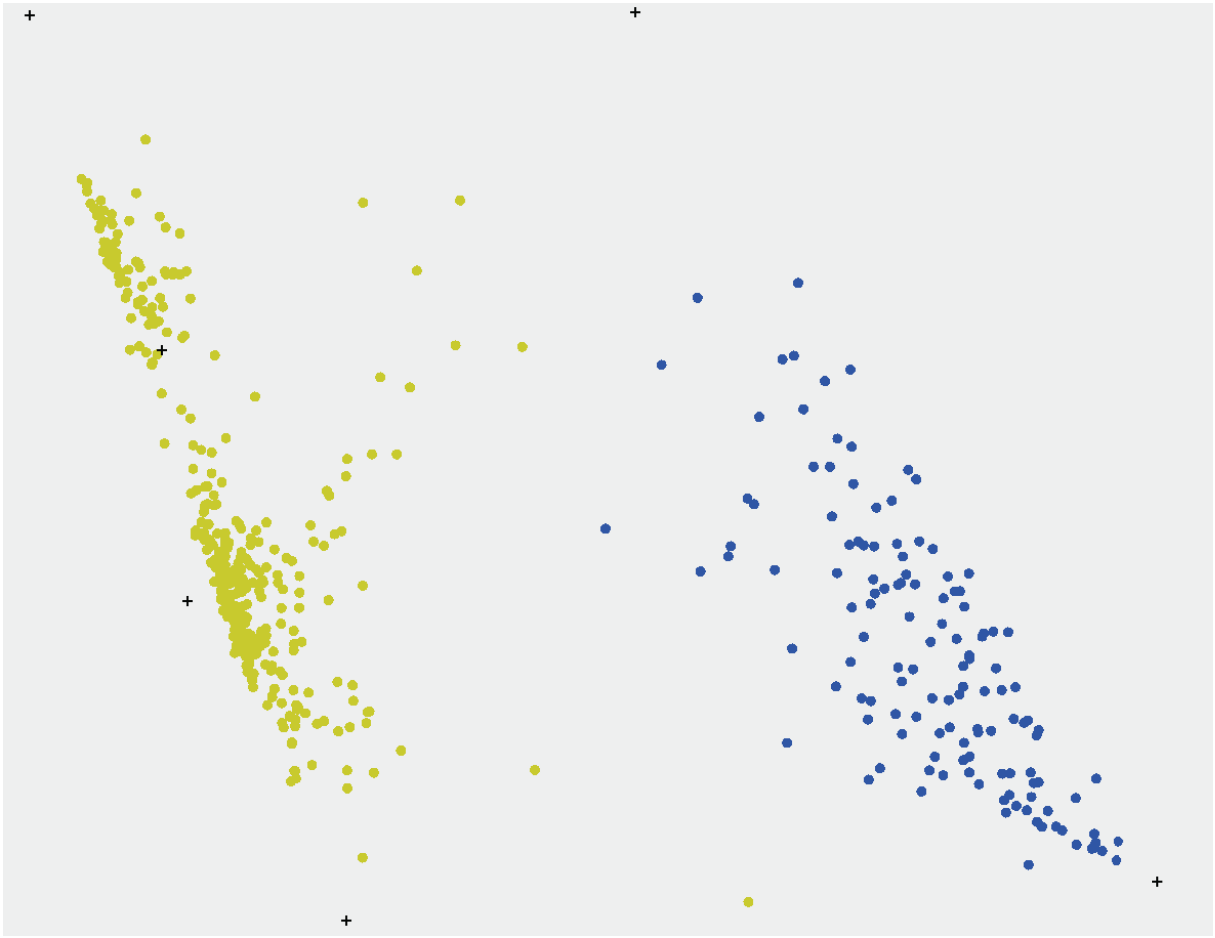


Figure 2.8: Guts dataset for 6 types. The blue points are from individual 1 and the yellow ones are from individual 2.

Chapter 3

Supervised NMF

3.1 Supervised NMF method

3.1.1 Review of Fisher NMF

In order to solve the above issue and improve the classification performance, more constraints should be added to NMF. A few authors have tried to add the Fisher constraint to the NMF to exploit more discrimination information for classification.

Wang (2004) [26] proposed a novel subspace method using Fisher linear discriminant analysis (Belhumeur 1997) [24]. Fisher defined the separation between these two distributions to be the ratio of the variance between the classes to the variance within the classes(Fisher 1936) [1]. The Fisher linear discriminant analysis is a method to maximize the ratio of the between-class scatter and the within-class scatter. Wang imposed Fisher constraints on NMF to get a new subspace method called FNMF (Fisher Non-negative Matrix Factorization). As we know in NMF, each column of weight matrix H is in one-to-one correspondence with a column of the original matrix X. The FNMF aims to maximize the between-class scatter (S_B) and minimize the within-class scatter (S_W) of H. The two scatters are defined as below:

$$S_W = \frac{1}{C}\sum_{i=1}^{C}\frac{1}{n_i}\sum_{j=1}^{n_i}(H_j - u_i)(H_j - u_i)^T$$

$$S_B = \frac{1}{C(C-1)}\sum_{i=1}^{C}\sum_{j=1}^{C}(u_i - u_j)(u_i - u_j)^T$$

Here n_i is the number of vectors in the ith class, C is the number of classes and u_i is the mean of classes i in H, $u_i = \frac{1}{n_i}\sum_{j=1}^{n_i} H_j$.

So the divergence function of X from its approximation WH is the objective function for FNMF:

$$D(X||WH) = \sum_{i,j}(X_{ij}log\frac{X_{ij}}{(WH)_{ij}} - X_{ij} + (WH)_{ij}) + \alpha S_W - \alpha S_B.$$

where α is a constant greater than 0.

To minimize the above function, Wang (2004) [26] gave the following update rules for which he proved convergence.

$$H_{kl} \leftarrow -b + \sqrt{b^2 + 4(\sum_{il} \frac{X_{ik}H'_{kl}}{\sum_k W_{ik}H'_{kl}})(\frac{2}{n_i C} - \frac{4}{n_i^2(C-1)})}$$

$$W_{kl} \leftarrow \frac{W_{kl}\sum_j X_{kj}\frac{H_{lj}}{\sum_k W_{kl}H_{lj}}}{\sum_j H_{lj}}$$

$$W_{kl} \leftarrow \frac{W_{kl}}{\sum_k W_{kl}}$$

where $b = \frac{4}{n_i C(C-1)}\sum_j(u_{kj} - (u_{ki} - \frac{H'_{kl}}{n_i})) - \frac{2}{n_i C}u_{ki} + 1$

Experiments based on the Cambridge ORL database (Wang 2004) [26] show that the FNMF works better then NMF for classification.

In the following years, many other supervised NMF methods were developed most of which were based on the FNMF.

One issue with FNMF is that it loses the interpretation of NMF. We develop a new approach to supervised NMF based on the idea that each class is based on its own features.

3.1.2 Developed supervised NMF

From the metadata we know the observations are from different classes. Our objective is usually to find the differences between the structures from the two classes. It is true that each class of data contain most of its own information and could be best described by its own feature matrix. So one way to distinguish features for different classes is to fit a feature matrix for each group separately and then combine them together. For example, if data X has t classes,

$$X = \left(\ X^{(1)}, X^{(2)}, \cdots, X^{(t)} \ \right)$$

$X^{(1)}, X^{(2)}, \cdots, X^{(t)}$ are t classes of observations.

$$X^{(1)} \approx W^{(1)} \times H^{(1)}$$

$$X^{(2)} \approx W^{(2)} \times H^{(2)}$$

$$\cdots$$

$$X^{(t)} \approx W^{(t)} \times H^{(t)}$$

Here $W^{(i)}$ and $H^{(i)}$ are the feature matrix and weight matrix for $X^{(i)}$. Both $W^{(i)}$ and $H^{(i)}$ are non-negative. To get the hidden structure of different classes in the whole data, we combine these feature matrices together and call this combined feature matrix for the whole data the W matrix.

$$W = (W^{(1)}, W^{(2)}, \cdots, W^{(t)})$$

It is straight forward that W is non-negative as all $W^{(i)}$ are non-negative. This W matrix contains all the features for each class. By maximizing Poisson log-likelihood

$$L(W, H) = \sum_{i,j} (X_{ij} \log(WH)_{ij} - (WH)_{ij},$$

we can get the weight matrix H for our fixed W matrix. An iterative algorithm for fitting H will be described in Section 3.2.

3.2 Algorithm for supervised NMF

As the W matrix is already non-negative, we need an algorithm which will keep the elements in H matrix non-negative and maximize the log-likelihood function subject to this constraint.

3.2.1 Newton's method

In optimization, Newton's method (Avriel 2003) [17] is applied to the derivative of a function to find its zeros. So here, it can be used to find the weight matrix H which maximizes the Poisson distribution's log-likelihood function. For each observation X_j, the log-likelihood function is:

$$L(H_j) = \sum_i (X_{ij} \ln(WH)_{ij} - (WH)_{ij})$$

The gradient of the log-likelihood function is:

$$\nabla L(H_j) = \sum_i (X_{ij} \frac{W_i}{(WH)_{ij}} - W_i)$$

The Hessian matrix for log-likelihood function is:

$$HessianL(H_j) = -\sum_i (X_{ij} \frac{{W_i}^T W_i}{(WH)_{ij}})$$

The iterative scheme is:

$$H_j = H_j - [HessianL(H_j)]^{-1} \nabla L(H_j)$$

So if the initial values for the iterative scheme are positive, the H will remain positive during the update. So the solution is a non-negative matrix H which maximizes the log-likelihood function.

However, when we apply this Newton's method to the real data, one issue is that the algorithm has a problem with convergence. It is a challenge to find an appropriate initial value for the iterative scheme to converge. Hence we develop a new algorithm to calculate the weight matrix.

3.2.2 Poisson regression coefficients selection method

After obtaining the features separately for each class, the combination of these features is used as the W matrix for the whole data. As we assume that the data are Poisson distributed and the W matrix is already fixed in the last step, finding the H matrix is a constrained Poisson regression problem on W. The idea is to maintain a list of the non-negative coefficients and fit a standard Poisson regression with identity link on the columns of W matrix corresponding to the coefficients in the list. We aim to find a list so that all coefficients are non-negative, and adding another feature to the list cannot improve the likelihood and still maintain the non-negative constraint.

So we developed the Poisson regression coefficients selection method. For each observation $X_{.j} = (X_{1j}, X_{2j}, \cdots, X_{mj})$, we keep fitting a Poisson regression on the

feature matrix W and remove the variables corresponding to the negative coefficients in coefficients vector $H_{.j} = h_{1j}, \cdots, h_{kj}$ until all the coefficients are non-negative. Using the remaining variables we calculate the log-likelihood value. Then we test a removed variable by adding it back and compare the new log-likelihood value to the old one. If the new value is greater than the old one, we add this variable back to the remaining variables and repeat the above steps, otherwise we remove this variable and test the next one.

For each observation X_j, the algorithm follows these steps:

1 Fit a Poisson regression with identity link but without intercept on W with the initial value of H_j set as the coefficients of linear least square regression of X_j on W. Eliminate those variables corresponding to negative coefficients.

2 If any variables were removed, go back to step 1 until all the coefficients are positive. In the end, the matrix consisting of remaining variables is W^+. Since X, W and H are all non-negative, the resulting W^+ cannot be empty.

3 Calculate the log-likelihood for W^+.

$$L(W^+) = \sum_{i=1}^{m} \left(X_{ij} \log(W^+ H_j)_i - (W^+ H_j)_i \right),$$

where $(W^+ H_j)_i$ denotes the ith element of the vector $W^+ H_j$

4 Add one variable in the removed pool to W^+, denote the new feature matrix as W^+_{new} and calculate the log-likelihood again.

$$L(W^+_{new}) = \sum_{i=1}^{m} \left(X_{ij} \log(W^+_{new} H_{j_{new}})_i - (W^+_{new} H_{j_{new}})_i \right),$$

where $H_{j_{new}} = (H_j(1-\varepsilon), \varepsilon)$, ε is a very small positive number close to 0. In our case, we use 10^{-7} as the value of ε.

5 Compare $L(W^+)$ with $L(W^+_{new})$, if $L(W^+) < L(W^+_{new})$, use this new W^+_{new} composed of W^+ and the new variable to repeat steps 1 to 5. Otherwise remove this variable and try to add another variable in the removed pool to W^+ and repeat step 4 to 5, until all removed variables have been tried.

In detail, the steps could be described as:

For each observation X_j (a column in X), we first fit the Poisson regression on W to get the unconstrained coefficients, then remove negative ones and remove the corresponding feature variables in the W matrix.

Repeat this procedure until there are no negative coefficients left.

Calculate the log-likelihood value using these remaining variables of W matrix, we call the remaining variables the positive W matrix.

Next we add back one removed variable each time into the positive W matrix and calculate the new log-likelihood value. to decide if this variable will be added back, we do not need to refit the Poisson regression when calculating the new log-likelihood value. As the old coefficient matrix is a local maximization for the log-likelihood function with the remaining variables, the corresponding point is on the contour line and the derivative at that point should be 0 with respect to all remaining variables. When we add another variable with a small positive coefficient into the system, if we are near to the original maximum, the log-likelihood for the new point will either increase or decrease, depending whether the derivative is positive or negative. So if we want to see whether a variable could increase the log-likelihood, we can just add a very small weight ε for the new variable, then calculate the new log-likelihood with the new rescaled weight matrix. We need to rescale the $H_{j_{new}}$ vector, so that $H'_{j_{new}}1 = X'_j1,\ 1 = (1, 1, \cdots, 1)$. This is because we assume the data follow the Poisson distribution, so the sum of the observation X_j should be equal to the sum of its mean vector WH_j. As each column of W has unit sum, $H'_jW'1 = H'_j1 = X'_j1$.

We compare this new log-likelihood value with the old one. If it decreases, the derivative is negative which means point with positive weight on the new variable will decrease the log-likelihood. Then the new variable should not be added. If the new one is larger than the old one, add this variable into the positive W matrix and do a Poisson regression on this new positive W matrix again and repeat the above steps until no variable can be added.

In this way, we can make sure that each time we decide to add a new variable to the positive W matrix, the likelihood becomes larger. This procedure keeps the log-likelihood function increasing under the constraints that all elements remain non-negative.

To see that the algorithm will converge, a key point is that our algorithm is only dealing with the discrete part of the optimization, and the Poisson regression takes care of the continuous optimization. Since we are optimizing over a finite number of possible sets of positive variables, convergence is guaranteed by the fact that each step increases the likelihood.

3.3 Choosing the number of types

3.3.1 Review of Wilcoxon rank-sum test

When the data can not be assumed following a normal distribution, the t-test is not appropriate for comparing two samples to assess whether they have the same mean. The Wilcoxon ranked test (also called the Mann-Whitney U test) is a nonparametric test for this situation, especially when one population tends to have larger values than the other. (Wilcoxon F. 1945) [10]

It is assumed that:

1 All the observations from both samples are independent from each other,

2 The responses can be compared,

3 The distributions of both groups are equal under the null hypothesis,

4 Under the alternative hypothesis, the probability of an observation from one population (X) exceeding an observation from the second population (Y) is not equal to 0.5.

To do a Wilcoxon rank-sum test, perform the following steps.

1 Rank the data. That means replace the data by their ranks, from smallest to greatest. For example, the data are:

Sample 1: 500 560 700 680 630 660

Sample 2: 650 510 560

they will be replaced by their ranks

Sample 1: 1 3.5 9 8 5 7

Sample 2: 6 2 3.5

If there are two (or more) tied values, like $(560, 560)$, their ranks are both the average of the ranks that would be assigned if they were different.

2 Calculate the Wilcoxon statistics Rs, as the sum of ranks in the first sample. In the example, $Rs = 33.5$.

Let n_1 and n_2 be the number of observations in the first and second sample. Here $n_1 = 6, n_2 = 3$.

The mean and variance of Rs under null hypothesis are

$$\mu_{Rs} = n_1(n_1 + n_2)/2,$$

$${\sigma_{Rs}}^2 = n_1 n_2(n_1 + n_2 + 1)/12.$$

So here $\mu_{Rs} = 27$, ${\sigma_{Rs}}^2 = 15$. Rs is approximately normally distributed.

3 The observed test statistic is

$$Z_{obs} = \frac{Rs - \mu_{Rs}}{\sigma_{Rs}} = \frac{33.5 - 27}{\sqrt{15}} = 1.678$$

Here the null hypothesis is: both distributions for the two samples are the same. So for the alternative that two distributions are different, the p-value is

$$P(|Z| > 1.678) = 2 \times 0.0475 = 0.095$$

This means under the level of 0.05, we could not reject the null hypothesis.

3.3.2 method for choosing the number of types

The number of types for each class of observations should be chosen to best describe its own class but not to describe other classes or noise. For discrimination purpose, the number of types for each class should be chose to best separate the classes. Thus in order to choose the best number of types for the first class, we will look the deviance statistics to see how well the chosen types will fit the first class better than other classes. Since the types are chosen from the first class, to make the

comparison objective, the deviance statistics need to be calculated on a test set of the first class. The deviance statistics are not Normally distributed, thus we will use the Wilcoxon rank-sum test based on the deviance statistics to test how well the classes are separated. For each class, we will try a sequence of values for the number of types and find the best value to discriminate this class from other classes.

We use a 2-class data case as an example to illustrate the ideas. We use a r-fold cross-validation on training data for both classes. In each cross-validation, we separate training data into training fold and test fold. To choose number of types for class 1, we apply the following steps to a range of values for k:

1 For each fixed value k, fit k types on the training folds from class 1 to get W.

2 Fit the remaining test fold data from class 1 and one fold of data from class 2 on W.

3 Calculate the deviance for each fitting.

4 Use a Wilcoxon ranked test on these deviances to get Z-values.

5 Sum the values of Z statistics from all different cross-validations. The sum of Z statistics should follow a normal distribution with mean of zero and standard deviation of $\sqrt{r}$.

6 Choose the smallest k for which the sum of Z-value is within one standard deviation of the largest sum of Z-value, where the standard deviation is estimated by treating the Z-values from the different folds as a sample from the distribution.

By using r-fold cross-validation and combined Z-value, we can effectively increase the power of this test, which is particularly important when number of observations is small.

3.4 Prediction

With the fixed W, we apply the above algorithm on training data to calculate the training H and on test data to calculate the test H. After getting H matrix, we have effectively reduced the dimension from m to k. We can use an off-the-shelf supervised

learning method to predict the class labels since $k < n$. Let Y be the response variable such that $Y = 0$ if this observation is in class 1, and $Y = 1$ otherwise. In the following chapter, we treat this training H as new variables and perform a logistic regression on H. We choose logistic regression because of its simplicity. The trained logistic regression model can then be used to do prediction on the test H.

Chapter 4

Application of supervised NMF

4.1 Simulation

4.1.1 Simulation Design

We simulate data according to our proposed model. As the data follow a Poisson distribution with mean $(WH)_{ij}$, to generate these data, we first generate the mean WH.

The mean is a linear combination of different features (different columns of W). So we fix W. We use the features obtained by applying NMF to the two classes in the IBD dataset (Qin 2010) [14].

We generate the H matrix by generating each entry from a uniform distribution on $[0, 1]$ then normalizing the column vectors so that the column sums of H are equal to the column sums of IBD data.

The WH gives us the mean and we add four levels of noise to the product WH. The noise is normally distributed with mean 0 and four different standard deviations.

$$noise_0 : sd_0 = 0$$

$$noise_1 : sd_1 = sd(W)/4$$

$$noise_2 : sd_2 = sd(W)/2$$

$$noise_3 : sd_3 = sd(W)$$

Here the $sd(W)$ is a vector of standard deviations for each row of W. This is a vector of length m (the number of genes or OTUs) which measures the variability for each type of gene or OTU across different people.

The column of WH plus the noise is the Poisson mean we use in the simulation. Each element of X is generated by independent Poisson distribution with the mean given by above mean matrix.

We simulate data with number of types 2, 5, 10 for class 1 and 3, 6, 9 for class 2. So the number of different combinations is 9 in total. They are 2&3, 2&6, 2&9, 5&3, 5&6, 5&9, 10&3, 10&6, 10&9. Considering the different noise levels, we have 36 scenarios. For each scenario, we simulate 25 replicates. In each replicate, we simulate 200 observations for each class. Then we separate the data into two parts: the first 200 observations as the training data and the other 200 as the test data.

4.1.2 Simulation Results

We choose the number of types from the training data using a 10-fold cross-validation and the method in Section 3.1.3. The Tables 4.1-4.6 summarize the results for the number of types chosen by our methods. Tables 4.1 -4.3 summarize the results for the number of types chosen for class 1. And Tables 4.4-4.6 summarize the results for the number of types chosen for class 2.

class 2		2 types for class 1			
		maximum	minimum	mean	sd
3 types	$noise_0$	7	2	2.24	1.0116
	$noise_1$	5	2	2.20	0.7071
	$noise_2$	6	2	2.72	1.2083
	$noise_3$	10	2	2.76	1.6653
6 types	$noise_0$	7	2	2.4	1.1547
	$noise_1$	7	2	2.32	1.0296
	$noise_2$	6	2	2.44	1.0033
	$noise_3$	8	2	2.88	1.6663
9 types	$noise_0$	9	2	2.4	1.4434
	$noise_1$	8	2	2.36	1.3191
	$noise_2$	9	2	2.36	1.4399
	$noise_3$	7	2	2.44	1.0832

Table 4.1: Simulation summary of the predicted number of types for class 1 with the true number of types for class 1 being 2

The Tables show that the algorithm tends to output slightly larger values than the true number of types in most scenarios, but the true number of types mostly are within one standard deviation of the mean of chosen number of types. When the true number of types is 10, the chosen number of types is always smaller than 10. The reason is that 10 is the largest value among the range of values we tried when choosing the number of types. We will try values from 1 to 12 or even larger numbers

class 2		5 types for class 1			
		maximum	minimum	mean	sd
3 types	$noise_0$	9	5	6.04	1.1358
	$noise_1$	10	3	6.72	1.4583
	$noise_2$	9	5	6.44	1.1576
	$noise_3$	10	5	7.12	1.6411
6 types	$noise_0$	9	5	6.92	1.2557
	$noise_1$	10	5	6.52	1.5843
	$noise_2$	10	5	6.84	1.3441
	$noise_3$	10	4	6.4	1.8708
9 types	$noise_0$	9	5	6.56	1.4166
	$noise_1$	10	5	6.76	1.4166
	$noise_2$	9	5	6.92	1.4978
	$noise_3$	10	4	6.84	1.7243

Table 4.2: Simulation summary of the predicted number of types for class 1 with the true number of types for class 1 being 5

in the future to see if the results will improve. The Tables also show that in most replicates, when the noise level become higher, the difference between the mean and the true number of types will increase. The results demonstrate that our method is quite effective in finding an appropriate number of types.

After the number of types are chosen, we perform a prediction on the test data using the trained logistic regression model on the training data based on the chosen number of types for each simulated data set. The prediction errors are shown in Tables 4.7-4.10 for different noise levels. The tables list the average test error for 25 data sets. For example, 0.04 means 0.04% test error among total of 5000 observations. In each table, the rows are the true number of types for class 1 and the columns are the true number of types for class 2.

We find when the true number of types get larger, the prediction errors tend to increase. That may be because we have chosen better number of types in the cases the true numbers of types are small. But in total, the prediction errors are quite small for all cases which means our supervised NMF method works well in prediction. The applications of our supervised NMF on real data are shown in the next section.

class 2		10 types for class 1			
		maximum	minimum	mean	sd
3 types	$noise_0$	10	7	8.72	0.8907
	$noise_1$	10	7	8.96	0.6758
	$noise_2$	10	7	8.72	0.7916
	$noise_3$	10	4	8.52	1.4177
6 types	$noise_0$	10	7	8.72	0.6782
	$noise_1$	10	8	8.92	0.6403
	$noise_2$	10	6	8.92	0.9539
	$noise_3$	10	5	8.36	1.6042
9 types	$noise_0$	10	6	8.68	0.8021
	$noise_1$	10	7	8.88	0.8327
	$noise_2$	10	7	8.64	0.8103
	$noise_3$	10	6	8.68	1.3450

Table 4.3: Simulation summary of the predicted number of types for class 1 with the true number of types for class 1 being 10

4.2 Real Data Analysis

We apply this supervised NMF on the animal data (Muegge 2011) [3] and the moving picture data (Caporaso 2011) [16]. The results are shown in the following subsections.

4.2.1 Animal data

We only use the carnivores and herbivores from the animals dataset (Muegge 2011) [3]. So the data we use here contains metagenomic sequencing of fecal samples from 28 mammals: 7 carnivores and 21 herbivores (Muegge 2011) [3]. As the number of observations are too small, we do a 7-fold cross-validation on the whole data. Each time we use 6-folds as training data and the remaining observations as test data. The Z-values are plotted in Figure 4.1.

The plots show that when the number of types change from 2 to 5, the Z-values stay the same. So 2 types are enough for both classes. Then we use the chosen number of types to get the weight matrix for the training data and test data and perform a prediction on the test data. The weight matrix for both training and test data in one time of the 7-fold cross-validation is shown in Figure 4.2.

class 1		3 types for class 2			
		maximum	minimum	mean	sd
2 types	$noise_0$	9	3	3.64	1.3808
	$noise_1$	6	3	3.52	0.8226
	$noise_2$	9	3	3.76	1.5078
	$noise_3$	7	3	3.8	1.3229
5 types	$noise_0$	7	3	4.04	1.3064
	$noise_1$	7	3	3.64	1.1136
	$noise_2$	5	3	3.4	0.7071
	$noise_3$	9	3	3.92	1.6563
10 types	$noise_0$	9	3	4.72	1.5948
	$noise_1$	10	3	3.92	1.7554
	$noise_2$	9	3	3.88	1.6155
	$noise_3$	10	3	4.36	2.0182

Table 4.4: Simulation summary of the predicted number of types for class 2 with the true number of types for class 2 being 3

The plot shows that both training Carnivores and test Carnivores could be well separated from Herbivores.

Both the training and test errors are 0 in each time of the 7-fold cross-validation data split.

The prediction errors are all 0 means our algorithm could separate the two classes of animals quite well. The huge number of variables in original data could be reduced to 4 features (2 for each class), which means the classes of animals are determined by four features. Also from the plot, we can also see that although we did not supervise the distinction between the two types of herbivores, there is some reasonable degree of separation between these classes.

4.2.2 Moving picture data

We also apply our method to the tongues data and guts data in the moving picture dataset (Caporaso 2011) [16].

Tongues data

As the data are time based, we choose the first 70 time points' observations of all 135 observations of person 1 and the first 190 time points of all 373 observations of person 2 as training data. The rest data are test data.

class 1		6 types for class 2			
		maximum	minimum	mean	sd
2 types	$noise_0$	10	6	7.12	1.09
	$noise_1$	9	6	7.16	1.0677
	$noise_2$	9	6	6.84	0.9866
	$noise_3$	10	6	8.24	1.5351
5 types	$noise_0$	9	6	7	1.0000
	$noise_1$	9	6	7.88	1.3940
	$noise_2$	10	6	7.36	1.4399
	$noise_3$	10	5	7.72	1.4583
10 types	$noise_0$	10	6	7.44	1.4457
	$noise_1$	10	6	7.68	1.2152
	$noise_2$	10	6	7.6	1.5
	$noise_3$	10	6	7.76	1.3317

Table 4.5: Simulation summary of the predicted number of types for class 2 with the true number of types for class 2 being 6

We perform a 5-fold cross-validation on the training data to find the number of types for both individuals. The Z-values are plotted in Figure 4.3.

The Figure 4.3 shows that 2 types are appropriate for person 1 as the mean Z-value when number of types is 2 falls within one standard deviation of the Z-value when number of types is 3. For person 2, 10 types are appropriate. Then based on the 12 features, we get the training H and test H. The weight matrix of the test data for 2 types from each class is shown in Figure 4.4 as this actually gives better prediction than 210. The prediction errors for different number of types are shown in Table 4.11.

Figure 4.4 shows that most of the observations could be separated correctly to their own classes. Looking at just these 4 features, most of observations in tongues data could be classified according to which individual they come from. And Table 4.11 show that the prediction error is smallest when the types are 22 for these two classes. Thats different from what we get in our method. This may be because we use a logistic regression in prediction, which may not be the best model for the tongues data. We will nd a more suitable model for the data in the future. Looking at just these 4 features, most of observations in tongues data could be classified according to which individual they come from.

class 1		9 types for class 2			
		maximum	minimum	mean	sd
3 types	$noise_0$	10	8	9	0.7638
	$noise_1$	10	8	8.88	0.6658
	$noise_2$	10	8	9.04	0.6758
	$noise_3$	10	6	8.92	1.0770
6 types	$noise_0$	10	7	9	0.8165
	$noise_1$	10	8	8.88	0.6658
	$noise_2$	10	7	9.28	0.7371
	$noise_3$	10	7	8.64	0.8602
9 types	$noise_0$	10	8	9.04	0.8406
	$noise_1$	10	8	8.92	0.7594
	$noise_2$	10	7	8.6	0.8660
	$noise_3$	10	6	8.88	0.9274

Table 4.6: Simulation summary of the predicted number of types for class 2 with the true number of types for class 2 being 9

%	2 types	5 types	10 types
3 types	0.04	0.24	0.80
6 types	0.12	0.40	1.24
9 types	0.48	1.20	1.12

Table 4.7: prediction error with $noise_0$

Each feature in feature matrix of tongues data is plotted in Figure 4.5.

Figure 4.5 shows that in each column of feature matrix, only a few elements, around 18, are non-zero over 1151 elements in total. So the feature matrix of tongues data is highly sparse and each feature is only consists of a few genes or OTUs. So each observation is a combination of a few different super types in the type matrix W. Our interpretation is that the super types with non-zero coefficients for a given observation include the major community information for the observation.The biological interpretation of these features should be very interesting and will be pursued later.

Guts data

Same as above, we choose the first 70 time points' observations of all 131 observations of person 1 and the first 190 time points of all 336 observations of person 2 as training data. A 5-folds cross validation with our method of choosing number of types is

%	2 types	5 types	10 types
3 types	0.06	0.54	0.76
6 types	0.34	0.34	1.22
9 types	0.44	1.10	1.04

Table 4.8: prediction error with $noise_1$

%	2 types	5 types	10 types
3 types	0.00	0.42	0.68
6 types	0.10	0.56	1.08
9 types	0.54	1.20	0.80

Table 4.9: prediction error with $noise_2$

applied on the training data. The results are shown in Figure 4.6.

As almost all the Z-values are the same for person 1, we choose 2 types for person 1. And for person 2, the Z-value for 2 types lies within one standard deviation of the largest Z-value, so we also choose 2 types for person 2. Then we perform a prediction on the test data. The results are shown in Figure 4.7 and the prediction error is 0.008811 for the test data.

We see that both training and test data are perfectly separated into two individuals which means the distinguishing features of guts data are included in a matrix consisting of 4 features. This matrix contains sufficient information for classification.

Each feature in feature matrix of guts data is plotted in Figure 4.8.

Figure 4.8 shows that in each column of feature matrix, only a few elements, less than 20, are non-zero over 2563 elements in total. So the feature matrix of guts data is highly sparse and each feature should consist biologically interesting signatures for each of these two individuals.

%	2 types	5 types	10 types
3 types	0.12	0.48	0.60
6 types	0.24	0.30	0.78
9 types	0.30	0.64	0.70

Table 4.10: prediction error with $noise_3$

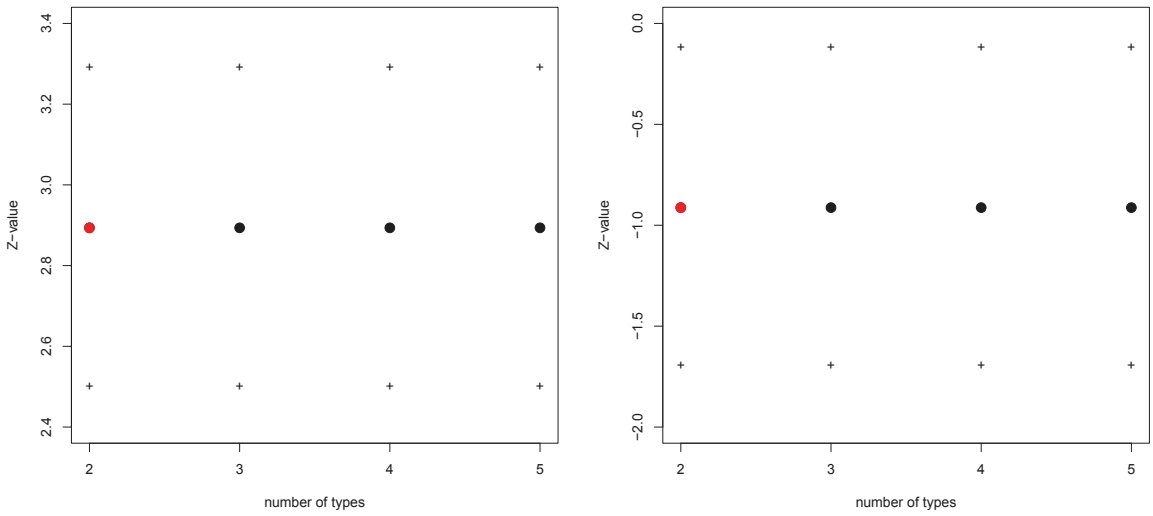


Figure 4.1: Z-values for animal dataset. The left panel is Z-values for the herbivores and the right panel is for the carnivores.

Figure 4.2: animals dataset weight matrix. The dark red points are training Foregut-fermenting Herbivores and light red ones are test Foregut-fermenting Herbivores, the dark green points are training Hindgut-fermenting Herbivores and the light green ones are test Hindgut-fermenting Herbivores, the dark blue points are training Carnivores and the light blue ones are test Carnivores.

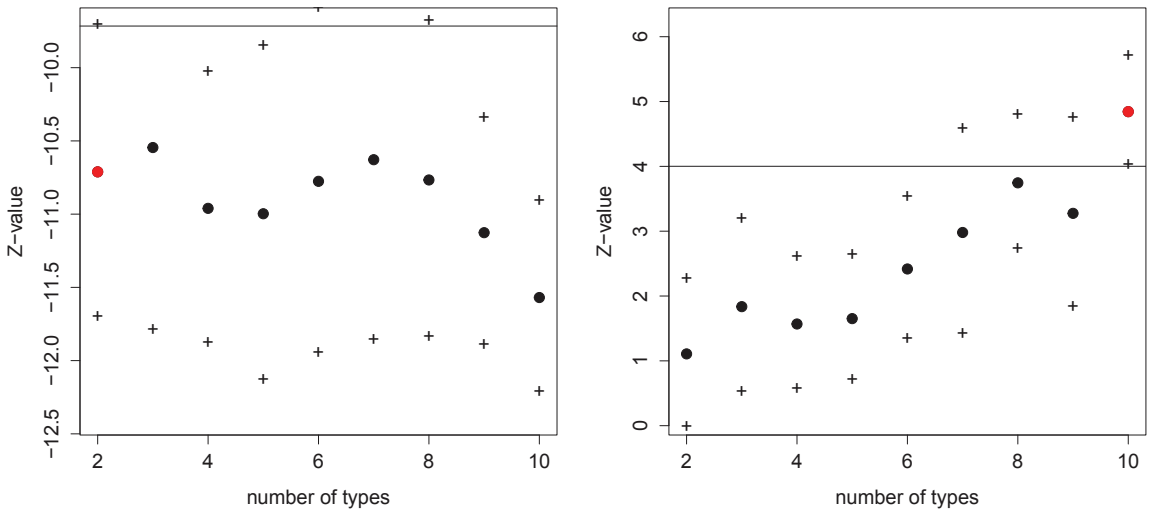


Figure 4.3: Z-values for tongues data. The left panel of Figure 4.3 shows Z-values for person 1's tongue and right panel shows Z-values for person 2's tongue.

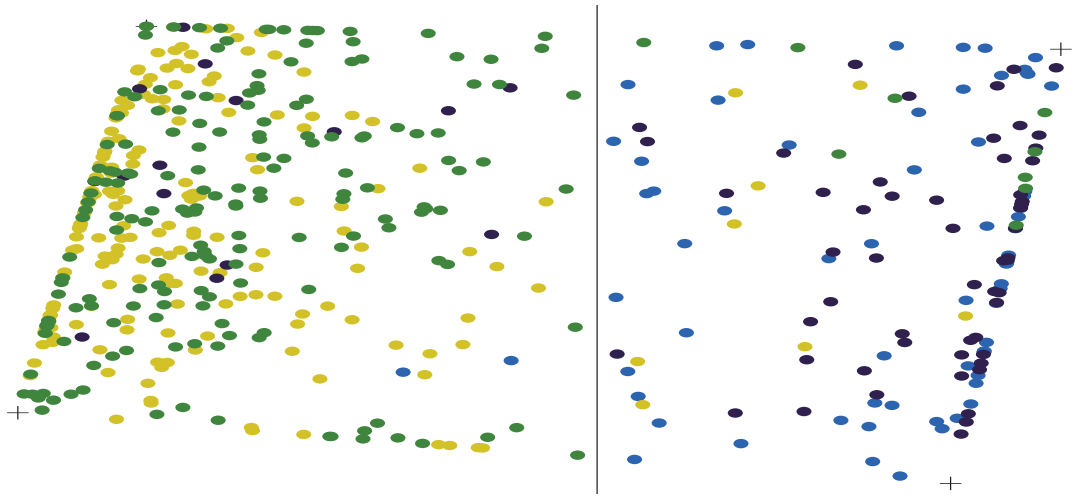
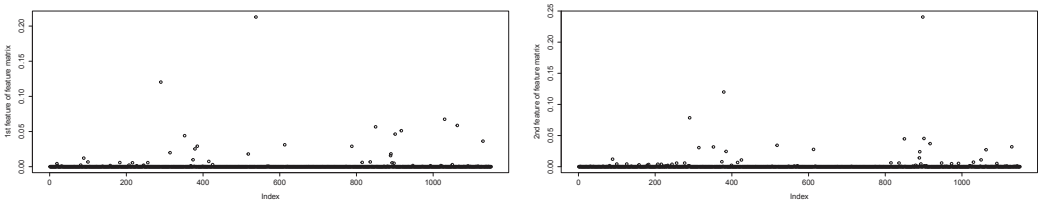


Figure 4.4: tongues data weight matrix. The dark blue points are training data from person 1 and dark green points are training data from person 2, the blue points are test data from person 1 and the dark yellow points are test data from person 2.

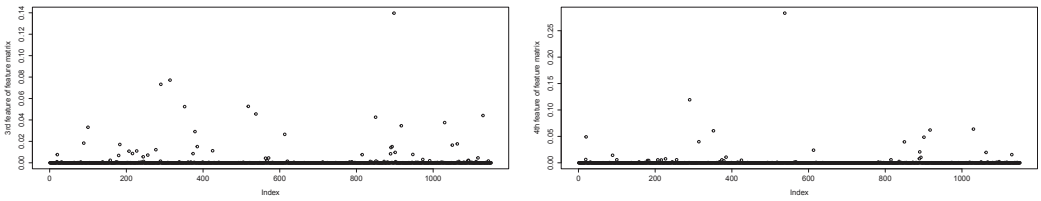
	training error	prediction error
type 2×2	0.01923	0.04032
type 2×3	0.01923	0.07258
type 2×4	0.03077	0.05242
type 2×5	0	0.04435
type 2×6	0	0.06048
type 2×8	0	0.06048
type 2×10	0	0.08065

Table 4.11: prediction errors for tongues data.



(a) First column in feature matrix of tongues data

(b) Second column in feature matrix of tongues data



(c) Third column in feature matrix of tongues data

(d) Forth column in feature matrix of tongues data

Figure 4.5: Plot of different features of tongues data

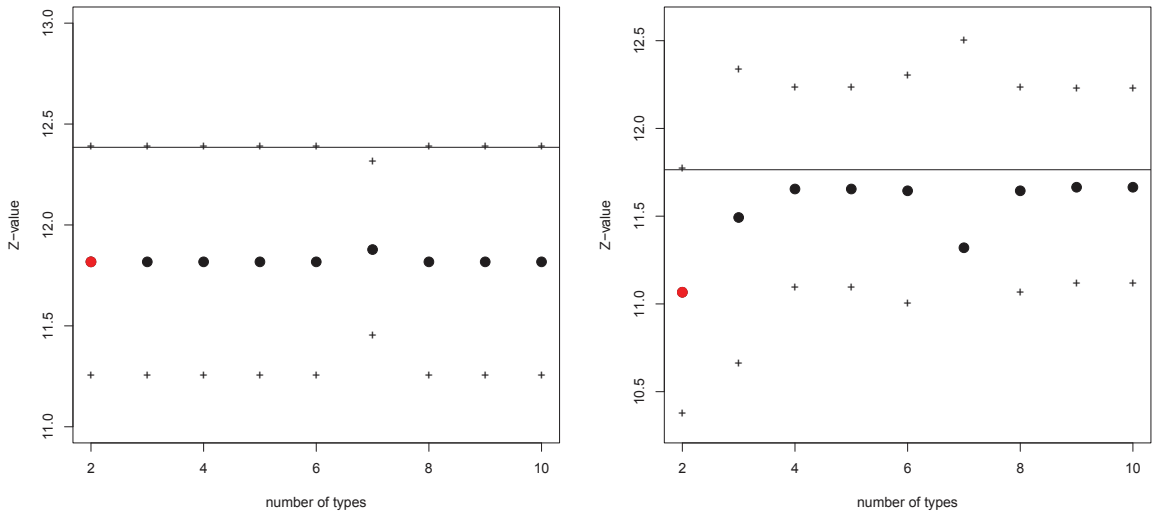


Figure 4.6: Z-values for guts data. The left panel of Figure 4.6 shows Z-values for person 1's gut and right panel of Figure 4.6 shows Z-values for person 2's gut.

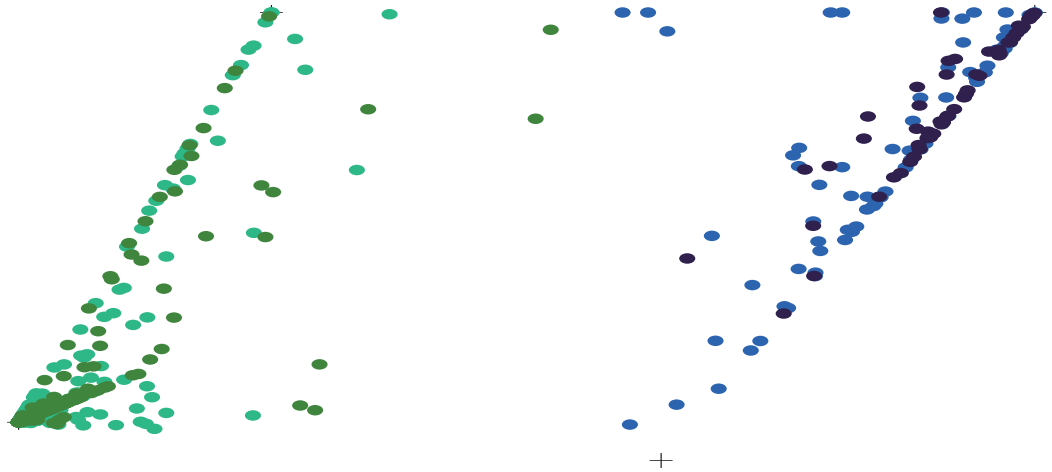
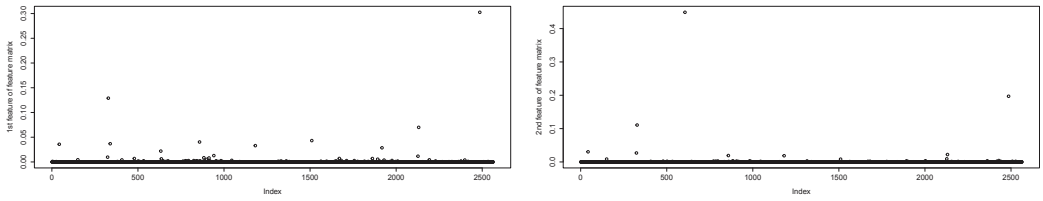
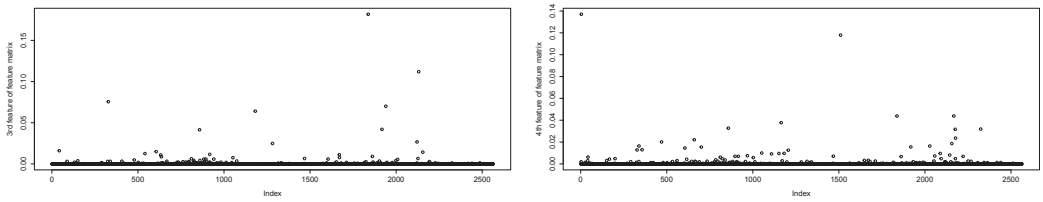


Figure 4.7: guts data weight matrix. The dark blue points are training data from person 1 and dark green points are training data from person 2, the blue points are test data from person 1 and the green points are test data from person 2.



(a) First column in feature matrix of guts data

(b) Second column in feature matrix of guts data



(c) Third column in feature matrix of guts data

(d) Forth column in feature matrix of guts data

Figure 4.8: Plot of different features of guts data

Chapter 5

Discussion

The results for simulation and real data show that our supervised NMF could recover the correct number of types based on which a good classification results can be achieved. The developed supervised NMF can effectively reduce the dimensionality of the data to a non-negative and most often sparse data matrix, which contain sufficient discriminative information for classification purpose. However, in both the animal and moving picture data examples, classification is not the study purpose. For IBD data, classification and prediction is important to some extent. In all metagenomic data analysis, the common important purpose is to find the community structure and function of different classes of objects. The effectiveness of classification based on H matrix only confirms that the super-types in W matrix are important types to describe the corresponding class of objects. A nice feature of NMF is that it is part based factorization. Recall the facial example (Lee and Seung 1999) [6] in Chapter 1, each face can be decomposed into a linear combination of typical nose, mouth, eye and so on. Thus W matrix is usually sparse too. These typical features are the community signatures for each class of objects in metagenomic analysis. Detailed work in future is needed to map these community signatures in W matrix to the metabolite pathways in the case that the data are metagenomic genes and to the phylogenetic trees in the case that data are measurement of OTUs.

Bibliography

[1] Fisher R. A. *The Use of Multiple Measurements in Taxonomic Problems.* Annals of Eugenics 7(2): 179-188, 1936.

[2] Coghlan Andy. *Milk fats clue to inflammatory bowel disease.* New Scientist, 2012.

[3] Muegge B. *Diet drives convergence in gut microbiome functions across mammalian phylogeny and within humans.* Science 332: 970-973, 2011.

[4] Lin C.-J. *On the convergence of multiplicative update algorithms for non-negative matrix factorization.* Technical Report, Department of Computer Science, National Taiwan University, 2005.

[5] Bertsekas D. *Nonlinear Programming.* Athena Scientific, Belmont, MA, 1999.

[6] Lee D.D. and Seung H.S. *Learning the parts of objects by non-negative matrix factorization.* Nature 401, 788-791, 1999.

[7] Lee D.D. and Seung H.S. *Algorithm for Non-negative Matrix Factorization.* Advances in Natural Information Processing Systems 13, 556-562, 2001.

[8] Gonzalez E. and Zhang Y. *Accelerating the Lee-Seung algorithm for nonnegative matrix factorization.* Tech. Rep. TR-05-02, Rice University, 2005.

[9] Shahnaz F., Berry M., and Plemmons R. *Document clustering using nonnegative matrix factorization.* Information Processing & Management 42 (2), 373-386, 2006.

[10] Wilcoxon F. *Individual comparisons by ranking methods.* Biometrics Bulletin 1 (6), 80-83, 1945.

[11] Schwarz G. *Estimating the dimension of a model.* Ann Stat, 6 (2), 461-464, 1978.

[12] Kim H. and Park H. *Sparse non-negative matrix factorizations via alternating non-negativity-constrained least squares for microarray data analysis.* Bioinformatics 23, 1495-1502, 2007.

[13] Piper J., Pauca V., Plemmons R., and Giffin M. *Object characterization from spectral data using nonnegative factorization and information theory.* In: Proc. 2004 AMOS Technical Conference. Maui, HI, 2004.

[14] Qin J. *A human gut microbial gene catalogue established by metagenomic sequencing.* Nature 464: 59-65, 2010.

[15] Wooley J.C., Godzik A., and Friedberg I. *A primer on Metagenomics.* PLOS Computational Biology., 2012.

[16] Caporaso J.G., Lauber C.L., Costello E.K., Berg-Lyons D., Gonzalez A., Stombaugh J., Knights D., Gajer P. andRavel J., Fierer N., Gordon J.I., and Knight R. *Moving pictures of the human microbiome.* Genome Biol, 12: R50, 2011.

[17] Avriel M. *Nonlinear Programming: Analysis and Methods.* Dover Publishing. ISBN 0486432270, 2003.

[18] Chu M., Diele F., Plemmons R., and Ragni S. *Optimality, computation and interpretation of nonnegative matrix factorizations.* $URLhttp$: $//www.wfu.edu/\ plemmons/papers/chu\_ple.pdf$, 2004.

[19] Priestley M.B. *Spectral Analysis and Time Series.* Academic Press. ISBN 0-12-564922-3, 1981.

[20] Berry M.W. and Browne M. *Algorithms and Applications for Approximate Nonnegative Matrix Factorization.* Department of Computer Science, University of Tennessee, Knoxville, TN, 2006.

[21] Hoyer P. *Non-negative Matrix Factorization with Sparseness Constraints.* J. of Machine Learning Research 5, 1457-1469, 2001.

[22] Paatero P. and Tapper U. *Positive matrix factorization: A non-negative factor model with optimal utilization of error estimates of data values.* Environmetrics 5, 111-126, 1994.

[23] Stoica P. and Selén Y. *Model-order selection: A review of information criterion rules.* IEEE Signal Proc Mag, 21 (4): 36-47, 2004.

[24] Belhumeur P.N., Hespanha J.P., and Kriegman D. *Eigenfaces vs. Fisherfaces: Recognition Using Class Specific Linear Projection.* IEEE Trans. PAMI, 19(7), pp.711-720, 1997.

[25] Hastie T., Tibshirani R., and Friedman J. *The Elements of Statistical Learning: Data Ming, Inference, and Prediciton.* Spring-Verlag. 553, 2009.

[26] Wang Y., Jia C., Hu C., and Turk M. *Fisher non-negative matrix factorization for learning local features.* Asian Conference on Computer Vision, Korea, January 27-30, 2004.