University of Southampton

Faculty of Engineering and Physical Sciences

Electronics and Computer Science

Non-Negative Matrix Factorisation: Algorithms and Applications

by

Steven E. Squires

Thesis for the degree of Doctor of Philosophy

March 2019

UNIVERSITY OF SOUTHAMPTON

ABSTRACT

FACULTY OF ENGINEERING AND PHYSICAL SCIENCES
Electronics and Computer Science

Doctor of Philosophy

NON-NEGATIVE MATRIX FACTORISATION: ALGORITHMS AND APPLICATIONS

by Steven E. Squires

Non-negative matrix factorisation (NMF) is attractive in data analysis because it can produce a sparse and parts based representation of the data. In this thesis we investigate and demonstrate solutions to several aspects of NMF. In particular, we consider the oft overlooked issue of model selection utilising a principled approach using information theory, provide a method for including external data into the NMF formulation, implement an autoencoder framework that can perform variants of NMF, and extend this autoencoder approach using variational methods to produce a probabilistic form of NMF.

Two problems of model selection are explored in this thesis by taking a minimum description length (MDL) approach. First we produce and demonstrate a method using MDL to provide an estimate of the optimal subspace size to project onto. Secondly we extend this work by utilising MDL within the objective function for NMF to provide an automatic method of regularising the factorised matrices. The final representation should then be produced from a principled trade-off between accuracy and complexity reducing the problem of over-fitting.

In standard NMF we factorise our data into two matrices without consideration of any external sources of information. If we could include these exogenous drivers into the NMF formulation it might enable an improved factorisation to be formed using information not directly available to the internal data. Our solution to this problem, called XNMF, finds a combined representation which includes these external drivers by utilising an extended version of the popular multiplicative update method. We prove theoretically that our method is guaranteed to reduce the objective function monotonically and that it also does so empirically by testing on financial data. In addition, we demonstrate that XNMF produces an improved representation and that it may produce better clusters in the data than standard NMF. We also investigate the broader utility of XNMF

by application to a problem in biology - spatial proteomics. We demonstrate that there are useful data analysis advantages to using XNMF and that there may be many other biological applications of this technique.

Our next contribution is to explore the use of autoencoders in producing the NMF factorisation (AE-NMF). We provide a set of methods that can produce the two factorised matrices and investigate advantages of using AE-NMF. In particular, we show that AE-NMF allows for the easy extension of NMF to perform a wider variety of tasks and with different objective functions.

Finally, we produce a method of combining AE-NMF with variational autoencoders to produce a probabilistic version of NMF. This method, unlike standard NMF, allows us to: generate new data; provides a probabilistic representation between the input and latent space; gives some sense of the uncertainy in our representation; and produces a representation that is regularised in a principled manner. Unlike some other forms of probabilistic NMF this approach does not require the use of sampling such as the Markov Chain Monte Carlo technique.

Contents

List of Figures

List of Tables

Declaration of Authorship

I, Steven E. Squires , declare that the thesis entitled *Non-Negative Matrix Factorisation: Algorithms and Applications* and the work presented in the thesis are both my own, and have been generated by me as the result of my own original research. I confirm that:

- this work was done wholly or mainly while in candidature for a research degree at this University;
- where any part of this thesis has previously been submitted for a degree or any other qualification at this University or any other institution, this has been clearly stated;
- where I have consulted the published work of others, this is always clearly attributed;
- where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work;
- I have acknowledged all main sources of help;
- where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself;
- parts of this work have been published as: [126, 125, 124]

Signed: _____

Date: _____

Acknowledgements

I would like to start by thanking my joint supervisors Niranjan and Adam. I could not have asked for a better combination to have been advised by over the last three years. Thank you both for the advice, encouragement, support and suggestions.

I also want to thank all the academics and students of the VLC, it has been a brilliant environment to work in, but also to interact in. Everyone has been so friendly and approachable its made the lab a pleasure to come into everyday.

To all the folks of the VLCCC, thanks for a fantastic few years. I did not expect to meet such great people or make such good friends during my PhD. I'm going to miss the trips to Stags, lunch, coffee/tea breaks, football, squash, barbecues and the rest. It's been brilliant.

To my family, thank you for all your care and support over the years: to Mum and Dad, Matt, Steph, William and Emily, and Ali and Ben. Finally, to Hannah, when I started my PhD was Dr Hannah Salter and now that I've finished is Mrs (Dr) Hannah Squires, I could not have done it without you.

Chapter 1

Introduction

1.1 Motivation

Dimensionality reduction methods can lead to better predictions and more interpretable data. Predictions can be improved when applying supervised learning techiques to dimensionality reduced data for a range of reasons, including noise reduction, reducing the issue of the curse of dimensionality and emphasising important relationships relevant to the task. Lower dimensional spaces can be significantly easier to interpret as we can focus on fewer dimensions and have reduced or removed a lot of the data which does not provide the main correlations but does obscure the key features.

Non-negative matrix factorisation (NMF) is a linear dimensionality reduction technique with similarities to, amongst others, principal component analysis (PCA), independent component analysis (ICA) and linear discriminant analysis. In comparison to the dominant technique, PCA, NMF can be much easier to interpret due to the non-negativity constraint which tends to produce a sparser, more parts based representation which is easier to understand, although this comes at the expense of a lower reconstruction accuracy compared to PCA.

In this thesis we are focused on producing new algorithms to improve NMF and on applying them to new applications to broaden the applicability of this technique. Interpretability of results is becoming increasingly important because modern methods, such as deep learning, produce very high prediction accuracy but simultaneously struggle to provide one of the other key feature of machine learning, explaining why a prediction is made. At its best, NMF offers greater interpretability than other methods and, with further enhancements and extensions, some of which we have contributed, will provide

continued improvements in understanding as we go forward. In this thesis we have identified and attempted to solve some weaknesses in NMF while also putting forward some methods to extend and improve this approach.

1.2 Contributions

We make contributions in four broad themes in this thesis:

1. **The use of minimum description length in NMF.** We apply minimum description length (MDL) to NMF to provide solutions to two problems: (1) how to select the rank, or subspace size (Chapter 3); (2) how to trade-off between the accuracy of the NMF reconstruction and the complexity of the model created (Chapter 4).

2. **Including external information into the NMF formulation.** We provide a method, based on multiplicative updates, to find a solution to this extended version of NMF, called XNMF. We provide a theoretical proof along with empirical evidence of its efficacy on financial data. We then demonstrate how XNMF can be used to provide data analysis solutions for a biological problem (see Chapters 5 and 6).

3. **Using autoencoders to perform NMF.** We demonstrate how to use an autoencoder to perform NMF (AE-NMF) and explore some of the advantages of doing so over standard NMF. We also show how AE-NMF can be used to perform XNMF and to include new objective functions (see Chapter 7).

4. **Producing a probabilistic version of AE-NMF using a variational autoencoder**. We adapt the variational autoencoder to apply to AE-NMF. Consequently, we produce a method which performs a probabilistic version of NMF using an autoencoder (see Chapter 8).

1.3 Publications

These publications, presentations and posters are based purely on research conducted during my PhD, except for the publication and oral presentatation "A Method of Integrating Spatial Proteomics and Protein-Protein Interaction Network Data" (included in Appendix D) along with the associated poster "Combining spatial proteomics and PPI network data" for which part of the work was conducted during my previous MSc in Computer Science with the rest completed during my PhD.

Publications

- Steven Squires, Adam Prügel-Bennett & Mahesan Niranjan (Under review), A Variational Autoencoder for Probabilistic Non-Negative Matrix Factorisation *International Conference on Learning Representations*.
- Steven Squires, Adam Prügel-Bennett & Mahesan Niranjan (Under review), Minimum Description Length as an Objective Function for Non-negative Matrix Factorisation. *Pattern Recognition*.
- Steven Squires, Luis Montesdeoca, Adam Prügel-Bennett & Mahesan Niranjan, Non-Negative Matrix Factorisation with Exogenous Inputs for Modelling Financial Data, In: *International Conference on Neural Information Processing*, Springer International Publishing, 2017.
- Steven Squires, Rob Ewing, Adam Prügel-Bennett & Mahesan Niranjan, A Method of Integrating Spatial Proteomics and Protein-Protein Interaction Network Data, In: *International Conference on Neural Information Processing*, Springer International Publishing, 2017.
- Steven Squires, Adam Prügel-Bennett & Mahesan Niranjan, Rank Selection in Non-negative Matrix Factorization using Minimum Description Length, *Neural Computation*, 2017.

Posters and Presentations

- Oral Presentation - Steven Squires, Luis Montesdeoca, Adam Prügel-Bennett & Mahesan Niranjan, Non-Negative Matrix Factorisation with Exogenous Inputs for Modelling Financial Data, *International Conference on Neural Information Processing*, 2017
- Oral Presentation - Steven Squires, Rob Ewing, Adam Prügel-Bennett & Mahesan Niranjan, A Method of Integrating Spatial Proteomics and Protein-Protein Interaction Network Data, *International Conference on Neural Information Processing*, 2017
- Poster (peer reviewed abstract) - Steven Squires, Adam Prügel-Bennett & Mahesan Niranjan, Minimum Description Length as an Objective Function for Non-negative Matrix Factorisation, *Advances in Data Science*, 2017
- Poster (peer reviewed extended abstract) - Steven Squires, Adam Prügel-Bennett & Mahesan Niranjan (2016), Combining spatial proteomics and PPI network data, *NIPS Computational Biology Workshop*, 2016.

1.4 Report Organisation

This report is structured as follows:

- *Chapter 2.* We begin by reviewing key concepts in the NMF field. We focus on a broad explanation of how NMF works, where it fits within machine learning, how it relates to other techniques and why it is of interest. As this thesis covers different themes within NMF we also review specific literature as necessary at the start of each chapter,

- *Chapter 3.* We discuss the importance, and current methods, of rank selection in NMF. We consider the weaknesses of some of the prominent techniques and then introduce the theoretical background behind our novel contribution using MDL. We demonstrate the efficacy of using MDL to select the subspace size on synthetic and real data.

- *Chapter 4.* We discuss potential downsides of the Frobenius norm objective function currently widely used in NMF. We then introduce the theory and implementation of using MDL as an objective function to create a different subspace representation. We demonstrate that our objective function and related algorithms find good solutions on real and semi-synthetic data

- *Chapter 5.* We describe the motivation for including additional information into the NMF formulation (XNMF). Then we introduce our new formulation and algorithm for finding solutions, proving that we monotonically reduce the objective function. We demonstrate empirically on financial data that XNMF can produce more stable clusters than standard NMF and produce an improved representation compared to standard NMF.

- *Chapter 6.* We use XNMF to integrate spatial proteomics and protein-protein interaction networks, consequently showing that XNMF can meet several of the key objectives of data analysis in a biological context.

- *Chapter 7.* We consider the similarities between autoencoders and NMF, and then discuss and apply the constraints necessary to make an autoencoder perform NMF. We demonstrate some of the advantages of AE-NMF and show how it can enable a broader use of NMF.

- *Chapter 8.* We produce a method for performing AE-NMF in a probabilistic manner. We demonstrate the effectiveness of this technique and how it can improve on standard NMF.

- *Chapter 9.* In this final chapter we draw general conclusions and provide a variety of potential directions for future work.

Chapter 2

Review of Key Concepts and Related Work

The work conducted for this thesis covers several areas of machine learning, therefore in each chapter we will review specific background and literature that relates to that particular piece of work. In this chapter we will discuss NMF itself in a broad manner to try and set the scene for this thesis and solidify the motivation of our research as well as reviewing some of what has been achieved in the field. In Section 2.1 we focus on the basics of NMF, including how it works and why it is of interest. Then in Section 2.2 we review where NMF sits within machine learning. In Section 2.3 we explain the relationship between NMF and some other similar techniques. In Sections 2.4, 2.5, 2.6 and 2.7, we review some algorithms, improvements, extensions and applications of NMF.

2.1 Basics of Non-Negative Matrix Factorisation

Consider a data matrix $\mathbf{V} \in \mathbb{R}^{m \times n}$ with m dimensions and n data points which has only non-negative elements. If we define two matrices, also with only non-negative elements: $\mathbf{W} \in \mathbb{R}^{m \times r}$ and $\mathbf{H} \in \mathbb{R}^{r \times n}$, then NMF can reduce the dimensionality of $\mathbf{V}$ through the approximation:

$$\mathbf{V} \approx \mathbf{WH} \tag{2.1}$$

where $r < \min(m, n)$ is the number of directions in the subspace of m that we are projecting onto. Pictorally NMF looks like:

$$\begin{bmatrix} \\ \mathbf{V} \\ \\ \end{bmatrix} = \begin{bmatrix} \\ \mathbf{W} \\ \\ \end{bmatrix} \times \begin{bmatrix} \mathbf{H} \end{bmatrix}$$

where the **W** and **H** matrices are both smaller than the original starting matrix, **V**.

The columns of **W** make up the new dimensions we are projecting onto. Each column of **H** represents the coefficients, or loadings, of each data point in this new subspace.

An informative way of looking at NMF, as pointed out by Lee and Seung [73], is to study each column of **V** at a time (that is each data point). Equation (2.1) then reduces to $\mathbf{v}_j \approx \mathbf{W}\mathbf{h}_j$ where $\mathbf{v}_j$ and $\mathbf{h}_j$ are column vectors of **V** and **H** respectively. The original m-dimensional data point, $\mathbf{v}_j$, is now represented in this new space by the r-dimensional $\mathbf{h}_j$. If we then consider each row (dimension) of **V** independently, then the i^{th} row of $\mathbf{v}_j$ is $V_{i,j}$ and the approximation is: $V_{i,j} = \sum_{k=1}^{r} W_{i,k}H_{k,j}$, i.e. $V_{i,j}$, is approximated as a linear combination of all the elements of $\mathbf{h}_j$.

A critical question is then, what does this new subspace represent? A practical example from facial recognition applications is that each column of **W** represents a feature of a face, such as a nose or mouth. The face is then built up from these features. Another example is in textual recognition. Here each column of **W** can be considered as a "topic", such as biology or physics or chemistry (if thinking of school science textbooks chapters for example). The topic is a combination of words (or weighted words). So the physics column of **W** might have high weightings of words such as Newton, gravity or force. Each text document would then be represented, via the columns of **H**, as a combination of the physics, chemistry and biology topics. In NMF, therefore, two processes must occur: one finds a new basis (the features, **W**) while the second finds the best combination of this new basis for each data point.

The tendency for NMF to produce a sparse and parts based representation is due to the non-negativity constraint. In PCA if an undesirable feature is included from one basis vector, it can often be subtracted away by another basis vector. This enables a non-sparse approach. In NMF by contrast, there is no subtraction possible. As only addition is allowed, the features are encouraged to be sparse and parts based to allow an effective build up of the whole [42]. It should be noted that NMF does not always produce a sparse and parts based solution, this limitation of standard NMF will be discussed later in this chapter.

NMF contrasts with other linear reduction techniques such as PCA which reduces the dimension but in a variance preserving, non-sparse and holistic manner, rather than a sparse, parts-based way. In a facial recognition example the basis vectors from PCA

become a distorted version of the whole face, rather than features of the face [72]. An advantage of NMF is that the form produced is potentially much more interpretable (although exactly what we mean when discussing "interpretability" has been challenged [79][1]) than the "eigenface" approach of PCA. In Figure 2.1 we show a comparison of the basis that is used to build up an image of a faces for PCA (left) and NMF (right). That is, we show the columns of $\mathbf{W}$ reshaped into 19×19 images. The holistic nature of PCA and the parts based nature of NMF are clear to see (for details on this data-set, see Chapter 3). The addition of topics or features is an intuitive way of building up a whole. Moreover, in some circumstances a parts based representation may be more useful than a holistic approach. For example, NMF can pick out faces which are occluded by the addition of sunglasses better than methods such as PCA would [42]. As the parts based approach selects features rather than the whole, when individual features are removed the rest of the features are unaffected (or at least significantly less affected).

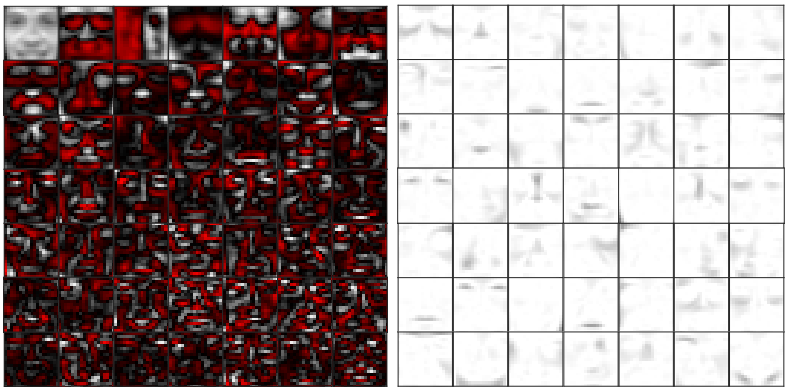


FIGURE 2.1: (Left) The basis that PCA uses to reconstruct a face. (Right) The basis that NMF uses to reconstruct a face. The holistic nature of PCA is easy to compare with the parts based nature of NMF for this data-set.

There are also several applications where it is not physically possible to have negative values, such as hyperspectral mixing [95]. In this cases allowing negative components is going against the physical constraints that exist in the real world. In addition, if negative terms are allowed the components that make up the representation cannot be true features as they contain negative values so are not built the same as the whole.

It has also been argued that human understanding of the whole is a function of the parts that make it up [72, 106, 135]. If this claim is true, then NMF functions in a manner closer to intelligent cognition than holistic approaches. This may be a contentious claim considering the poor general understanding of human intelligence. In any case, even if this claim were proved to be spurious, NMF would still be a highly useful technique for the other reasons outlined above.

[1]Their claim is that most people take interpretability as a positive idea but without specifying exactly what they mean by interpretability or exactly why it is useful. In this section we lay out what we mean by interpretability and why we think it is important.

There is a range of algorithms for NMF, some of which we will discuss in Section 2.4, most of them involve minimising an objective function such as:

$$\min_{\mathbf{W},\mathbf{H}} ||\mathbf{V} - \mathbf{WH}||^2_{\text{Fro}} \text{ subject to } W_{i,j} \geq 0, H_{i,j} \geq 0. \tag{2.2}$$

Other objective functions such as the Kullbeck-Leibler divergence are also options, but in this thesis we are concerned only with the Frobenius norm, which assumes that the noise in our data is Gaussian [42].

Most methods to find appropriate **W** and **H** matrices involve a two part process, one of **W** or **H** is held constant whilst the other is updated. While this iterative process ensures that the objective function is convex while one of the factorised matrices is held constant [143] the overall objective function remains non-convex [43].

In this section we have tried to explain the key features of the basic version of NMF, how it works and why it is of interest. In the next section we will lay out where NMF exists in the broader context of machine learning.

2.2 The Context of NMF within Machine Learning

This thesis is concerned with NMF which is a subfield of dimensionality reduction, itself a subfield of unsupervised learning, which is usually classified as one of the three main fields of machine learning (alongside supervised and reinforcement learning). In this section we will ground NMF within unsupervised learning, demonstrating where it sits with relation to other techniques.

2.2.1 Unsupervised Learning

Unsupervised learning differs from supervised learning in that there are no labels associated with the data-points. The aim is to extract information without an explicit target. Using unsupervised learning on its own involves extracting some form of representation, such as which clusters each data-point belongs to or what lower dimensional space it sits on. Or, as is quite common, outputs of an unsupervised method, treated as features of a problem domain, are used to feed to another technique, such as a supervised learning approach.

As with all machine learning tasks, when applying unsupervised learning, we are trying to find structure in the data. There are many types of unsupervised learning including:

clustering, Hebbian learning, anomaly detection and dimensionality reduction. Unsupervised learning is too broad a field to review in any detail, instead we focus on techniques closely related to NMF. However, as NMF can be used to perform clustering we want to briefy review that set of methods.

Clustering is one of the most popular methods of unsupervised learning, which aims to assign data-points to groups where all the members are similar in some way. There are different flavours of clustering with k-means [89] being one of the most popular. K-means is an iterative process which assigns data-points to a cluster in one step and then reassigns the location of the clusters in the next. It is a special case of a mixure Gaussian model [92], commonly these methods are performed using some form of expectation-maximisation algorithm [28].

In this thesis we focus on the dimensionality reduction side of unsupervised learning. Dimensionality reduction enables a decrease in the number of dimensions whilst retaining significant information. It is a technique which offers reduction in noise, increase in interpretability, increase in speed of subsequent techniques applied to the reduced data and partial solution to the curse of dimensionality.

2.2.2 Dimensionality Reduction

Dimensionality reduction is an important field in machine learning for several related reasons. In many classic statistical problems the number of data-points, n, was significantly larger than the number of dimensions, m. This meant that overfitting the data was rarely a significant concern. In modern data problems it is regularly the case that $m > n$ meaning that solutions can be found to the fitting problem that extract little real information, instead modelling noise, and provide scant predictive power on unseen data. By reducing the dimensionality m down to r where $r < m$ and ideally, $r \ll m$, we should be able to reduce this problem [23].

Another challenge with high dimensional data is that as the dimensionality increases the density of the datapoints in the space becomes increasingly small [62]. This problem is related to the previous overfitting problem in that it becomes simple to separate out the data into different classes - but does not provide an ability to distinguish between unseen data-points. This low density (high sparsity) of data-points makes it very hard to draw good distinguishing boundaries between clusters of points as there are few datapoint to utilise. There are also general problems with computing power required when the dimensionality is high. For example, if using a neural network, the larger the number of dimensions into the first layer, the more weights that must be computed throughout

the network [70]. Alternatively, if working with matrices, more dimensions require more computational power to invert or otherwise act upon the matrix.

There is a range of dimensionality reduction methods which can be performed either in a supervised or unsupervised manner. Supervised dimensionality reduction includes the use of regularisation such as the L_1 (lasso) norm [129] which preferentially sends to zeros those variables that add less value to the data fit. Another supervised method is linear discriminant analysis (LDA) which is a generalisation of Fisher discriminant analysis [36] and attempts to find a projection which effectively separates out the classes. Feature selection involves selecting those features that provide the most information [18], usually without actually changing the features themselves.

Factor analysis is a method that aims to describe some observed variables as a linear sum of fewer latent variables. It aims to find the relationship $\mathbf{X} - \boldsymbol{\mu} = \mathbf{AB} + \boldsymbol{\epsilon}$ where $\mathbf{X}$ are observed variables, $\boldsymbol{\mu}$ are their means, $\mathbf{A}$ and $\mathbf{B}$ hold common factors and latent variables respectively, and $\boldsymbol{\epsilon}$ are stochastic errors. Factor analysis aims to find both the common factors (similar to the $\mathbf{W}$ in NMF) and the latent variables ($\mathbf{H}$ in NMF).

A similar technique to factor analysis, and one of the most famous and effective dimensionality reduction techniques, is principal component analysis (PCA) [147], along with the related singular value decomposition (SVD) which reduces the dimension by projecting the data onto the directions that maximises the variance of the data. By ignoring the directions which hold less variance we can reduce the number of dimensions whilst keeping the maximum information.

By contrast, independent component analysis (ICA) [74, 20] is a technique that assumes that the signals being received are made up of independent sources. The aim is then to find the representation that separates out these sources as much as possible, for example by minimising the mutual information between the sources.

PCA, ICA and NMF all perform blind signal separation, which is a problem that requires the splitting up of mixed sources with the only data available being the end mixture. The differences between the techniques are down to different constraints and/or objective functions: PCA imposes orthogonality, NMF imposes non-negativity, and ICA maximises the separability of the sources. While there is considerable overlap between the techniques the differences in constraints and objectives can cause large differences in outcome and are suitable for different problems.

2.3 Relationship between NMF and Similar Techniques

In the previous section we considered the place of NMF in machine learning. In this section we want to look more specifically at a small number of techniques which are closely related to NMF and focus on how NMF is different and similar to them.

Many of the dimensionality reduction techniques appear to be very similar. Often the differences are only in small changes in certain constraints or slight changes in formulation. For example, PCA and NMF have considerable similarities [37]. PCA can be viewed as a matrix factorisation technique which takes an input vector $\mathbf{v} \in \mathbb{R}^{m\times 1}$ and reconstructs it using $\mathbf{A} \in \mathbb{R}^{m\times r}$ and $\mathbf{b} \in \mathbb{R}^{r\times 1}$ [37]. Here $\mathbf{A}$ represents the directions of the subspace and $\mathbf{b}$ the components of the datapoint in the new subspace. This formulation of PCA is the same as NMF, but the difference appears when the constraints are imposed, with PCA the columns of $\mathbf{A}$ must be orthogonal while in NMF all the components of both matrices must be non-negative. These seemingly small changes in constraints create significant differences in the representation formed and the algorithms required to carry out the factorisation.

In some respects, sparse principal component analysis (sPCA) [160] has even greater similarity to NMF. They impose the lasso penalty to reduce some of the components to zero, this should create a sparser representation, like NMF. However, while sPCA will produce a sparser solution it has no non-negativity requirement, it is trying to maximise the variance whilst keeping to the sparseness constraint, which is a different constraint to non-negativity.

A very close relationship exists between Probabilistic Latent Semantic Analysis (PLSA), a popular method for analysing natural language data, and NMF, one study showed that both are types of a more general method [14]. This idea was expanded upon to show that a solution of PLSA is also a solution of NMF when the KL divergence objective function is considered [40, 122]. While solutions of one of these methods is a solution of the other they are different algorithms, a fact that has been demonstrated to be useful if they are used together to jump out of local minima [32]. We note here that in this thesis we mainly focus on using the Frobenius norm objective function and do not consider the KL-divergence form.

2.4 Algorithms for NMF

Before algorithms for finding $\mathbf{W}$ and $\mathbf{H}$ can be applied an objective function needs to be chosen, usually these are the Frobenius norm or Kullback-Leibler divergence [143].

In this thesis we solely consider the Frobenius norm. It should briefly be noted that a potential problem emerges with respect to the objective function, in that actually maximising the similarity between $\mathbf{V}$ and $\mathbf{WH}$ is not exactly what we want to do. Ideally, we would find the $\mathbf{W}$ and $\mathbf{H}$ which together best compress the data and extract the features. This will be discussed further in Chapter 4. For now we will accept that $\mathbf{W}$ and $\mathbf{H}$ are found by minimising the objective function subject to constraints [42]:

$$\min \frac{1}{2}||\mathbf{V} - \mathbf{WH}||^2_{\text{Fro}} \text{ subject to } W_{i,j} \geq 0, H_{i,j} \geq 0 \quad \forall\, i, j. \tag{2.3}$$

The NMF algorithms are run until they either reach optimality conditions, an iteration limit or a time limit. Optimality conditions for $W_{i,j}$, required for all i and j, are:

$$W_{i,j} \geq 0, \quad (\nabla_{\mathbf{W}} F)_{i,j} = (\mathbf{WHH}^T - \mathbf{VH}^T)_{i,j} \geq 0, \quad (\mathbf{W} \circ \nabla_W F)_{i,j} = 0 \tag{2.4}$$

and similarly for $H_{i,j}$. The $\circ$ term denotes component-wise multiplication and F is the objective function. All three terms must be true for an optimal solution: all elements of $\mathbf{W}$ must be non-negative by definition; if $(\nabla_{\mathbf{W}} F)_{i,j} = 0$ then a minimum has been reached; if $(\nabla_{\mathbf{W}} F)_{i,j} > 0$ and the other conditions hold then $W_{i,j} = 0$ and cannot become smaller; the third term ensures that either $(\nabla_W F)_{i,j} = 0$ or $W_{i,j} = 0$. So there are two scenarios that give optimal solutions: 1) if $(\nabla_{\mathbf{W}} F)_{i,j} = 0$ then a stationary point has been reached; 2) if $W_{i,j} = 0$ and $(\nabla_{\mathbf{W}} F)_{i,j} > 0$ then the minimum occurs when $W_{i,j} < 0$ which is unreachable within NMF constraints so $W_{i,j} = 0$ is an optimal point. If $(\nabla_{\mathbf{W}} F)_{i,j} < 0$ then if $W_{i,j}$ increases the error will fall so a minimum has not yet been reached. In reality, and how we ran our experiments for this thesis, it is often the case that NMF algorithms are stopped before optimality conditions are reached due to time or computational constraints.

Most NMF algorithms follow the type of structure in Algorithm 1 (this is adapted from [42]). The stopping criteria here means either that the optimality conditions for all terms are met or a time or iteration limit reached. Steps 3 and 4 update $\mathbf{W}$ and $\mathbf{H}$ independently whilst keeping the other constant. This is because updating one with the other fixed is a convex problem, whilst updating both simultaneously is a non-convex problem. There are a range of techniques for the update step used in NMF of which we will review a small number.

Multiplicative Updates

Algorithm 1 General NMF algorithm

Input: Data matrix $\mathbf{V}$, subspace size r
Output: $\mathbf{W}$ and $\mathbf{H}$
1: Initialise $\mathbf{W}^{(0)}$ and $\mathbf{H}^{(0)}$
2: **for** $t = 1, 2, ...$ (until stopping criteria reached) **do**
3: $\quad \mathbf{W}^{(t)} = \text{update}(\mathbf{V}, \mathbf{H}^{(t-1)}, \mathbf{W}^{(t-1)})$
4: $\quad \mathbf{H}^{(t)T} = \text{update}(\mathbf{V}, \mathbf{W}^{(t)T}, \mathbf{H}^{(t-1)T})$
5: **end for**

The multiplicative update (MU) was first proposed by Lee and Seung [72]. At each iteration $\mathbf{W}$ and $\mathbf{H}$ are modified by [42]:

$$\mathbf{W} \leftarrow \mathbf{W} \circ \frac{[\mathbf{V}\mathbf{H}^T]}{[\mathbf{W}\mathbf{H}\mathbf{H}^T]} = \mathbf{W} - \frac{[\mathbf{W}]}{[\mathbf{W}\mathbf{H}\mathbf{H}^T]} \circ \nabla_{\mathbf{W}} F \tag{2.5}$$

and

$$\mathbf{H} \leftarrow \mathbf{H} \circ \frac{[\mathbf{W}^T\mathbf{V}]}{[\mathbf{W}^T\mathbf{W}\mathbf{H}]} = \mathbf{H} - \frac{[\mathbf{H}]}{[\mathbf{W}^T\mathbf{W}\mathbf{H}]} \circ \nabla_{\mathbf{H}} F \tag{2.6}$$

where $\frac{[\,]}{[\,]}$ means component-wise division. Note that $\nabla_{\mathbf{W}} F = \mathbf{W}\mathbf{H}\mathbf{H}^T - \mathbf{V}\mathbf{H}^T$. When $\nabla_W F = 0$ then $[\mathbf{W}\mathbf{H}\mathbf{H}^T] = [\mathbf{V}\mathbf{H}^T]$ and the updated $\mathbf{W}$ is the same as the previous $\mathbf{W}$ because the optimality conditions are met, a minima has been reached. When $\nabla_{\mathbf{W}} F > 0$ then $\frac{[\mathbf{V}\mathbf{H}^T]}{[\mathbf{W}\mathbf{H}\mathbf{H}^T]} < 1$ because $\mathbf{W}\mathbf{H}\mathbf{H}^T > \mathbf{V}\mathbf{H}^T$ so $\mathbf{W}$ is reduced in size to decrease the objective function. When $\nabla_{\mathbf{W}} F < 0$ then $\frac{[\mathbf{V}\mathbf{H}^T]}{[\mathbf{W}\mathbf{H}\mathbf{H}^T]} > 1$ so the size of $\mathbf{W}$ is increased pushing the solution closer to the minimum. More importantly, Lee and Seung [73] proved that the multiplicative update is guaranteed to monotonically reduce the objective function. In Chapter 5 and Appendix C we discuss this proof in much greater detail when extending it to prove that our extended multiplicative update method for XNMF also monotonically reduces the objective function. The MU is a simple and easy to implement technique which is guaranteed to monotonically reduce the objective function, all of which contributes to its popularity. However, it can be slow to reach reasonable solutions hence the incentive for alternative methods to be developed [42, 44]. We also include the scaled gradient descent interpretation of the MU, shown on the right hand side of the equals sign in Equations 2.5 and 2.6, which we will use in Chapter 7.

Alternating Least Squares and Alternating Nonnegative Least Squares

The Alternating Least Squares (ALS) algorithm computes the optimal solution of an unconstrained problem, with no non-negativity constraints. It then projects the components of the resulting matrix ($\mathbf{W}$ or $\mathbf{H}$) into the non-negative orthant [42]. This technique is unlikely to converge and may just oscillate [42]. It may be useful for initialising the $\mathbf{W}$ and $\mathbf{H}$ matrices in the first few steps as it is fairly fast. Alternating Nonnegative Least Squares (ANLS) does the same as ALS but includes the non-negativity constraint,

this can be computationally more costly than ALS but each iteration does reduce the objective function [78, 42].

Hierarchical Alternating Least Squares

For hierarchical alternating least squares (HALS) the objective function optimisation problem is solved one column at a time, rather than for the whole matrix, using an exact coordinate descent method [41, 44]. Part of the explanation for the success of this method is that the columns of $\mathbf{W}$ (and correspondingly the rows of $\mathbf{H}$) tend to share few non-zero terms. If columns of $\mathbf{W}$ are very similar they are redundant and the r should probably be reduced. A change in one column should have little impact on another so they can be updated individually, if NMF is achieving the sparse, parts based representation desired. HALS can be a fast batch technique which has been shown to provide good solutions for problems including images of faces and sets of documents [42].

Incremental NMF

The techniques discussed above all use the whole data set in a batch approach which has two significant drawbacks: 1) they can be relatively slow especially on large datasets, 2) if new data-points are added to the sample the entire batch must be re-run, rather than being able to update individually. Incremental or on-line NMF updates the $\mathbf{W}$ and $\mathbf{H}$ matrices as new data points are added. Mairal et al. [90] provide a method for incremental NMF which they claim to be: faster, effective on large data sets and which converges (almost certainly) to a stationary point. Other work on online NMF has included the use of robust stochastic approximation [49] and multiplicative update rules which assume that new samples do not affect the coefficients of previous data-points [137].

2.5 Improvements to NMF

Improvements to NMF mainly fall into two categories: 1) improving speed of convergence; and 2) improving the quality of the factorisation. The value of the first point is obvious, especially with increasing sizes of datasets, but the second requires some further elaboration. The quality of the factorisation depends on whether: (a) the approximation between $\mathbf{WH}$ and $\mathbf{V}$ is reasonable, so an acceptable minimum is reached; (b) $\mathbf{W}$ and/or (depending on your requirements) $\mathbf{H}$ are sparse; (c) real features are extracted from the data.

Recall that in Algorithm 1, where an explanation of the general method of performing NMF was shown, there are only a few choices to be made about the algorithm before

starting the updates. The **W** and **H** matrices must be initialised, any additional constraints (such as levels of sparsity) decided upon, the update method chosen and the size of the subspace selected. In this section, methods of potentially improving these three pre-update parts of the algorithm are explored.

Initialising W and H

The initialisation of **W** and **H** impacts on both the speed and quality categories discussed above. Many popular methods, such as Hoyer's algorithm [57], initialise the matrices at random which may result in both a slow speed of convergence and also a poor quality factorisation. Gillis [42] discusses some potential methods for better initialisation including: clustering techniques (also see Gong and Nandi [47]); using the SVD; column subset selection. Using genetic algorithms has also been proposed as a potential option to improve the initialisation [112]. Other groups produce a geometric interpretation which enables them to initialise the factorised matrices [154]. In this thesis we have mostly initialised **W** and **H** at random, unless otherwise stated.

Sparseness Constraints

An important paper in NMF research as by Hoyer in 2004 [57] who showed that while some datasets show a significant degree of sparseness when NMF is applied others do not. For images, for example, the level of sparseness tends to depend on the orientation of the images. Without additional sparseness constraints NMF does not factorise the data into a parts based representation, negating the key part of the technique. In addition, in some circumstances it may be desirable to impose addition sparseness onto either of **W** or **H**. For example, if a clustering version of NMF is desired imposing a very high level of sparsity onto the columns of **H** can enable clustering to occur [31]. In Hoyer's algorithm an L_1 norm for **W** and/or **H** is set to the desired level, then a gradient descent algorithm is applied which projects the elements of the matrices into the constrained space. Adding sparseness constraints can enable NMF to be more effectively used in some applications. For example a version of sparse NMF was used to identify biological processes in micro-array data [77], while other work applied similar ideas to a range of image data [81]. The sparseness constraint also adds to the uniqueness of a solution as well as the parts based representation [143].

Selection of r

The number of new basis vectors that make up **W** is given by r. Recent reviews by Gillis [42], and Wang and Zhang [143] identify the selection of r as an active area of research which has not been properly explored. This topic will be explored in greater detail in Chapter 3, here some of the recent literature investigating selection of r is reviewed.

Gillis [42] points to three common methods to choose r: trial and error; estimation using the SVD; and use of expert insights. Wang and Zhang [143] also discuss the common use of trial and error. They point out that with more theoretical results on the non-negative rank of the matrices or the use of manifold learning techniques might enable better estimates or bounds on the value of r. The three methods discussed by Gillis will be reviewed and flaws identified in Chapter 3.

A recent paper by Ulfarsson and Solo [133] used Stein's unbiased risk estimator (SURE) to select the best value of r. SURE provides an estimate of the mean-squared error. This means that finding the minimum of the SURE is an estimate of the minimum error. Hence it enables the estimation of the parameters of interest, in this case the best value for r. They derive an equation based upon the MU from [72] and then conduct experiments using both synthetic and real data. They conclude that their method provides reasonable predictions of the best r value to use.

A different method by Kanagal and Sindhwani [63] utilises cross-validation to estimate r. They split $\mathbf{V}$ into smaller matrices which are then kept as "held in" or "held out" parts. We implemented these methods and demonstrated some weaknesses in them, the results are available in Appendix A.

2.6 NMF Extensions

In this section we explore the potential for NMF to be applied in a broader range of ways and to a wider class of problems.

Supervised NMF

NMF is an unsupervised learning technique, however there are some approaches that convert it into a supervised method. In a paper by Wang and Gao [141] they apply a max-min distance NMF. They add in two additions to the standard objective function: 1) a minimisation of distances betweens pairs of points in the same class; and 2) the maximisation of pairs of points in different classes. In this manner they utilise the known class labels to try and improve the ability of the NMF representation to discriminate between classes. A similar idea is to add the Fisher constraint which maximises the between-class scatter whilst minimising the within-class scatter [60]. Gupta and Xiao [51] apply a maximum margin to the NMF, effectively merging NMF with an SVM.

Weighted NMF

Weighted NMF introduces an additional weighting matrix, $\mathbf{Y} \in \mathbb{R}^{m \times n}$, such that Equation 2.1 becomes $\mathbf{Y} \circ \mathbf{V} \approx \mathbf{Y} \circ \mathbf{WH}$ [143]. An advantage of weighted NMF is that it allows more importance to be placed on some elements of the data than others, thereby biasing the representation so that there is a closer match between those higher weighted elements of $\mathbf{V}$ and $\mathbf{WH}$. This type of bias may be useful if some part of the data is considered more important, such as the central pixels of a picture. In addition, weighted NMF allowed Kanagal and Sindhwani [63] to apply their form of cross-validation.

Non-negative Tensor Factorisation NTF

In NMF all data is converted into a vector form before being merged together into the matrix, $\mathbf{V}$. This process of converting data into a vector may involve the loss of information as structure in the data could be highly relevant. For example, a grey-scale picture might be better kept in its two-dimensional form and put together with other pictures as a three dimensional tensor [81]. NMF is essentially a sub-field of non-negative tensor factorisation (NTF).

Non-linear NMF

NMF is a linear dimensionality reduction technique. In supervised learning the use of non-linear mapping has proved enormously useful for classification tasks, such as the use of support vector machines [21]. Applying a non-linear (or kernel) approach to NMF allows for a non-linear representation of data which may be able to extract more features from the data [156, 13]. Adding constraints into the NMF formulation can also enable NMF to work in a non-linear manner by incorporating information about the manifold of the data [15, 16, 48]

2.7 Applications of NMF

Most of this thesis is concerned with new algorithms and approaches to NMF. With the exception of the work in Chapter 6, the data-sets we use are mainly to demonstrate the value of our techniques. However, the reason for producing these new methods is to be able to usefully apply them to real world problems. We therefore want to provide some further motivation for the work in this thesis by considering some applications of NMF.

NMF is popular due to its ability to extract sparse and interpretable representations [143] and therefore its applications tend to be in areas where these two requirements are high. For example, Gillis [42] gives examples of the use of NMF in image processing [80, 152], text mining [120] and hyperspectral imaging [95]. Other applications include using NMF (and SVD) to model networks [91].

NMF has also proven popular in biological applications [30] where it has been used with multimodal data to note changes in ADHD sufferers [3], clustering of micro-array data [97], classifying brain tumours [134, 153], exploring protein-protein interactions [139, 103] and investigating brain networks in rats [144].

We have not covered applications of NMF in any depth or with real breadth in this section, the review by Wang and Zhang [143] provides many more examples. Our purpose is, instead, to demonstrate the uses that NMF has found in a wide variety of domains and to provide some motivation for why we have produced this thesis which attempts to provide improvements to NMF, which will hopefully find uses in applications like those referenced in this section.

Chapter 3

Rank Selection in Non-negative Matrix Factorization using Minimum Description Length

In this chapter we consider how to choose the subspace size of our factorised matrices when performing NMF. An effective choice should minimise the noise whilst extracting the key features. We propose a mechanism for selecting the subspace size by using a minimum description length (MDL) technique. We demonstrate that our method provides plausible estimates for real data as well as accurately predicting the known size of synthetic data.

3.1 Introduction

3.1.1 Rank Selection in Non-negative Matrix Factorization

As noted in Section 2.5, a recent review [42] identified three main methods for selection of the subspace size r: use of expert insight; trial and error; and use of singular value decomposition. Expert insights are invaluable but suffer for three main reasons: 1) there may be no expert capable of selecting a good choice of r; 2) the experts may select r incorrectly; 3) even if an expert is able to effectively select r then independent confirmation is useful to add weight to the expert opinion.

Trial and error in this context means trying different values of r and then manual selection of one that best fits the aim of the researcher for that particular application. This method suffers as it is hard to know what a "good" solution looks like. Trial and

error can be dangerous in that it allows researchers to tune their results in a manner which produces the solution best for their work, so that a "good" solution becomes the solution that confirms their hypothesis.

Singular value decomposition (SVD) is applied by selecting r when the values of the singular values becomes "small". The challenge is that unless there is a clear fall towards zero the choice of where the values become "small" is very difficult to make. It is also not clear that the optimal number of dimensions for the variance preserving SVD would be the same as for NMF.

There are several more involved methods that have been proposed for the selection of the rank. Examples include the use of cross-validation [63, 104] and the use of Stein's unbiased risk estimator [133]. Cross-validation, in particular, is a common technique across supervised learning for assessing the quality of a model. In NMF, an unsupervised model, cross-validation essentially requires the imputation of missing data. There are different techniques for achieving this and [63] showed that these different techniques can produce significantly different estimates of r or sometimes no estimate at all (see Appendix A for details).

There is also an approach to NMF by [10] using a Bayesian formulation which offers the benefit of selecting r whilst finding $\mathbf{W}$ and $\mathbf{H}$. They impose a prior belief that the rank should be small and from there find a solution which fulfils this prior but this requires domain expertise to determine the choice of a good prior. A MDL approach, such as ours, can be interpreted in a Bayesian manner with the message lengths replaced with probabilities [88]. Our aim is to offer our approach as an additional method to help to guide a choice of r for researchers using NMF, in particular when there is little or no domain knowledge available.

3.1.2 Approach and Contribution

Our approach is to utilise a minimum description length (MDL) technique to find the best trade-off between a low r which misses key features and a high r which models noise. We suggest a pair of methods for applying MDL to NMF to assess the best choice of r. Our algorithms allow for the estimation of the best value of r and can produce a range of graphs that can be used to analyse the quality of the estimation.

In the next section we will introduce the background and theory behind MDL, then propose our solution to find the minimum description length for use in selecting the subspace size. In Section 3.3 we apply our MDL technique to real and synthetic data demonstrating the validity of the technique. Finally, in Section 3.4 we discuss the results

we obtained and explain why we believe our technique is a useful addition to the NMF toolbox.

3.2 Minimum Description Length

3.2.1 Background and Theory

MDL is a method for selecting between models of varying complexity. At its core is the idea that the best model is one that compresses the data most effectively. As the best way of compressing the data would also involve the smallest transmission cost when sending an encoded message, compression of the data and transmitting the shortest message are essentially equivalent.

In the NMF case the message is the matrix $\mathbf{V}$ which is approximated using $\mathbf{WH}$. The model is simple when r is small and, consequently, $\mathbf{W}$ and $\mathbf{H}$ have fewer elements which are cheap to encode. However, with a small r the approximation $\mathbf{WH} \approx \mathbf{V}$ is likely to be poor, requiring an addition to the message to correct the poor approximation. The MDL principle is to choose the model that minimises the total message length [136, 8, 114]. By trading off between the complexity and accuracy of the model, we hope to find the level of complexity which minimises the transmission of noise whilst maximising the transmission of real features (or the information content of the data).

There needs to be pre-agreement between the message transmitter and receiver about the level of precision, $\delta\mathcal{D}$, that the data, $\mathcal{D}$, should be set to. The message must be communicated to this agreed precision. This means the message will consist of the model, $\mathcal{H}$, and corrections to the model to reproduce the original data matrix exactly. Therefore the message length, $L(\mathcal{D}, \mathcal{H})$, consists of two parts [88]:

$$L(\mathcal{D}, \mathcal{H}) = L(\mathcal{H}) + L(\mathcal{D}|\mathcal{H})$$

where $L(\mathcal{H})$ is the length of the hypothesis, or the complexity of the model, and $L(\mathcal{D}|\mathcal{H})$ encodes the accuracy of the model. More complex models will tend to have a larger $L(\mathcal{H})$ and a smaller $L(\mathcal{D}|\mathcal{H})$.

Two important points should be made here. First we are not interested in how to actually optimally encode the message, we are only interested in the message length itself. Secondly, for model selection we are only interested in the relative length of each message. Any additional pieces of information required in the message that are consistent across all the different values of r are irrelevant in MDL because they will

increase the total description length by a constant amount and make no difference to the location of the minimum. The only terms that matter are those that will be different across different values of r. In other words, we are not interested in the message itself, or the absolute cost of encoding the message, but in the relative cost of sending the message at different values of r.

3.2.2 Proposed MDL Algorithm

To assess an appropriate subspace size for NMF using MDL we must first specify the components of $L(\mathcal{H})$ and $L(\mathcal{D}|\mathcal{H})$. The encoded length of the hypothesis, or the complexity of the model, $L(\mathcal{H})$ is $L(\mathbf{W}) + L(\mathbf{H})$ where $L(\mathbf{W})$ and $L(\mathbf{H})$ are the length of messages required to encode the matrices $\mathbf{W}$ and $\mathbf{H}$ respectively. The $L(\mathcal{D}|\mathcal{H})$ term is the length of the correction required to ensure that $\mathbf{V}$ can be reproduced exactly (to pre-specified precision) and is the encoded length of the matrix of errors, $L(\mathbf{E})$, where $\mathbf{E} = \mathbf{V} - \mathbf{WH}$. Implementation of MDL then requires the estimation of the minimum length of code that would allow the three matrices $\mathbf{E}$, $\mathbf{W}$ and $\mathbf{H}$ to be encoded into a message. When r is small the matrices $\mathbf{W}$ and $\mathbf{H}$ are small and so cost relatively little to encode, but the error matrix, $\mathbf{E}$, is large and therefore expensive to encode. As we increase r the errors reduce so the cost of transmitting $\mathbf{E}$ falls, but the cost of transmitting the model, $\mathbf{W}$ and $\mathbf{H}$, increases. At some point there should be an r value at which the total length is minimised, this is then the minimum description length and gives us a choice for r.

The principle of MDL relies on the use of the best possible encoding of the data, that is the encoding with the lowest cost. An upper bound on potential encodings can be estimated by considering the information content of each element. In general, any value which occurs multiple times is cheap to encode. To understand why, assume that the values in the error matrix are Gaussian distributed, then many elements will fall into a range that is close to the mean which can be assigned a short code. Any element far from the mean will require a longer code and is therefore more expensive to send. The Shannon information content allows us to estimate this cost using probabilities and is defined as [88]:

$$h(x) = -\log_2 P(x)$$

where x is the value of an element and $P(x)$ the probability of that value occurring. The aim is to find the probability of a value occurring in the $\mathbf{W}$, $\mathbf{H}$ and $\mathbf{E}$ matrices then to convert that value to a cost using the Shannon information content.

To estimate the probabilities we separate the data into bins of width $\delta\mathcal{D}$, which is the precision of the data. This value should be assessed from the data itself. We then apply two methods to estimate the probability of a term occurring in that bin. The first is to use the frequency of terms in that bin, n_i, compared to the total number of terms, N, so that $P(x) = \frac{n_i}{N}$ where $P(x)$ is the probability of an element x to be in the i^{th} bin. There is a considerable problem with this method in that while we can estimate the probabilities of each element in each bin, and hence the bound on the cost of sending the data, we also would need to send a specification of the histograms themselves, essentially the starting and end points of the histograms, along with the code used for each histogram. The bin width could be assumed to be the precision. The encoding of starting and end points of the histogram are likely to be fairly similar across r and there should be fairly inexpensive methods of encoding which bins are assigned to which codes but it is not a trivial task to complete. It is, however, likely that the parameters of this histogram model will be dwarfed by the cost of encoding the data itself. In the rest of this report we will refer to this technique as the histogram method.

The second method of estimating probabilities is more consistent with MDL principles but also suffers from a potential problem. Instead of using the frequencies of the histograms themselves, probability distributions are applied to the binned data, which allows us to find the probability density, ρ_i of each bin, i. The probability for an element, x, in the i^{th} bin is then $P(x) = (\rho_i \times \delta\mathcal{D})$. The advantage over the previous method is that the technique required to send the message is quite straightforward. As long as we use fairly simple distributions we must simply encode the parameters of the model and send them. The receiver can then recreate the distributions and will therefore be able to recreate the message. The only change in the model as r changes will be in the few parameters of the distribution (for a Gaussian distribution the mean and standard deviation, for example) which is highly unlikely to have any noticeable effect on the description lengths for any reasonably large data matrix. The potential problem with this method is that if the distributions do not fit well with the data, the estimates of the probabilities will not be accurate. This will then overestimate the description length.

The probability distributions to fit the non-zero terms from $\mathbf{W}$ and $\mathbf{H}$ should possess some features: it must be non-negative therefore the probability density should tend towards zero or infinity at zero values and the probability density should tend towards zero as the value becomes large. A simple choice is the gamma distribution which has a probability density function (PDF) of:

$$\rho(x) = \frac{\beta^\alpha}{\Gamma(\alpha)} x^{(\alpha-1)} e^{-\beta x}$$

where Γ is the gamma function, α and β are parameters. This is a flexible family of distributions that is often able to approximate real world distributions quite accurately.

At this point there is both a challenge and an opportunity. Part of the value of NMF is that it naturally tends to result in sparse matrices with a relatively high proportion of zero terms. The opportunity is that these zero terms could be sent very cheaply as separate matrices and the challenge is that these zero terms may result in highly inaccurate distributions being set to the $\mathbf{W}$ and $\mathbf{H}$ terms. The PDF of the gamma distribution either falls to zero or tends to infinity at zero depending on the parameter α. If the non-zero data is best fit by a distribution which tends to zero the estimates of the probabilities will be very poor. Most seriously they might well significantly overestimate the cost of sending the zero terms. It may be better to split the data up into zero-terms and non-zero terms. The separation of zero and non-zero terms requires some threshold to be set. Above the threshold the data is modelled using a gamma distribution and below the threshold the data is separately encoded, as described below. The Matlab code we provide allows for the manual choice of the threshold but also for an automatic choice made by applying MDL techniques themselves.

The automatic threshold is selected by systematically searching through the space of zero-thresholds for both $\mathbf{W}$ and $\mathbf{H}$ from zero up to the edge of the first bin. The total description length is then calculated and the lowest value is selected. This can result in different thresholds for $\mathbf{W}$ and $\mathbf{H}$ and also across different r terms.

The matrices containing the zero terms, $\mathbf{W}_0$ and $\mathbf{H}_0$, are encoded via the probability of being a zero which is given by n_0/n_T where n_0 is the number of zero values and n_T is the total number in $\mathbf{W}$ or $\mathbf{H}$. This leads to:

$$L(\mathbf{X}_0) = -n_0 \log_2 \frac{n_0}{n_T} - (n_T - n_0) \log_2 \left(\frac{n_T - n_0}{n_T} \right)$$

where $\mathbf{X}_0$ represents either $\mathbf{W}_0$ or $\mathbf{H}_0$ and the n terms are the numbers for $\mathbf{W}_0$ or $\mathbf{H}_0$ respectively. The second term encodes the cost of specifying the terms that are non-zero. This can be viewed as sending a code specifying a matrix of zeros and ones followed by the distribution and the codes for the non-zero terms.

This separation of the $\mathbf{W}$ and $\mathbf{H}$ matrices results in a total description length of:

$$L(\mathcal{D}, \mathcal{H}) = L(\mathbf{W}_0) + L(\mathbf{W}_+) + L(\mathbf{H}_0) + L(\mathbf{H}_+) + L(\mathbf{E}) \tag{3.1}$$

where $L(\mathbf{W}_0)$, $L(\mathbf{H}_0)$ are the description lengths required to encode the zeros in the $\mathbf{W}$ and $\mathbf{H}$ matrices respectively; $L(\mathbf{W}_+)$, $L(\mathbf{H}_+)$ are the description lengths required

to encode the non-zero terms in the $\mathbf{W}$ and $\mathbf{H}$ matrices respectively; and $L(\mathbf{E})$ is the description length to encode the error terms.

The non-zero data is assigned to bins of width $\delta\mathcal{D}$ and a gamma distribution is separately fitted to the $\mathbf{W}_+$ and $\mathbf{H}_+$ data. The probabilities, followed by the Shannon information content and hence the description lengths are then calculated. The $\mathbf{W}$ and $\mathbf{H}$ matrices that would be found from the message are calculated followed by the error matrix $\mathbf{E}$. A Gaussian probability distribution is set to the error matrix to enable the extraction of the probabilities and description lengths. The five terms that make up the description length are summed to give the total description length as in Eq. (3.1). Our general technique is explained in Algorithm 2 with a more detailed algorithm for the histogram and distribution method shown in Algorithms 3 and 4 respectively (see text for details of the notation).

Algorithm 2 MDL algorithm for each r value with automatic moving zero threshold

Input: $\mathbf{V}$, $\mathbf{W}$, $\mathbf{H}$, δD
Output: Description lengths for each r
1: **for** zero threshold values of $\mathbf{W}$ and $\mathbf{H}$
2: Separate out zero values, calculate $L(\mathbf{W}_0)$ and $L(\mathbf{H}_0)$
3: Fit gamma distributions to $\mathbf{W}_+$ and $\mathbf{H}_+$, calculate $L(\mathbf{W}_+)$ and $L(\mathbf{H}_+)$
4: Calculate $\mathbf{E}$ then $L(\mathbf{E})$
5: Calculate $L(\mathcal{D}, \mathcal{H})$
6: **if** $L(\mathcal{D}, \mathcal{H})$ is smaller than previous smallest, **then** store description lengths **endif**
7: **end for**
8: Return $L(\mathcal{D}, \mathcal{H})$

Algorithm 3 MDL algorithm for each r value for the histogram method with fixed zero threshold

Input: $\mathbf{V}$, $\mathbf{W}$, $\mathbf{H}$, δD, zero threshold
Output: Description lengths for each r
1: Calculate $\mathbf{E} = \mathbf{V} - \mathbf{WH}$
2: Separate elements of $\mathbf{E}$, $\mathbf{W}$ and $\mathbf{H}$ into bins of width δD
3: For elements below the zero threshold calculate: $L(\mathbf{X}_0) = -n_0 \log_2 \frac{n_0}{n_T} - (n_T - n_0) \log_2 \left(\frac{n_T - n_0}{n_T}\right)$
4: For elements above the zero threshold calculate: $L(\mathbf{X}_+) = -\sum_i^B \log_2 \frac{n_i}{N}$ (B is number of bins)
5: Calculate: $L(\mathcal{D}, \mathcal{H}) = L(\mathbf{W}_0) + L(\mathbf{W}_+) + L(\mathbf{H}_0) + L(\mathbf{H}_+) + L(\mathbf{E})$
6: Return $L(\mathcal{D}, \mathcal{H})$

3.3 Application of Minimum Description Length

To demonstrate the application of our MDL technique we have applied it to real and synthetic datasets. The results shown in this chapter utilise the NMF method of [57]

Algorithm 4 MDL algorithm for each r value for the distribution method with fixed zero threshold

Input: $\mathbf{V}$, $\mathbf{W}$, $\mathbf{H}$, δD, zero threshold
Output: Description lengths for each r
1: Calculate $\mathbf{E} = \mathbf{V} - \mathbf{WH}$
2: Separate elements of $\mathbf{E}$, $\mathbf{W}$ and $\mathbf{H}$ into bins of width δD
3: For elements below the zero threshold calculate: $L(\mathbf{X}_0) = -n_0 \log_2 \frac{n_0}{n_T} - (n_T - n_0) \log_2 \left(\frac{n_T - n_0}{n_T} \right)$
4: For elements above the zero threshold calculate probability density, ρ_i by fitting a gamma distribution to the binned data
5: Calculate description length: $L(\mathbf{X}_+) = -\sum_i \log_2 \rho_i \delta D$
6: Calculate total description length: $L(\mathcal{D}, \mathcal{H}) = L(\mathbf{W}_0) + L(\mathbf{W}_+) + L(\mathbf{H}_0) + L(\mathbf{H}_+) + L(\mathbf{E})$
7: Return $L(\mathcal{D}, \mathcal{H})$

without additional sparseness constraints included (but we also tested other methods taken from [42] and see no notable differences). It is important to emphasise that there is no ground truth in the real data we assess, so we cannot demonstrate beyond reasonable doubt that our technique works effectively. However, there are several criteria we would expect our method to meet if it is capable of selecting an appropriate r:

1. That the MDL technique performs in the manner we anticipate, i.e. that $L(\mathcal{D}|\mathcal{H})$ would fall and $L(\mathcal{H})$ should rise as r increases, and there should be a turning point in $L(\mathcal{D}, \mathcal{H})$.

2. That the MDL technique picks a plausible value of r for real data, especially if this is similar to choices made using other methods, such as use of external knowledge.

3. That the MDL technique can reasonably estimate r-values from synthetic data with a known r.

4. That MDL shows clear estimates of r for different types of data.

5. That the choice of r is robust to some variation in the data.

In Figure 3.1 we demonstrate the success of MDL in achieving the first and second points and part of the fifth. The left plot shows a set of 2429 real images of faces [1] (see Table 3.1) with 361 dimensions (pixels) used by [72]. The description lengths change exactly as we would expect: the length of the errors falls with increasing r, at the same time the $L(\mathbf{W})$ and $L(\mathbf{H})$ terms grow larger. The MDL algorithm produces the pattern that we would expect, fulfilling our first criterion. This same plot also demonstrates that MDL can meet the second criterion, the straight line down to $r = 80$ shows the r-value of the minimum description, but the turning point is fairly flat and a reasonable

choice could be anywhere from $r = 50$ to $r = 100$. Here we should note that when Lee and Seung [72] used this data-set they chose $r = 49$ for their subspace size. They do not specify how they chose r but it may well have been via a trial and error approach choosing that value when it gave good plots for their paper. Using MDL we have found a result in a similar range with no parameter tuning or assumptions beyond the choice of precision. This estimated value of r is certainly a sensible value and the turning point is clear. In our experiments the estimate of r does not change significantly with different choice of precision as long as the precision is chosen sensibly (based upon the precision of the data itself). We have also included results from re-running the NMF algorithm on the data, which show no difference in the choice of r and produce virtually identical lengths, in fact the differences in the results are difficult to spot in the figure. This identical solution to re-runs of the data is significant as NMF does not necessarily produce one unique solution. Instead all the re-runs of the algorithm are likely to have produced somewhat different $\mathbf{W}$ and $\mathbf{H}$ matrices. Our estimation of r does not change at all implying a level of consistency across different NMF solutions. A final point to be noted from this plot is that the solid line shows results from the distributions while the dashed line shows the results from using the histograms alone. There is no difference in estimation of r and only small differences in description length values between the two methods. As both methods have potential, but complementary, flaws, the similarity in output implies that these flaws do not adversely affect the conclusion.

TABLE 3.1: Data-set names, the type of data, the number of dimensions, m, and number of data-points, n.

Name	**Type**	m	n	**Source**
Faces [1]	Image	361	2429	http://cbcl.mit.edu/software-datasets /FaceData2.html
Genes [2]	Biological	5000	38	http://www.broadinstitute.org/cgi-bin /cancer/datasets.cgi
FTSE 100	Financial	1305	94	University of Southampton Bloomberg information terminal

The right plot in Figure 3.1 shows MDL applied to synthetic data with $m = 1000$ and $n = 2000$. This simple synthetic data is created by creating two matrices $\mathbf{W} \in \mathbb{R}^{m \times r}$ and $\mathbf{H} \in \mathbb{R}^{r \times n}$ with random locations of random non-zero terms. These are multiplied together and a small amount of additional Gaussian noise added (any elements that go below zero are set to zero). The size of the subspace r is 150 and is estimated correctly by the MDL approach. Clearly this data is simple and the selection of an appropriate r from here does not prove our approach is effective but it does show that MDL can find the appropriate value of r for some datasets. Again there is no difference in the

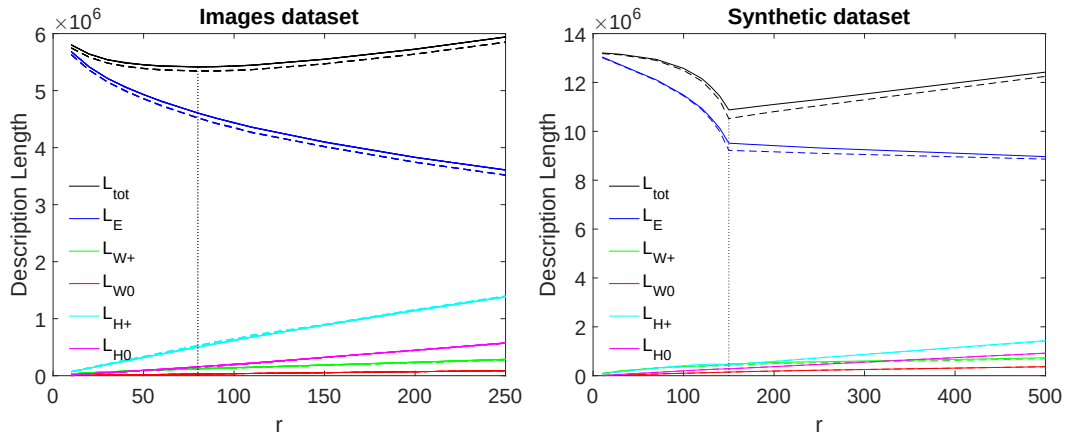


FIGURE 3.1: Left) The description lengths for the Faces dataset, showing a minimum of the total description length at around $r = 80$, with a reasonable range from $r = 50$ to $r = 100$. The solid line is for the description lengths found using the distributions and the dashed line from the histograms, the choice of r is the same. Right) The description lengths for synthetic data with a real r of 150. MDL shows a clear minimum at $r = 150$, perfectly estimating the correct value.

conclusions drawn from the histograms and the distributions. We thus claim that our MDL algorithms can fulfil the third criteria we set out.

The left plot in Figure 3.2 shows the total description lengths for several different data types (see Table 3.1) and allows us to meet our second and fourth criteria. There is no real ground-truth to these datasets so it is not possible to confirm that MDL is picking a good choice of r. It is, though, selecting an r that seems to be reasonable for each of the plots and also different from each other. If we were seeing all the turning points at similar values we might suspect that it was a feature of the algorithm rather than the data, the different locations of the turning points suggests that it is extracting information from the data itself. The Genes dataset has been extensively used, often with an implied r of 2 or 3 [30] which is similar to our estimate of between 2 and 5. Our estimate of the FTSE 100 dataset r-value is around $r = 8$, where the value of r could be considered as the number of economic sectors, such as energy, telecommunications, IT etc. An estimate of the number of these sectors of around ten would be reasonable, and is close to our evaluation.

The right plot of Figure 3.2 shows a range of results for synthetic data created as discussed earlier but with r values of 25, 50, 80, 120 and 150. The black vertical lines show the actual location of the real r value. For all results except for $r = 25$, where our prediction is marginally different, the MDL estimation is identical to the actual value. Our algorithm is correctly estimating the real value of r.

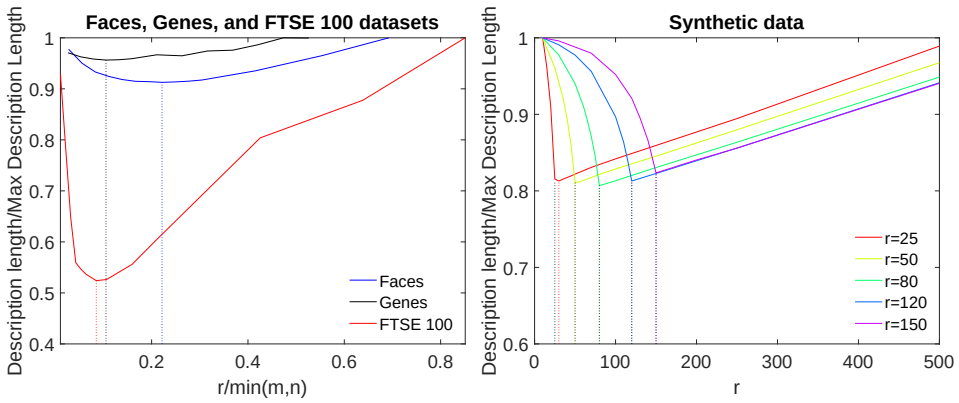


FIGURE 3.2: Left) The total description length for a range of datasets showing where turning points occur, potentially signalling an effective choice of r. While no ground truth is known the plots show minima at sensible locations, which correspond reasonably well with estimates from other sources. Right) The total description length for a range of synthetic data, MDL correctly identifies the r term for each dataset except for $r = 25$ which is still very close (the red vertical line) to the correct value (the black line).

The final aspect of our technique we will consider is the robustness of our technique to certain changes. We have already demonstrated that our technique is robust to re-runs of the NMF algorithm which can produce significantly different $\mathbf{W}$ and $\mathbf{H}$ matrices. To further attempt to get an impression of the uncertainty in our technique we applied bootstrapping (random sampling with replacement) to the faces dataset to produce five different variations of the original, in addition to the non-bootstrapped variant. NMF was then used to find the $\mathbf{W}$ and $\mathbf{H}$ matrices and our MDL technique applied. In the left plot of Figure 3.3 we see the results from applying the MDL techniques to this dataset. The solid line shows the results of applying MDL using the distributions for the images of faces dataset. The dotted lines show the same but for bootstrapped data, this is hard to see as the five plotted lines, one for each bootstrap, are almost identical. The dashed line shows the same but for MDL applied using the histograms and the dash-dot line for the equivalent bootstrapped results. The differences are marginal and the choice of r is similar for both. There is almost no difference seen in results when bootstrapping the data, we therefore consider the method to be reasonably robust to this type of alteration of the data.

The right hand plot of Figure 3.3 shows how the location of the MDL selection of r changes with the number of samples from $n = 500$ up to $n = 2429$ (the full dataset). The vertical lines record the value of the minimum for each sample size. It is apparent that the choice of r decreases with a smaller sample size. The final two terms with $n = 2000$ and $n = 2429$ are similar, but there is a considerable fall in the selected r-value with smaller n. We offer an explanation of why we see such a fall in the best choice of r consistent with these results. If we consider the data to be made up of features with

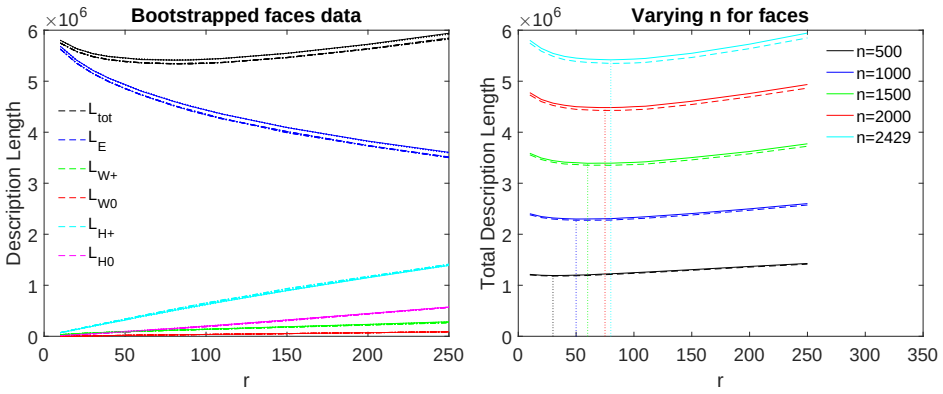


FIGURE 3.3: Left) The solid line shows the results of applying MDL using the distributions for the images of faces dataset. The dotted line shows the same but for bootstrapped data, these are hard to see as the results are almost identical. The dashed line shows the same but for MDL applied using the histograms and the dash-dot line for the equivalent bootstrapped results. Again there is almost no difference, our results show no significant variation under bootstrapping. Right) The results of L_{tot} for varying reduced size of n for the images of faces dataset. We see a clear reduction in the optimal r value as n is reduced. The dashed line is for the histogram plots and the solid line for the distributions.

a range of importance, in the case of a set of images of faces important features might be eyes, noses, ears or mouths. These features, and variants of them, will be required for almost all faces. On the other hand features such as moustaches are far less common. With lower numbers of samples it may be better to assume a moustache feature, which may be used by a small number of images, is not worth considering as a feature, instead a moustache can be considered as noise and accepted as part of the corrections made by the $\mathbf{E}$ matrix. As the number of samples increases it becomes possible to recognise that the extra feature is not noise and so the number of features expand, the capacity of the model increases with more data. We can, potentially, see the features that appear at low n as the more important features and as n increases we gain the features that are either less important to much of the data or important to only a small subset of the data. In reality this analysis of the data may be overly simplistic, in that the smaller number of features are likely to partially include the less important features, we may well then see combined features rather than the less relevant features being completely absent. Either way, as the number of data points increases so does the capacity of the model.

3.4 Conclusion

Our novel technique is to apply an MDL technique to selection of r. Before considering the results there are several attractive features of MDL. First, all the data is used in

MDL, there is no need to keep hold-out folds so no need to average out the results from the different folds or to consider the variance in the results when drawing your conclusions, as there is in techniques such as cross-validation. Second, MDL is an elegant technique with intuitive appeal which gives a natural trade-off between errors and model size. Third, the only potentially arbitrary parameter is precision, δD, but this has only a minor influence on the relative description length of different models and, in any case, may not be arbitrary if the precision of the data itself can be used.

We have applied our MDL technique to a range of real and synthetic data. MDL is able to accurately estimate r in synthetic data as well as providing reasonable estimates of the best r in real data. Our technique is robust to re-runs of the NMF algorithm and to bootstrapping the data, producing the same predictions of the best r.

Our technique has been tested on a range of data and is likely to work better on some data than others. If the distributions of the matrices $\mathbf{W}$, $\mathbf{H}$ and $\mathbf{E}$ match our set distributions well then we would expect to make good predictions. Conversely if the distributions do not match the data well our estimates may be inaccurate. In particular, if $\mathbf{V}$ is highly sparse, most of the errors will probably be zero and our algorithm may require some alteration. An advantage of our algorithms is that problems should be observed in differences in results from the histogram and gamma distribution methods. Our algorithms also allow for the production of a range of graphs to test the similarity of distributions to the actual data, which should highlight potential problems. Extensions to this work would likely be to investigate whether other distributions do a better job than our Gaussian and gamma choices.

There are a range of techniques for assessing an appropriate value of r in the literature. The best method is likely to utilise several techniques to select an appropriate r. We would suggest our technique adds to the potential toolbox that NMF researchers utilise to form judgements about the choice of r.

Chapter 4

Minimum Description Length as an Objective Function

When performing NMF, often the error between the actual and recreated matrices is used as an objective function, but this method may not produce the type of representation we desire as it allows for the complexity of the model to grow, constrained only by the size of the subspace and the non-negativity requirement. If additional constraints, such as sparsity, are imposed the question of parameter selection becomes critical. Instead of adding sparsity constraints in an ad-hoc manner we propose a novel objective function created by using the principle of minimum description length (MDL). Our formulation, MDL-NMF, automatically trades off between the complexity and accuracy of the model using a principled approach with little parameter selection or the need for domain expertise. We demonstrate our model works effectively on three heterogeneous data-sets and on a range of semi-synthetic data showing the broad applicability of our method.

4.1 Introduction

As discussed in Chapter 2, Hoyer [57] showed that for some data-sets standard NMF does not produce sparse solutions and it can be useful to impose sparsity explicitly. That paper introduced additional sparsity parameters through a projection operator which constrains the **W** and **H** matrices to have a defined level of sparseness. This projection operator operates after a gradient descent step and it projects the solution onto the constraint space so that it satisfies the desired L_1 norm (and also the L_2 constraint, but this is not relevant for the sparsity level). The formulation of Hoyer works well if there is a specific goal in mind. If you want to extract understandable features you might

want to impose significant levels of sparsity on **W** while if you are more interested in a representation closer to clustering, imposing sparsity constraints on **H** can be effective. These choices are domain dependent and require ad-hoc choices. However, if there is no prior knowledge or domain expertise, how do you decide the level of sparsity to impose on the factorized matrices and the distribution of the imposed sparsity between the two matrices? At a more fundamental level we ask whether the minimisation of the error alone is an optimal strategy.

With no additional constraints NMF algorithms attempt to minimise the error between the recreated and original matrices, or for other objective functions, such as the Kullbeck-Leibler divergence, to maximise the similarity in distribution between the actual and recreated matrices. In many situations we would expect this to result in over-fitting of the data - modelling noise rather than signal, and resulting in a model which would generalise poorly. By generalise poorly we mean that due to the model fitting noise (as well as structure) for the data it has seen, if the same model could be directly applied to other data it would do so poorly. For the data itself it means we are both creating a model which is more complex than it needs to be and one that may not be as interpretable as it could be. If we we to use the dimensionality reduced data as in input into, say, a supervised methods we would not expect it to perform worse than if it generalised better. The algorithm aims to maximise the similarity between the actual and recreated matrices with no regard for how complex the model being created is. As this is unsupervised learning the algorithm sees all the data and cannot be easily tested on unseen data.

There are two features of the formulation that contend with overfitting: 1) the small size of the subspace r, which means that it is likely real latent structure will be uncovered; 2) the non-negativity constraint which works to prevent overfitting by preventing the subtraction of structure, so providing an emphasis on sparse models which do not include unwanted structure. The choice of r was investigated in Chapter 3 [126] and its importance is clear when considering that if $r = m$ or $r = n$ then the model can perfectly recreate the actual data. The quality of the fit increases continuously with an increasing r. The non-negativity constraint can produce sparse solutions which act against overfitting but, without additional constaints, may not be sufficient as demonstrated by Hoyer [57]. The objective function without sparsity constraints may be biased in favour of improving accuracy with less consideration of the level of complexity of the model (which causes the overfitting). The imposition of sparsity constraints is an effective but ad-hoc and arbitrary method to impose an additional constraint on the model becoming too complex. A more fundamental approach is desirable.

A Bayesian approach to NMF has been demonstrated by, for example [119], which overcomes some of these related issues, especially the ad-hoc nature of additional sparsity constraints. Other groups have also applied a Bayesian formulation [10], including to the choice of r. Bayesian approaches have great value in NMF but they still require knowledge of the domain to enable prior distributions to be chosen. Other related work has been performed using particle filtering ideas [142].

We propose an alternative objective function using a formulation based on minimum description length (MDL). This formulation allows an automatic and natural trade-off between accuracy and complexity of the model without the need for considerable domain expertise. To our knowledge, no previous work has used MDL principles to guide the formation of the $\mathbf{W}$ and $\mathbf{H}$ matrices.

The remainder of this chapter is organised as follows. In Section 4.2 we present the principles behind our application of MDL to the NMF objective function; in Section 4.3 we explain our methodology and algorithm; in Section 4.4 we demonstrate the empirical effectiveness of our model on real and semi-synthetic data and finally in Section 4.5 we draw conclusions and propose future directions of research.

4.2 Minimum Description Length

Unlike in Chapter 3 where we used MDL to estimate the optimal subspace size here we want to replace the Frobenius norm objective function with an MDL version. To apply MDL to NMF, we regard the data matrix $\mathbf{V}$ as a message to be communicated and the factors, $\mathbf{W}$ and $\mathbf{H}$, as a reduced representation that can be more efficiently transmitted. The combined number of elements of $\mathbf{W}$ and $\mathbf{H}$ is usually much smaller than the number of elements in $\mathbf{V}$ ($m \times n \gg (m+n)r$) so should, usually, have a shorter required code length. In addition, we usually expect $\mathbf{W}$ and $\mathbf{H}$ to be fairly sparse compared to $\mathbf{V}$ which, again, makes them cheaper to encode, as will be discussed below. We need to reproduce the data matrix $\mathbf{V}$ to some pre-agreed precision so we also need a correction matrix, $\mathbf{E} = \mathbf{V} - \mathbf{WH}$ which is the accuracy of the model, with required code length $L(\mathcal{D}|\mathcal{H})$.

In this MDL formulation the objective function becomes

$$\begin{aligned} f(\mathbf{W},\mathbf{H}) =& L(\mathcal{D},\mathcal{H}) \\ =& L(\mathcal{H}) + L(\mathcal{D}|\mathcal{H}) \\ =& L(\mathbf{W}) + L(\mathbf{H}) + L(\mathbf{E}) \end{aligned} \tag{4.1}$$

where $L(\mathbf{W})$, $L(\mathbf{H})$ and $L(\mathbf{E})$ are the lengths of the code required for $\mathbf{W}$, $\mathbf{H}$ and $\mathbf{E}$ respectively. The terms, $L(\mathbf{W}) + L(\mathbf{H}) = L(\mathcal{H})$ can be viewed as a penalty term similar to Tikhonov [130] and lasso [129] regularisers in classic regression problems. Models of high complexity require correspondingly long codes. However, with a rise in model complexity we expect to see a model which can more accurately recreate the original data matrix and so would expect to see the value of the elements in $\mathbf{E}$ fall which should cause a reduction in $L(\mathbf{E})$. Therefore, an algorithm that finds a local minimum of Equation (4.1) should find a pair of matrices $\mathbf{W}$ and $\mathbf{H}$ which are automatically regulated to trade-off between complexity and accuracy.

In MDL there is no need to know how to encode the matrices, all we need to do is to estimate the length the code would need to be. As in Chapter 3 we do this using the Shannon information content [121] which is defined as $h(x) = -\log_2 P(x)$ where x is the value of an element in the relevant matrix and $P(x)$ is the probability of that value. Including this estimate of the length of the code gives us an objective function:

$$f(\mathbf{W},\mathbf{H}) = -\left[\sum_{i=1}^{m}\sum_{j=1}^{r}\log_2 P(W_{ij}) + \sum_{i=1}^{r}\sum_{j=1}^{n}\log_2 P(H_{ij})\right. \tag{4.2}$$

$$\left. + \sum_{i=1}^{m}\sum_{j=1}^{n}\log_2 P(E_{ij})\right] \tag{4.3}$$

where E_{ij}, W_{ij} and H_{ij} represent the i^{th}, j^{th} element of $\mathbf{E}$, $\mathbf{W}$ and $\mathbf{H}$ respectively. We have now specified the outline of our model (MDL-NMF) and why it should result in an automatic trade-off between complexity and accuracy. In the next section we will describe our method which estimates the probabilities and then finds appropriate $\mathbf{W}$ and $\mathbf{H}$ matrices.

4.3 Methods and Algorithm

We propose two methods of estimating the probabilities, in the same manner as in Chapter 3 [126]. The first is a non-parametric method of sorting the elements of the

matrix into bins and finding the probability by $P(x) = \frac{b_i}{N}$ where b is the number of elements in the i^{th} bin and N is the total number of elements of that matrix. The other, parametric, method applies probability distributions to the **E**, **W** and **H** matrices to extract the probability density of each value.

All the results in this paper were produced using the parametric model as it is computationally faster than the non-parametric approach because the objective function can be differentiated, allowing us to apply gradient descent methods to approach a minimum. Finding a fast non-parametric method might be an interesting future avenue of research.

To find the probability distribution we first put the elements of the **W**, **H** and **E** matrices, individually, into bins. We then fit a Gaussian distribution, with mean μ and standard deviation σ, to the **E** matrix and two different gamma distributions to the **W** (with parameters α and β) and **H** (with parameters a and b) matrices.

As the objective function of the distribution method is differentiable we seek a solution by following a gradient descent approach. Therefore we need to find $\frac{\partial f}{\partial W_{ij}}$ and $\frac{\partial f}{\partial H_{ij}}$ while noting that **E** is a function of **W** and **H**. As with most NMF techniques [42] we update **W** and **H** separately, making two separate convex functions to optimise as opposed to the original non-convex problem if we try and update both **W** and **H** together. We therefore apply:

$$W_{ij} \leftarrow W_{ij} - \lambda_W \frac{\partial f}{\partial W_{ij}}, \qquad H_{ij} \leftarrow H_{ij} - \lambda_H \frac{\partial f}{\partial H_{ij}} \tag{4.4}$$

where λ_W and λ_H are learning rate parameters,

$$\frac{\partial f}{\partial W_{ij}} = -\frac{[(\alpha-1)W_{ij}^{-1} - \beta]}{\ln(2)} - \frac{1}{\ln(2)\sigma^2}\sum_{k=1}^{n}[(E_{ik}-\mu)H_{jk}] \tag{4.5}$$

and

$$\frac{\partial f}{\partial H_{ij}} = -\frac{[(a-1)H_{ij}^{-1} - b]}{\ln(2)} - \frac{1}{\ln(2)\sigma^2}\sum_{k=1}^{m}[(E_{kj}-\mu)W_{ki}]\,. \tag{4.6}$$

The derivation of these equations is given in Appendix B. We can also consider the updates from a matrix multiplication perspective which can significantly speed up computational time. In this case we have:

$$\boldsymbol{W} \leftarrow \boldsymbol{W} - \lambda_W \nabla_{\boldsymbol{W}} f, \qquad \boldsymbol{H} \leftarrow \boldsymbol{H} - \lambda_H \nabla_{\boldsymbol{H}} f \tag{4.7}$$

where

$$\nabla_{\boldsymbol{W}} f = \widetilde{\boldsymbol{W}} + \widetilde{\boldsymbol{E}} \boldsymbol{H}^T \qquad \nabla_{\mathbf{H}} f = \widetilde{\boldsymbol{H}} + \boldsymbol{W}^T \widetilde{\boldsymbol{E}} \tag{4.8}$$

and we have defined three new matrices, $\widetilde{\boldsymbol{W}}$, $\widetilde{\boldsymbol{H}}$ and $\widetilde{\boldsymbol{E}}$, with elements:

$$\widetilde{W}_{ij} = -\Big(\frac{(\alpha-1)W_{ij}^{-1} - \beta}{\ln(2)}\Big), \widetilde{H}_{ij} = -\Big(\frac{(a-1)H_{ij}^{-1} - b}{\ln(2)}\Big), \tag{4.9}$$

$$\widetilde{E}_{ij} = \frac{1}{\ln(2)\sigma^2}(E_{ij} - \mu) \tag{4.10}$$

respectively.

As the **W** and **H** matrices are changed, the distributions will also shift, so we also need to change the parameters of the distributions in response. In Algorithm 5 we specify the algorithm for finding the **W** and **H** matrices for the MDL-NMF formulation. The precision term δ depends on the data and sets the bin width. We do not impose a specific stopping criteria - generally we run the algorithm until there appears to be no further fall in the objective function or it begins to rise.

Perhaps the most widely used NMF algorithm is the multiplicative updates (MU) method of Lee and Seung [72, 73] which is, effectively, gradient descent of the Frobenius norm error with an automatically changing learning rate which ensures that the objective function monotonically decreases. In contrast, due to the static learning rate, we have no guarantee that the MDL-NMF objective function will monotonically decrease. We compensate for this by reducing the learning rate if the objective function is not falling.

If different distributions were chosen the only change required would be in the $\nabla_{\boldsymbol{W}} f$ and $\nabla_{\boldsymbol{H}} f$ terms. These would need to be worked out again for the new distributions, otherwise the method and algorithm would remain the same.

Algorithm 5 MDL-NMF implementation

Input: Data matrix $\mathbf{V}$, subspace size r.
Output: $\mathbf{W}$, $\mathbf{H}$
Initialise: $\mathbf{W}^{(o)}$, $\mathbf{H}^{(o)}$.
Define: learning rate λ, precision δ
Calculate: parameters of the distribution, initial objective function f_0
for t=1,2,....stopping criteria reached
    if $W_{i,j} < \delta/2$ or $H_{i,j} < \delta/2$ set to $\delta/2$
    Update: $\mathbf{W} \leftarrow \mathbf{W} - \lambda \nabla_{\mathbf{W}} f_{t-1}$
    Update: $\mathbf{H} \leftarrow \mathbf{H} - \lambda \nabla_{\mathbf{H}} f_{t-1}$
    Calculate f_t
    if $f_t \geq f_{t-1}$ reduce λ, revert $\mathbf{W}$ and $\mathbf{H}$ to previous values
    Recalculate: parameters of distributions using maximum likelihood estimates.
end for

4.4 Results and Discussion

In the previous sections we have given an explanation of why our MDL-NMF method should work and the potential advantages over methods that minimise an error alone, or those that impose sparsity with ad-hoc parameter tuning. We now present empirical results to demonstrate the efficacy of our approach. In Table 4.1 we display the three data-sets we tested our MDL-NMF method on, including the type of data, the number of dimensions and number of data-points. These data-sets were chosen for their considerable heterogeneity to show the breadth of application of our method.

TABLE 4.1: Data-sets, the type of data, the number of dimensions, m, and number of data-points, n.

Name	**Type**	m	n	**Source**
Faces	Image	361	2429	http://cbcl.mit.edu/ software-datasets /FaceData2.html
Transcriptome	Biological	5000	38	http://www.broad institute.org/cgi-bin /cancer/datasets.cgi
FTSE 100	Financial	1305	94	University Bloomberg information terminal

The faces data is a set of 19-by-19 grey-scale images of faces which are individually converted to a 361 dimension vector then stacked up to make the $361 - \text{by} - 2429$ matrix. The transcriptome [46] data set is gene expression data for 38 samples, for people with two different types of leukemia. Each column of the FTSE 100 data represents the day closing price of one stock over a five year period, these are stacked with the other companies (94 rather than than 100 due to companies dropping into and out

of the FTSE 100 over time) from the FTSE 100 to make our data-set. These data-sets are from a wide range of applications, all have previously been studied using NMF techniques, [72, 29, 125], and provide evidence for the large range of effective use of MDL-NMF. They are from very different domains, with significantly different m/n ratios.

Making a direct comparison between dimensionality reduction techniques is a challenge. Unlike in supervised learning there is no direct model to test held-out data on, therefore we cannot easily check the generalisation error to compare our results to other methods. To attempt to compensate for this we have also tested MDL-NMF on semi-synthetic data.

The use of synthetic data has two main problems: 1) that it is difficult to know what the underlying structure should look like; 2) that synthetic data can be created and adjusted to provide data that produce favourable results for any particular technique instead of a realistic portrayal of the effectiveness of the method. Therefore, instead of using completely synthetic data we used a form of semi-synthetic data based upon the three data-sets in Table 4.1.

Our general principle to create semi-synthetic data is to perform NMF on the real data, producing $\mathbf{W}$ and $\mathbf{H}$. From there we know what the underlying structure of the semi-synthetic data is - assuming standard NMF uncovers real structure. We then rebuild the data, add noise and use our MDL-NMF technique to attempt to uncover the known true results.

We have not provided extensive results for how long the algorithm takes to run - if using the matrix multiplication method a full run from a randomised starting point to final solution for the largest and slowest dataset (faces) takes less than three minutes on an ordinary desktop computer. We would suggest the MDL-NMF method may be better used as a tuning method, with initial $\mathbf{W}$ and $\mathbf{H}$ matrices set as similar to the results from standard NMF, in which case MDL-NMF runs even faster. Either way MDL-NMF, while slower than standard NMF, takes an insignificant length of time to run on these data-sets. The scalability of this work to much larger datasets would require further research.

In the following subsections we demonstrate the potential value of MDL-NMF by investigating: 1) how well it reproduces the original data; 2) the process by which MDL-NMF learns the representation; 3) the ability of MDL-NMF to extract signal rather than noise from the semi-synthetic data.

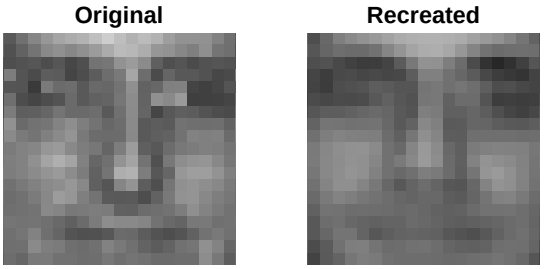
4.4.1 Recreation of the original data

Any NMF algorithm must be able to reproduce the original data with a reasonable degree of similarity. The precise level of similarity desired is not straightforward to specify because part of the purpose of NMF is to suppress noise in the data. In any real data set we do not know what is signal and what is noise. Therefore, if an NMF algorithm perfectly recreates a real world data-set we would not necessarily consider that a success, as it is likely to be modelling noise. On the other hand, if we are using data-sets with a reasonably high signal to noise ratio we would expect to be able to recreate much of the data using NMF techniques.

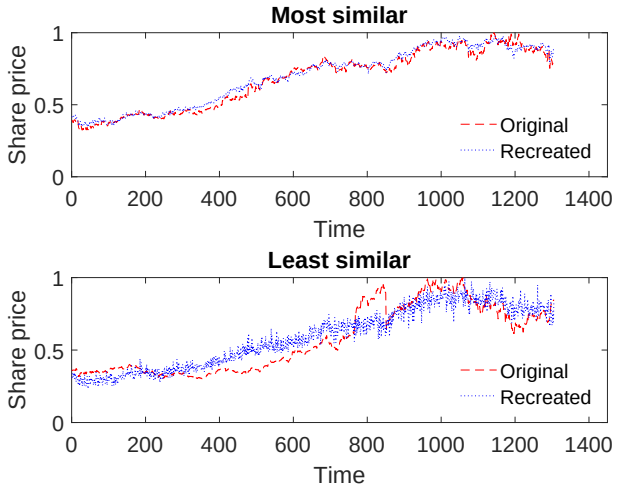
In Figure 4.1 examples of the final recreated output from each of the three data-sets are shown. In Figure 4.1(a) we show one randomly selected sample from the images of faces data-set. The left face is the original with the recreated face (for $r = 80$) on the right. In Figure 4.1(b) are plots of original (red dashed line) and recreated (blue dotted line) prices against time for one stock from the FTSE 100 dataset (with $r = 10$). The top and bottom plots show the most and least accurately recreated stocks respectively. The stock that is recreated most inaccurately, still produces a good reproduction of the actual prices. In Figure 4.1(c) for ease of visualisation of the genomic data, we randomly sampled 5000 of the elements of the data matrix $\mathbf{V}$ (x-axis) and compared the values of them to the same elements of the recreated matrix $\mathbf{WH}$ (y-axis) with $r = 4$. For all three data-sets our MDL-NMF method produces an effective recreation of the data.

4.4.2 The learning process

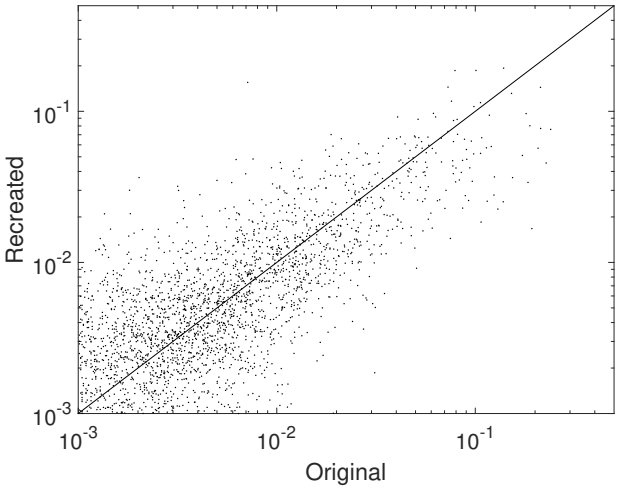
We now observe the learning dynamics by taking snapshots of the reconstruction during the learning process. Our MDL-NMF objective function is not purely based on minimising the error and therefore may follow a somewhat different trajectory to finding a minimum, it is therefore useful to demonstrate what does happen as we reduce the objective function. The analysis of learning dynamics can be useful to improve understanding which may lead to more effective learning algorithms or other aspects of these methods. For example learning dynamics are currently under investigation for deep neural network where recent work has investigated the dynamics of deep linear networks [118] and information bottlenecks in deep networks [131]. In Figure 4.2 we show how the reproduced data looks as the algorithm proceeds. Due to the differences between the data-sets they take a different number of iterations to arrive at a reasonable solution and the snapshots were chosen at appropriate iteration intervals to demonstrate how the quality of the fit improves. Figure 4.2(a) is an image from the faces dataset; Figure 4.2(b) is a FTSE 100 stock with the real (red dashed) and recreated (blue dotted)



(a)



(b)



(c)

FIGURE 4.1: Plots showing the quality of the final recreated data. (a) One randomly selected sample from the images of faces dataset. The left image is the original and the right image is the recreated face for $r = 80$. (b) Actual (red dashed line) and recreated (blue dotted line) prices against time for one stock for $r = 10$. The top is the stock with most similarity between actual and recreated, while the bottom plot is the least similar. (c) 5000 randomly selected elements of the data matrices with the recreated value plotted against original values for the transcriptome data with $r = 4$. MDL-NMF effectively recreates all three datasets.

prices changing with iterations; Figure 4.2(c) is 5000 of the elements of the recreated matrix versus the real values from the transcriptome data. All three plots show the clear trend from a randomised starting point to a reasonable final recreation of the original data.

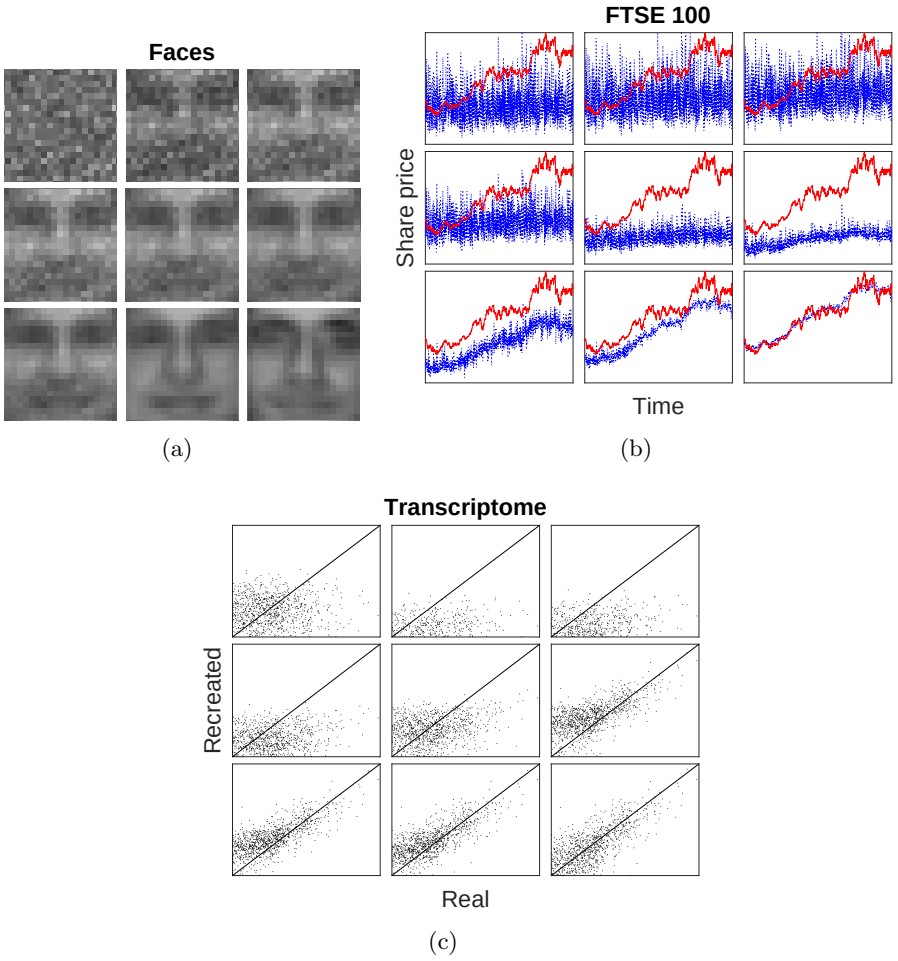
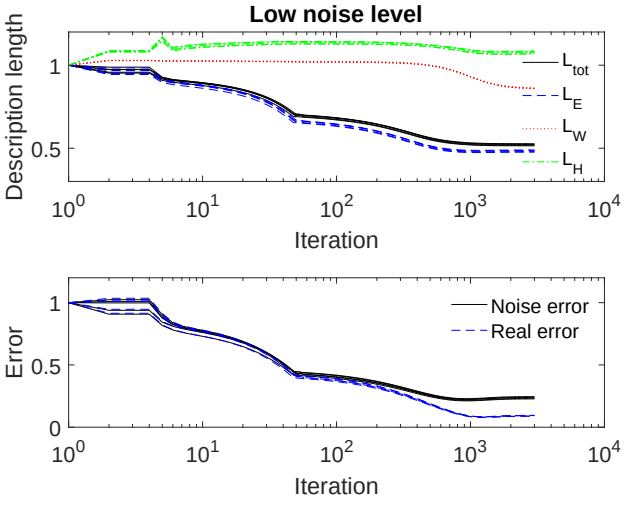


FIGURE 4.2: The reproduced data recorded at different iterations of the training process. (a) A sample from the faces dataset; (b) a randomly selected stock with the real (red dashed) and recreated (blue dotted) prices; (c) 5000 recreated results versus original values for the transcriptome data. All three show the change from a randomised starting point to a reasonable recreation of the original data.

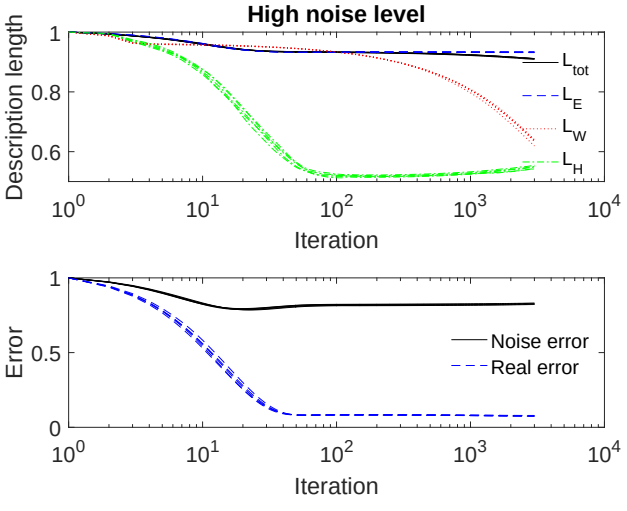
In Figure 4.3 we display the changes in both the description lengths and errors with iteration for a low and high noise variant of the semi-synthetic transcriptome data. There are five repeats of the MDL-NMF algorithm displayed on the same graph for the same data with different initialisations of the $\mathbf{W}$ and $\mathbf{H}$ matrices. For these plots we initialised the matrices for MDL-NMF by running standard NMF on the semi-synthetic data and then adding Gaussian noise to each element of the two matrices. The true error is defined as $\mathbf{E}_{\text{true}} = ||\mathbf{V}_{\text{true}} - \mathbf{WH}||^2_{\text{Fro}}$ where $\mathbf{V}_{\text{true}}$ is the underlying data matrix before we added noise while the noisy error is $\mathbf{E}_{\text{noise}} = ||\mathbf{V}_{\text{noise}} - \mathbf{WH}||^2_{\text{Fro}}$ where $\mathbf{V}_{\text{noise}}$

is the semi-synthetic data-set after noise is added. All the plots are normalised so the description lengths and errors all start at one.

There are some interesting features of the graphs. At a high noise variant the MDL-NMF significantly reduces the true error while the noisy error is not reduced much. At the low noise level most of the reduction in description length is achieved by reducing the noisy error - that also reduces the true error. With the high noise level the L_E terms does not reduce much, instead we get much larger relative reduction in the L_W and L_H values with commensurate significant falls in the true error. This may imply that the algorithm is reducing the complexity of the model (L_W and L_H) in preference to the error term (L_E) when there is a high level of noise. These type of plots may be useful analysis tools to use when investigating new data sets.



(a)



(b)

FIGURE 4.3: The changes in description lengths and errors with iteration for the semi-synthetic transcriptome data for a low noise variant and a high noise variant.

4.4.3 Representing true data over noise

A good NMF technique should be able to extract true underlying signal from noisy data. For our semi-synthetic data we can look at how well our MDL-NMF technique does with extracting the true signal from the noise we added.

In Figure 4.4(a) we show the true error against the noise error for MDL-NMF. Any point below the straight line shows a version where MDL-NMF finds a better error on the true data than the noise data. These results are from a range of variants of the semi-synthetic data: varying additional noise levels; varying r values used to create the semi-synthetic data; varying r values used to try and extract the true representations; true $\mathbf{W}$ and $\mathbf{H}$ matrices that are smoothed; using sparsity induced NMF (sNMF) to create the true matrices. The different symbols represent the three different types of semi-synthetic data. It is clear that almost all the results show a lower true error than noisy error MDL-NMF is able to cut through the added noise to represent the true data better than the noise data. There is no clear conclusion to draw here about whether MDL-NMF is performing better for certain methods or others.

In Figure 4.4b we show the same results but for standard NMF using multiplicative updates [72]. We note here that we originally made the data using a NMF method therefore we might expect NMF to effectively find the real data beneath the noise. Standard NMF effectively does for most of the data points but there are several points where it produces a better fit to the noisy data than the true data, in which case it is overfitting to the noise.

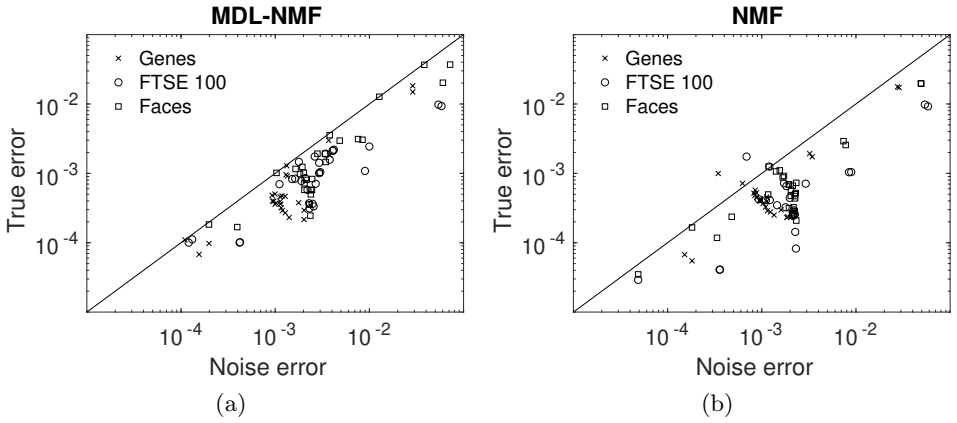


FIGURE 4.4: The real error against noise error show that MDL-NMF almost always finds a better fit to the real data than the noise added data. Results for a range of semi-synthetic data-set types are shown. (a) for MDL-NMF. (b) for NMF.

A direct comparison between NMF and MDL-NMF may not be completely fair, as NMF was used to create the semi-synthetic data in the first place. Also we are assuming that the original creation of the semi-synthetic data removed all the noise from the original

data - if it did not then there is still noise left in the true error. Accepting these limitations, if we see any improvement in representing real data using MDL-NMF over standard NMF then it implies that MDL-NMF can be a useful tool to use alongside NMF.

In Figure 4.5(a) we show real error against levels of noise for the three semi-synthetic datasets for both MDL-NMF and NMF. In Figure 4.5(b) we display the same but this time using sparsity induced NMF (sNMF) from Hoyer [57] with induced sparsity chosen to be the actual sparsity of the known semi-synthetic **W** and **H**. This favours the sNMF over MDl-NMF which has no direct information about the level of sparsity of the **W** and **H** matrices. We would therefore expect sNMF to produce significantly better results as the noise increases and it becomes increasingly more difficult to find the true data beneath the noise.

The comparison between MDL-NMF and NMF shows very little difference between the two methods with low level of noise, but as the noise level increases we get a marginally improved representation with MDL-NMF. This is true for all three data-sets although it is most apparent for the Faces. This finding is sensible, at low levels of additional noise there is little need to regularise but as the noise level increases we would expect to see increased levels of over-fitting. While there is a small improvement, it is not significant enough to provide strong evidence that MDL-NMF is doing a better job here that standard NMF. The comparison with sNMF is instructive on this, when we set the sparseness level to what the underlying matrices have, we get much less overfitting—the sNMF does much better than either MDL-NMF and NMF. So the sparseness is reducing much of the overfitting. MDL-NMF shows a promising ability to, at least a little, compensate for the increases in noise. We initialised the MDL-NMF matrices using the matrices produced by NMF with added noise. It is possible that with better initialisation choices we could do even better than the small improvements we show here.

4.5 Conclusions

The standard NMF objective function which minimizes the error may result in producing matrices which are too complex and overfit the data. We provide an alternative, based upon an MDL approach which provides an automatic and natural tradeoff between the accuracy and complexity of the representation.

We demonstrated the effectiveness of MDL-NMF on three heterogeneous data-sets, producing representations which match well with the real data. We also tested our model

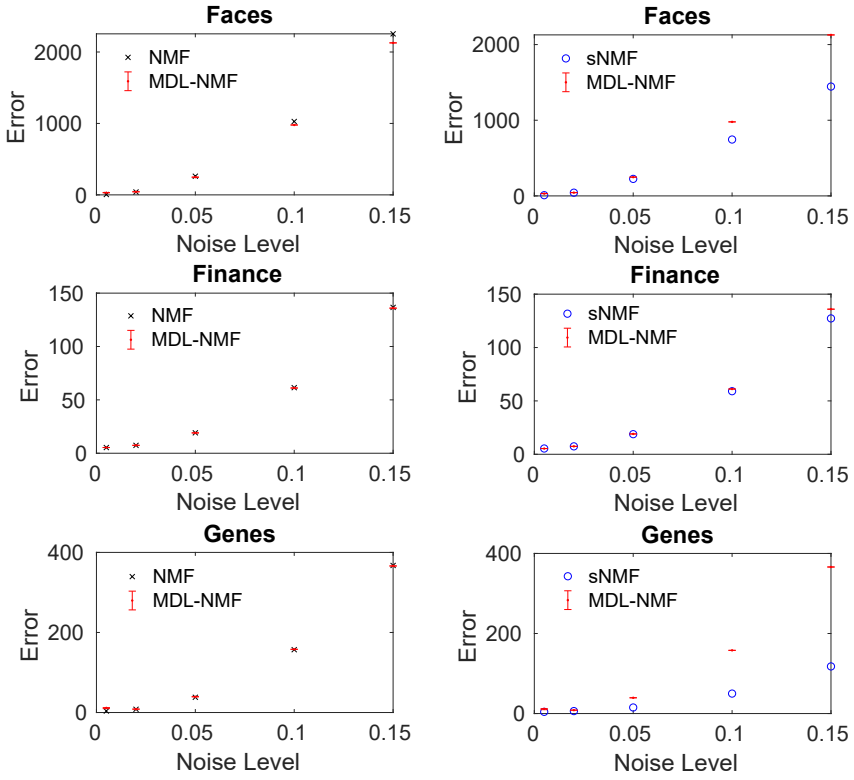


FIGURE 4.5: A comparison of the real error found for MDL-NMF against NMF and sNMF for the three semi-synthetic data sets.

on semi-synthetic data producing representations which represent the real data significantly more than the noise. In addition, studying the changes in the description lengths with iteration may be a useful analysis tool to use to investigate a dataset.

MDL-NMF produces superior results at extracting real data from noise over NMF for some semi-synthetic data especially when the added noise becomes high. It would be a worthwhile procedure to run MDL-NMF after a standard NMF run using the NMF matrices as initialisations for MDL-NMF to see how the two representations compare.

We have proposed using an MDL based objective function for NMF and demonstrated that it works effectively. However, the details are highly flexible. The choice of both distributions to model the matrices and the optimization scheme for finding the $\mathbf{W}$ and $\mathbf{H}$ matrices may both be subject to considerable improvement. There may also be significantly faster and more efficient methods of finding the $\mathbf{W}$ and $\mathbf{H}$ matrices in the MDL-NMF formulation than that proposed in this thesis. The choice of the learning rates λ_W and λ_H could potentially be improved as could the initialisation of the $\mathbf{W}$ and $\mathbf{H}$ matrices. Our aim is to introduce the method and background and demonstrate that it has potential to be a useful tool.

Chapter 5

Non-Negative Matrix Factorisation with Exogenous Inputs

In this chapter we produce a method of extending NMF called non-negative matrix factorisation with exogenous inputs (XNMF), which allows for the inclusion of known external data into the NMF formulation. We will introduce the motivation behind XNMF, derive a method of finding the factorised matrices, prove that it monotonically reduces the new XNMF objective function, and that it produces promising empirical results.

We demonstrate that XNMF is suitable for the analysis of multi-variate financial time series data in which the variation in data is explained by latent subspace factors and contributions from a set of observed macro-economic variables. On share prices from the FTSE 100 index time series, we show that the proposed model produces a lower reconstruction error than NMF and is effective in clustering stocks in similar trading sectors together via the latent representations learned. While we focus our attention here on financial data we believe there will be a range of potential applications - including biological which we explore in Chapter 6.

5.1 Introduction

Financial systems are inherently complex, driven by the objectives of market players, along with monetary and fiscal policies of governments. Pure time series analysis has

been applied extensively to asset returns [145, 127, 101], exchange rates [7] and derivatives [100, 96]. NMF has been applied to financial data in several ways, such as identifying underlying trends in stock market data [27]. Also variants of NMF applied to portfolio diversification have been used to minimise risk. For example, the trends that are produced by NMF (essentially the columns of $\mathbf{W}$) with the coefficients for each stock can be analysed to split the stocks into different groups [26, 140]. Factorizing multivariate asset return data into low rank factors can potentially discover low dimensional representations that are determined by sectors of assets that are likely to show similar responses. However, statistical signal analysis methods usually do not take into account exogenous information from macro-economic variables (referred to in this chapter as macro-variables) that have significant contributions to market movements.

We expect such factorizations to potentially uncover sector-specific drivers from among a wide range of macro-variables available. Specifically, our model represents the variation in any asset as consisting of contributions from sector-specific components and selected macro-variables. Hence the main novel contributions in this chapter are the specification of such a factorization model and a learning algorithm for it. We empirically demonstrate the effective performance of our approach on share price data from FTSE 100 companies and theoretically prove that the XNMF algorithm is guaranteed to monotonically reduce the objective function.

This chapter is structured as follows: in Section 5.2 we present our model including the underlying mathematics and the proof of monotonic reduction of the objective function; in Section 5.3 we discuss the real and synthetic data we used; in Section 5.4 we display our results; and in Section 5.5 we conclude and summarize our results.

5.2 Model and Learning Algorithm

Our aim, in general, is to find a combined representation of some internal data using a combination of the subspace and coefficients produced together with some fixed external data. In this chapter we consider financial data therefore the internal data is share price data and the external data are macro-economic variables (we call macro-variables here). However, while we focus on financial data this method is potentially applicable across many other domains.

In this chapter our input matrix, $\mathbf{V} \in \mathbb{R}^{m \times n}$ is a set of n stocks, with share prices (end of day prices) recorded for m time points (each point being one trading day). We can utilise standard NMF methods to find representations such that $\mathbf{V} \approx \mathbf{W}_1\mathbf{H}_1$ and, separately, $\mathbf{V} \approx \mathbf{W}_2\mathbf{H}_2$ where $\mathbf{W}_1 \in \mathbb{R}^{m \times r_1}$, $\mathbf{H}_1 \in \mathbb{R}^{r_1 \times n}$ and $\mathbf{H}_2 \in \mathbb{R}^{r_2 \times n}$ are all matrices to be

found. The macro-variables (our external data) are recorded in $\mathbf{W}_2 \in \mathbb{R}^{m \times r_2}$ and are fixed quantities, we will give specific examples of external data in the next section. Here r_1 is a parameter to select and r_2 is the number of macro-variables.

Performing NMF separately does not allow us to integrate the exogenous and internal data. We will now focus on how to formulate a combined representation and how to find solutions, but first we will briefly revisit the multiplicative updates introduced in Chapter 2 [72]. This method gives updates for $\mathbf{W}$ and $\mathbf{H}$ of

$$\mathbf{W} \leftarrow \mathbf{W} \odot \frac{\left[\mathbf{V}\mathbf{H}^T\right]}{\left[\mathbf{W}\mathbf{H}\mathbf{H}^T\right]}, \qquad \mathbf{H} \leftarrow \mathbf{H} \odot \frac{\left[\mathbf{W}^T\mathbf{V}\right]}{\left[\mathbf{W}^T\mathbf{W}\mathbf{H}\right]} \tag{5.1}$$

where $\odot$ is the Hadamard product and $\frac{[]}{[]}$ denotes element-wise division. These updates monotonically reduce the objective function $\frac{1}{2}||\mathbf{V} - \mathbf{W}\mathbf{H}||^2_{\text{Fro}}$. In our combined representation we want to find matrices $\mathbf{W}_1$, $\mathbf{H}_1$ and $\mathbf{H}_2$ that satisfy $\mathbf{V} \approx \mathbf{W}_1\mathbf{H}_1 + \mathbf{W}_2\mathbf{H}_2$ which requires us to minimise

$$f = \frac{1}{2}||\mathbf{V} - \mathbf{W}_1\mathbf{H}_1 - \mathbf{W}_2\mathbf{H}_2||^2_{\text{Fro}}. \tag{5.2}$$

This differs from normal NMF due to the additional matrices, we therefore need to derive an update rule to find solutions for the three matrices ($\mathbf{W}_2$ is kept constant). As multiplicative updates are effective, popular and theoretically sound [73] we aim to extend them to allow for the extra matrices. As minimising Equation (5.2) with respect to $\mathbf{W}_1$, $\mathbf{H}_1$ and $\mathbf{H}_2$ together is non-convex we hold two of the matrices constant whilst updating the third using multiplicative updates. Each individual problem is then convex, although the overall problem remains non-convex and there is no guarantee of reaching an optimal solution. Multiplicative updates are a type of scaled gradient descent therefore we need to find $\nabla_{\mathbf{W}_1} f$, $\nabla_{\mathbf{H}_1} f$ and $\nabla_{\mathbf{H}_2} f$. First we multiply out Equation (5.2) and get:

$$\begin{aligned} f =& \frac{1}{2}\text{tr}\left[(\mathbf{V} - \mathbf{W}_1\mathbf{H}_1 - \mathbf{W}_2\mathbf{H}_2)^T(\mathbf{V} - \mathbf{W}_1\mathbf{H}_1 - \mathbf{W}_2\mathbf{H}_2)\right] \\ =& \frac{1}{2}\text{tr}\Big[\mathbf{V}^T\mathbf{V} - \mathbf{V}^T\mathbf{W}_1\mathbf{H}_1 - \mathbf{V}^T\mathbf{W}_2\mathbf{H}_2 - \\ & \mathbf{H}_1^T\mathbf{W}_1^T\mathbf{V} + \mathbf{H}_1^T\mathbf{W}_1^T\mathbf{W}_1\mathbf{H}_1 + \mathbf{H}_1^T\mathbf{W}_1^T\mathbf{W}_2\mathbf{H}_2 - \\ & \mathbf{H}_2^T\mathbf{W}_2^T\mathbf{V} + \mathbf{H}_2^T\mathbf{W}_2^T\mathbf{W}_1\mathbf{H}_1 + \mathbf{H}_2^T\mathbf{W}_2^T\mathbf{W}_2\mathbf{H}_2\Big]. \end{aligned} \tag{5.3}$$

We then differentiate Equation 5.3 with respect to $\mathbf{W}_1$, $\mathbf{H}_1$ and $\mathbf{H}_2$ respectively to give three equations:

$$\nabla_{\mathbf{W}_1} f = (\mathbf{W}_1\mathbf{H}_1\mathbf{H}_1^T + \mathbf{W}_2\mathbf{H}_2\mathbf{H}_1^T - \mathbf{V}\mathbf{H}_1^T), \tag{5.4}$$

$$\nabla_{\mathbf{H}_1} f = (\mathbf{W}_1^T\mathbf{W}_1\mathbf{H}_1 + \mathbf{W}_1^T\mathbf{W}_2\mathbf{H}_2 - \mathbf{W}_1^T\mathbf{V}) \tag{5.5}$$

and

$$\nabla_{\mathbf{H}_2} f = (\mathbf{W}_2^T\mathbf{W}_2\mathbf{H}_2 + \mathbf{W}_2^T\mathbf{W}_1\mathbf{H}_1 - \mathbf{W}_2^T\mathbf{V}). \tag{5.6}$$

We apply multiplicative updates to $\mathbf{W}_1$, $\mathbf{H}_1$ and $\mathbf{H}_2$ by:

$$\mathbf{W}_1 \leftarrow \mathbf{W}_1 \odot \frac{\left[\mathbf{V}\mathbf{H}_1^T\right]}{\left[\mathbf{W}_1\mathbf{H}_1\mathbf{H}_1^T + \mathbf{W}_2\mathbf{H}_2\mathbf{H}_1^T\right]}, \tag{5.7}$$

$$\mathbf{H}_1 \leftarrow \mathbf{H}_1 \odot \frac{\left[\mathbf{W}_1^T\mathbf{V}\right]}{\left[\mathbf{W}_1^T\mathbf{W}_1\mathbf{H}_1 + \mathbf{W}_1^T\mathbf{W}_2\mathbf{H}_2\right]} \tag{5.8}$$

and

$$\mathbf{H}_2 \leftarrow \mathbf{H}_2 \odot \frac{\left[\mathbf{W}_2^T\mathbf{V}\right]}{\left[\mathbf{W}_2^T\mathbf{W}_2\mathbf{H}_2 + \mathbf{W}_2^T\mathbf{W}_1\mathbf{H}_1\right]} \tag{5.9}$$

where $\odot$ is the Hadamard product and $\frac{[]}{[]}$ indicates element-wise division. We will discuss how changes to $\mathbf{W}_1$ reduces the objective function noting that the same argument also applies to changes in $\mathbf{H}_1$ and $\mathbf{H}_2$. As we want to follow the gradient down towards a minimum, if $\nabla_{\mathbf{W}_1} f < 0$ then we want to increase $\mathbf{W}_1$. This is equivalent to $\mathbf{V}\mathbf{H}_1^T > \mathbf{W}_1\mathbf{H}_1\mathbf{H}_1^T + \mathbf{W}_2\mathbf{H}_2\mathbf{H}_1^T$, and, as shown in Equation (5.7), $\mathbf{W}_1$ is increased. Conversely if $\nabla_{\mathbf{W}_1} f > 0$ then we need $\mathbf{W}_1$ to decrease, which the multiplicative update does because $\mathbf{W}_1\mathbf{H}_1\mathbf{H}_1^T + \mathbf{W}_2\mathbf{H}_2\mathbf{H}_1^T > \mathbf{V}\mathbf{H}_1^T$. The final eventuality, that $\nabla_{\mathbf{W}_1} f = 0$, implies we have found a minimum of $\mathbf{W}_1$ and so want to keep $\mathbf{W}_1$ the same. Our multiplicative update multiplies $\mathbf{W}_1$ by one, fulfilling our requirement. We should note that if $\nabla_{\mathbf{W}_1} f = 0$ we are not necessarily at a minimum of the objective function as the other two matrices may still change which might change the situation of $\mathbf{W}_1$ such that $\nabla_{\mathbf{W}_1} f$ is no longer zero.

While this argument shows that the updates move in the correct direction, that is no guarantee of a monotonic reduction of the objective function as we could overshoot the minimum. However, part of the value of multiplicative updates is that Lee and Seung proved that they do produce a monotonic reduction [73].

We prove that our algorithm monotonically reduces Equation (5.2) by extending the proof of Lee and Seung [73] to cover the XNMF objective function using the same notation they did (see Appendix C for the full proof). Definition 1 and lemma 1 from their paper remain the same but we change the $K(h^t)$ diagonal matrix of lemma 2 to

$$K_{a,b}(h^t_{(1)}) = \delta_{a,b}(\mathbf{W}_1^T\mathbf{W}_1\mathbf{h}_1^t + \mathbf{W}_1^T\mathbf{W}_2\mathbf{h}_2)_a/h^t_{(1)a} \tag{5.10}$$

which changes only the $K(h^t_{(1)})$ term of $G(\mathbf{h}_1, \mathbf{h}_1^t)$. We then prove that $G(\mathbf{h}_1, \mathbf{h}_1^t)$ is an auxiliary function of the altered $F(\mathbf{h}_1)$:

$$F(\mathbf{h}_1) = \frac{1}{2}\left(\mathbf{v} - \mathbf{W}_1\mathbf{h}_1 - \mathbf{W}_2\mathbf{h}_2\right)^T\left(\mathbf{v} - \mathbf{W}_1\mathbf{h}_1 - \mathbf{W}_2\mathbf{h}_2\right) \tag{5.11}$$

which requires the proof that $M_{a,b}(\mathbf{h}_1^t) = h^t_{(1)a}(K(\mathbf{h}_1^t) - \mathbf{W}_1^T\mathbf{W}_1)_{a,b}h^t_{(1)b}$ is positive semidefinite:

$$\begin{aligned}
&\boldsymbol{\nu}^T\mathbf{M}\boldsymbol{\nu} \\
&= \sum_{a,b}\nu_a M_{a,b}\nu_b \\
&= \sum_{a,b}\left[h^t_{(1)a}\Big(\big(\mathbf{W}_1^T\mathbf{W}_1\mathbf{h}_1^t + \mathbf{W}_1\mathbf{W}_2\mathbf{h}_2\big)_a/h^t_{(1)a}\Big)_{a,b}h^t_{(1)b}\nu_a^2 - \nu_a h^t_{(1)a}(\mathbf{W}_1^T\mathbf{W}_1)_{a,b}h^t_{(1)b}\nu_b\right] \\
&= \sum_{a,b}\left[(\mathbf{W}_1^T\mathbf{W}_1)_{a,b}h^t_{(1)a}h^t_{(1)b}\nu_a^2 - \nu_a h^t_{(1)a}(\mathbf{W}_1^T\mathbf{W}_1)_{a,b}h^t_{(1)b}\nu_b + (\mathbf{W}_1^T\mathbf{W}_2)_{a,b}h^t_{(2)b}h^t_{(1)a}\nu_a^2\right] \\
&= \sum_{a,b}\left[(\mathbf{W}_1^T\mathbf{W}_1)_{a,b}h^t_{(1)a}h^t_{(1)b}\big(\frac{1}{2}\nu_a^2 + \frac{1}{2}\nu_b^2 - \nu_a\nu_b\big) + (\mathbf{W}_1^T\mathbf{W}_2)_{a,b}h^t_{(2)b}h^t_{(1)a}\nu_a^2\right] \\
&= \sum_{a,b}\left[(\mathbf{W}_1^T\mathbf{W}_1)_{a,b}h^t_{(1)a}h^t_{(1)b}\big(\nu_a - \nu_b\big)^2 + (\mathbf{W}_1^T\mathbf{W}_2)_{a,b}h^t_{(2)b}h^t_{(1)a}\nu_a^2\right] \\
&\geq 0.
\end{aligned} \tag{5.12}$$

Our proof is then the same as Lee and Seung except, due to the different $K(h^t)$, we end with:

$$h^{t+1}_{(1)a} = h^t_{(1)a}\frac{(\mathbf{W}_1^T\mathbf{v})_a}{(\mathbf{W}_1^T\mathbf{W}_1\mathbf{h}_1 + \mathbf{W}_1^T\mathbf{W}_2\mathbf{h}_2)_a}. \tag{5.13}$$

TABLE 5.1: Macro-variables used in this study

Macro-variable	**Frequency**	**Macro-variable**	**Frequency**
Gross Domestic Product	Quarterly	Unemployment	Monthly
Interest Rate	Monthly	Inflation Index Rate	Monthly
Imports Goods&Services	Quarterly	Exports	Monthly
Oil Imports	Monthly	Gross National Income	Quarterly
M1 Money Supply	Monthly	Productivity	Quarterly
GBP/USD	Daily	Contribution to CPI	Monthly
Balance of Payments	Monthly	Oil Investment	Daily
Government Gross Reserve	Monthly		

which proves that our algorithm will monotonically reduce the objective function for $\mathbf{H}_1$. Proofs for $\mathbf{W}_1$ and $\mathbf{H}_2$ are equivalent.

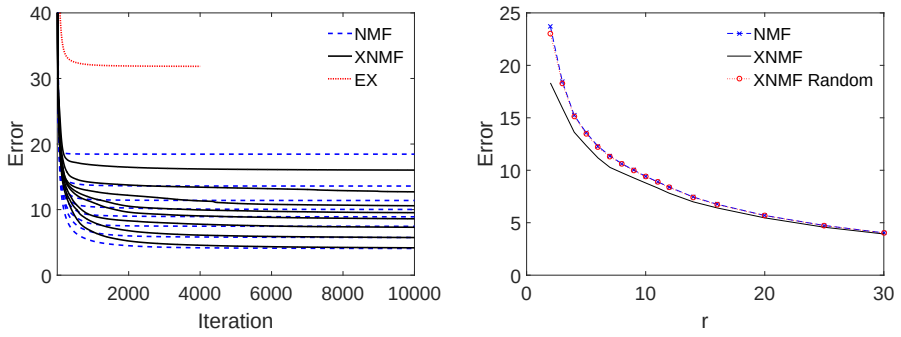
5.3 Data

We demonstrate the effectiveness of our model and learning algorithm empirically using daily data from FTSE 100 companies taken over a twenty year period. To deal with non-stationarity that may exist over such a long period in time, we also split the data into four equal sections in time and show results on all four separately.

In Table 5.1 we show the macro-variables used in this study. The choice of which macro-variables to use is somewhat arbitrary, there are many potential macro-variables, and they can be changed. To compensate for the differences in frequency between the share data (recorded on work days) and the macro-variable data we have linearly interpolated between all the macro-variable data so that the dimensionality (the number of time points) are equal. It is also the case that some of these macro-variables vary significantly while some (such as the interest rate) may not change at all during some time period. We will discuss some of these issues in the future work part of Chapter 9.

5.4 Results

We first confirm empirically that our algorithm achieves the desired goal, the reduction in the error until it reaches a minimum. In Figure 5.1(a) we show how the error changes with iteration for different values of r for the three different algorithms. We will use the same terminology throughout: NMF results are from the algorithm which minimised $||\mathbf{V} - \mathbf{W}_1\mathbf{H}_1||^2_{\text{Fro}}$, XNMF (exogenous inputs NMF) is for the minimisation of $||\mathbf{V} - \mathbf{W}_1\mathbf{H}_1 - \mathbf{W}_2\mathbf{H}_2||^2_{\text{Fro}}$ and EX (exogenous inputs alone) is for the minimisation of $||\mathbf{V} -$



(a) (b)

FIGURE 5.1: (a) The extended multiplicative update algorithm reduces the error monotonically with iteration until a plateau is reached. The multiple blue dashed (NMF) and solid black (XNMF) lines are for different sizes of the subspace, r. Generally the XNMF algorithm requires more iterations to approach a minimum than the NMF algorithm, but reaches a lower final error. (b) The final errors for different sizes of the subspace, r, for NMF (blue dashed lines with crosses), XNMF (solid black line) and XNMF using a $\mathbf{W}_2$ with random values (red dotted with circles). At all values of r that were implemented XNMF produces smaller errors than NMF or the randomised XNMF. As r is increased the difference between the errors produced by the algorithms reduces as the capacity of the NMF model increases and begins to overfit the data.

$\mathbf{W}_2\mathbf{H}_2||^2_{\text{Fro}}$. The blue dashed lines are for different values of r for NMF and the solid black lines for different values of r_1 for XNMF.

The EX algorithm (red dotted line) produces a poor approximation as it contains no information from the actual stocks themselves. The results of particular note are those of the XNMF algorithm which works as we expect it to, we see a fall in the objective function with iteration until it approaches a minimum where the error plateaus. The XNMF algorithm takes more iterations than the NMF algorithm to approach a minimum which might be expected as we have three matrices to optimise rather than two. In addition, the third matrix may make the objective function more non-convex than with just two matrices to optimise.

In Figure 5.1(b) we show the final errors from performing normal NMF (blue dashed line with crosses) and XNMF (solid black line) for different sizes of the subspace, r. At low values of r the model does not have enough subspace dimensions (columns of $\mathbf{W}_1$) to effectively fit the data and so the errors are high. The additional macro-variables here make a significant difference to the quality of the fit. As r increases the benefit of the additional information decreases as the increased capacity of the $\mathbf{W}_1\mathbf{H}_1$ part of the model means that a good fit to the data is possible without any additional information. As r increases it is likely that the model is overfitting the data, so any use of NMF requires a sensible choice of r to be made (see Chapter 3 [126]). We also include a version of XNMF (red dotted line with circles) called XNMF Random where the $\mathbf{W}_2$ matrix is composed of random numbers. The NMF and XNMF Random plots are hard

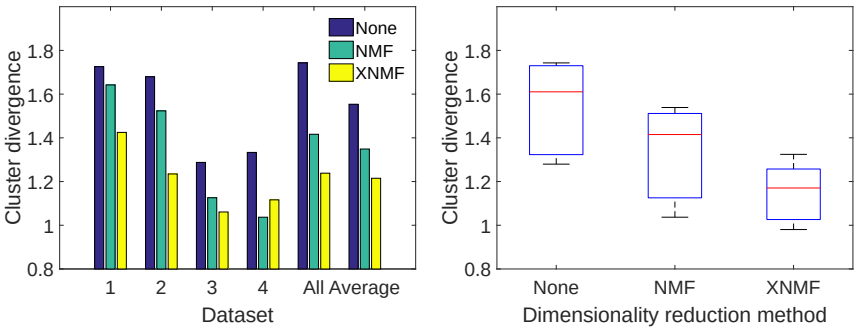


FIGURE 5.2: a) A representation of how much clusters diverge with time. K-means clustering was applied to non-dimensionality reduced data (dark blue bars), dimensional reduction using NMF (light blue bars) and dimensional reduction using XNMF (yellow bars) for four times periods and for a combination of the four periods. The clusters produced from data with no dimensional reduction diverge the most, with application of NMF the divergence is reduced and with XNMF we see the smallest divergence, the clusters tend to hold together better through time. b) Boxplots of the same results demonstrating the improvement of XNMF over NMF.

to distinguish demonstrating that the XNMF method is extracting real information from the external data, and not just reducing the error by increasing the number of elements to be optimised.

A particular appeal of NMF is noise suppression, by reducing the noise we might expect to be able to extract more real features from the data. A key result demonstrated with gene expression data is that the reduction in noise achieved by matrix factorization leads to stable clustering and biologically relevant inference about genes [12, 30]. In financial data we are often interested in how stocks and shares move together through time, a balanced portfolio would not contain lots of shares which are likely to fall in the same period. If we can effectively cluster the shares we can then build a more resilient portfolio.

We explored this idea of NMF producing improved clustering by exploring applications of the popular K-means clustering approach to the share price data. We are particularly interested in the quality of the clustering in the future, clusters that hold together better would be desirable. While NMF is not a clustering technique we can use the dimensionality reduction to create a new sub-space in which we apply clustering.

We performed K-means clustering on three versions of the data: a) no dimensionality reduction; b) dimensionality reduced using NMF; c) dimensionality reduced using XNMF. A measure of the similarity of a cluster is the average distance to the cluster centre using the non-dimensionality reduced data. For simplicity we used Euclidean distances. We are interested in the change in the average distance to the cluster centre as this gives us a measure of how similar the cluster is at different time points. In general, we

would expect an increase in distance as clusters will tend to diverge with time. If we see a smaller increase using the dimensionality reduced versions, it shows that the NMF techniques are allowing us to produce clusters which generalise better.

In Figure 5.2 we see the results of this forward prediction of clustering. First the data was split in half into a "training" set, the first half of the data in time, and a "testing" set, the second half of the data. The training data was then clustering into seven cluster centres using, respectively: the raw data, $\mathbf{V}$; $\mathbf{H}_1$ from NMF; and $\mathbf{H}_1$ from XNMF. We chose the size of the subspace, r, using a combination of domain knowledge about numbers of sectors in the data, and automatic techniques to assess subspace size [126]. The y-axis shows the ratio of the average distances from each data-point to its cluster centre between the testing data and the training data. A smaller value means the cluster stayed closer together. We see a clear trend, the raw data performs the worst whilst XNMF gives the best performance, and NMF gives a result in between the other two.

5.5 Conclusion

In this chapter we introduce a matrix factorization model suitable for multi-variate financial time series that includes known exogenous macro-variables. We use real FTSE 100 stock data to show that the multiplicative update factorization algorithm of XNMF produces lower errors than standard NMF and that stock clusters formed with the addition of exogenous data stay tighter bound through time. We also prove theoretically that the algorithm is guaranteed to monotonically reduce the objective function.

Chapter 6

XNMF for Combining Spatial Proteomics and Protein-Protein Interaction Networks

In this chapter we use NMF and XNMF as tools for the analysis of two types of biological data: spatial proteomics and protein-protein interaction (PPI) networks. In particular, we use XNMF to integrate these data-sets together to investigate if a combined representation can provide additional insight. The only work that we know of that explicitly attempts to integrate spatial proteomics with PPI network data was completed partially during this PhD (see Appendix D and [124]).

The structure of this chapter is as follows: in Section 6.1 we review some of the methods in the literature used to integrate biological data and introduce the motivation behind this study. In Section 6.2 we discuss the data-sets and the processes we used to enable XNMF to be successfully applied. In Section 6.3 we show and analyse our results. Finally in Section 6.4 we summarise and conclude.

6.1 Introduction

While NMF has been used extensively for biological applications, we do not intend to review this body of work in any detail. Instead we want to try and build up the motivation for why we are applying NMF and XNMF to this particular biological problem by discussing some examples of how NMF is used and why it is successful alongside the importance of these two data-sets and the value of integration of data-sets.

6.1.1 NMF in Biology

NMF has been successfully utilised to perform molecular pattern discovery [30] in part due to the improved interpretability over other methods and partly due to how well NMF deals with heterogeneous data [45]. Other groups have used NMF because it allows for good interpretation of spatial gene expression [149].

In this chapter we are especially interested in using XNMF to integrate two separate data-sets. An example of integration using NMF was performed by concatenating the input matrices together and making the assumption that one $\mathbf{W}$ matrix can be found that enables a joint representation [157]. In fact, this method of concatenating matrices and performing NMF is quite common [86, 159, 11, 138]. The downside of this technique is that it does not allow for addition of external data with known, and constant, subspaces we want to project onto while at the same time forming new subspaces.

NMF is also used in biology as a preprocessing step before the application of classification methods [153]. A version of NMF called Discriminant Non-Negative Matrix Factorization has also been used as a direct method to perform gene ranking [61] as has a semi-supervised version of NMF which was used to perform cancer classification [158].

6.1.2 Spatial proteomics

Biological cell are extremely complicated both in terms of structure and of function. In this chapter we are interested only in that cells can be considered to be made up of compartments. These compartments, some of which are called organelles, are physically separated from one another such that there are restrictions on what can pass into and out of these regions of the cell [39]. Spatial proteomics is concerned with the location of proteins within the cell, and specifically within the cellular compartments.

Proteins can only perform their function in direct physical contact with other proteins or parts of the cell, therefore knowing the location of a protein can aid in understanding its function. It has also been shown that there is a direct connection between diseases and subcellular protein localisation [108], consequently understanding certain diseases and cellular function depends on a reliable and accurate knowledge of protein localisation. The type of spatial proteomics data we are interested in is produced using various experimental techniques [38], the specific details of which are not of interest in this study. An important point, though, is that the spatial protoemics experiments cannot easily split the cells into particular compartments and then study the contents of that compartment, without considerable contamination [39]. Instead, the data that we are using is produced by splitting the cells into density fractions then measuring the abundance

of proteins in each fraction [39]. According to De Duve's principle [25] proteins from the same compartment will tend to have similar profiles across the density fractions. If we then know the spatial location of some proteins, from some other method, we can then match the proteins with unknown locations to their spatial compartment.

6.1.3 PPI networks

Protein-protein interaction (PPIs) networks record lists of known interactions between proteins. PPIs have been studied for many years due, in part, to their importance in understanding cellular function. Interactions between pairs or groups of proteins, or proteins and other parts of the cell, have significant consequences for cell functionality including links to disease [58]. PPI networks chart these known or predicted interactions.

NMF has been applied to PPI networks, including a Bayesian version of NMF used to cluster PPI networks [102, 103]. Other work has used forms of regularised NMF to predict protein function from PPI networks [148]

6.1.4 Integrating Biological Data

There is considerable interest in combining multiple sources of high throughput biological measurements. Examples include the integration of spatial and temporal patterns of gene expression [117], combining sequence and secondary structure of proteins [146], and the integrated analysis of the transcriptome and proteome [50].

Spatial proteomics and PPI networks should have significant similarities. A pair of proteins can only physically interact if they are in the same spatial location at the same time, hence we would expect that there would be a link between proteins that interact and those that share a spatial location. In principle, accurate PPI networks might be able to predict which proteins co-localise. Conversely, spatial proteomics cannot on its own specify whether an interaction exists, but if two proteins are in the same compartment it may be more likely due to the increased likelihood that they share a function. In addition, proteins that never exist in the same spatial location cannot directly interact.

There has been work conducted which uses the PPI networks to make predictions on protein localisation. In particular, PPI networks were used together with sequence predictors to classify proteins into spatial locations [33]. We have also performed work on integrating spatial proteomics and PPI network data which is included in Appendix D of this thesis [124]. In that work we demonstrated there was a correlation between spatial proteomics and PPI network data, we then proposed a method to combine these data-sets and demonstrated that we could use this integrated data-set to make predictions.

6.1.5 Motivation

The purpose of this study is to investigate whether we can usefully use XNMF to improve the analysis of biological data. While we focus on two types of data we hope this method will be applicable in a wider variety of domains.

The reason we believe XNMF could be useful in biology is that NMF has proven to have some advantages over methods like PCA especially in terms of interpretation - which could be highly useful in biology where interpretation may lead to greater understanding of the underlying science. In addition, NMF has demonstrated its value at integrating data [157] and integrating different types of data in biology is important as there are many biological processes involved in that highly complex environment. If NMF is a useful technique then we would expect XNMF to, potentially, provide even more useful analysis.

Spatial proteomics data is being produced in increasing quantities and quality but there is lack of methods and procedures to analyse this data [39] which is why we focus our attention particularly on that data-type. XNMF allows for inclusion of outside information into the spatial proteomics framework, potentially allowing integration of different data types using NMF. For the reasons outlined earlier, PPI networks seem a sensible choice of outside information due to their strong relationship with spatial proteomics [124]. We hope that by using XNMF in this way we can build improved representations of spatial proteomics data but also show that XNMF could be useful for a range of other biological data-sets.

6.2 Data and Methodology

6.2.1 Data-sets

In this study we use two data types: spatial proteomics and PPI networks. The spatial proteomics data we use is presented as abundances of each protein per density fraction and has some known labels, known as markers, which specify which organelle (compartment) a protein is believed to exist in (see Table 6.1).

The spatial proteomics used in this study is from Arabidopsis thaliana [34] which has 16 fractions and includes 689 proteins. There are two versions of the marker set, one has 27 markers across five organelles and we call it the “original marker” set. The authors then use the spatial proteomics data-set along with outside information to assign other proteins to the organelles resulting in a total of 142 markers in the set we call the “extended

	Fraction$_1$	**Fraction**$_2$	...	**Fraction**$_m$	**Marker**
Prot$_1$	q$_{1,1}$	q$_{1,2}$	...	q$_{1,n}$	Organelle$_1$
⋮	⋮	⋮	⋮	⋮	⋮
Prot$_i$	q$_{i,1}$	q$_{i,2}$	...	q$_{i,n}$	Unknown
⋮	⋮	⋮	⋮	⋮	⋮
Prot$_m$	q$_{m,1}$	q$_{m,2}$	...	q$_{m,n}$	Organelle$_k$

TABLE 6.1: The structure of a spatial proteomics dataset. Each of m-proteins has n dimensions of protein relative abundances, $q_{i,j}$, within each density fraction. If the protein subcellular location is known it is specified as a marker. This table is adapted from Gatto et al. [39].

Organelle	**Original Markers**	**Extended Markers**
ER	6	49
Golgi	5	27
Mit	8	26
PM	4	28
Vac	4	12
Unknown	662	547

TABLE 6.2: The number of original markers and extended markers for the five organelles from the Arabidopsis thaliana data-set used in this study [34].

marker" set. The organelles are: the endoplasmic reticulum (ER); mitochondrion (Mit); plasma membrane (PM); Golgi apparatus (Golgi); and the vacuole (Vac), but as we are not interested in biological considerations in this study we will just think of them as five classes, or clusters, of the data. The values for the original and extended markers sets for each of the organelles are presented in Table 6.2.

PPI network data is provided as pairs of proteins with known (or scores of) interactions. There are various public repositories of PPI network data, we use STRING [59]. The STRING data provides scores for the confidence in the interaction. We extract all the interactions for proteins in our spatial proteomics data-set with all the interactions known to any other proteins in our data-set and convert it into a $m \times m$ adjacency matrix, $\mathbf{V}_{\text{PPI}}$, with a 1 representing a known interaction and 0 if no known interaction exists. We could have used the confidence values themselves or used some form of threshold but to make it as simple as possible we considered that any evidence of a interaction between two proteins would produce a 1 in our matrix. The 689 Arabidopsis proteins have a total of 237,016 PPIs to all proteins in the PPI network. There are then 16,256 links between the 689 proteins that make up our spatial proteomics data-set.

The number of proteins in the PPI network data-set is 24,283 compared to only 689 proteins in the spatial proteomics data. It is therefore interesting to consider how the spatial proteomics proteins fit compared to the PPI network proteins - for example we might be interested in whether the spatial proteomics proteins have more or fewer PPI

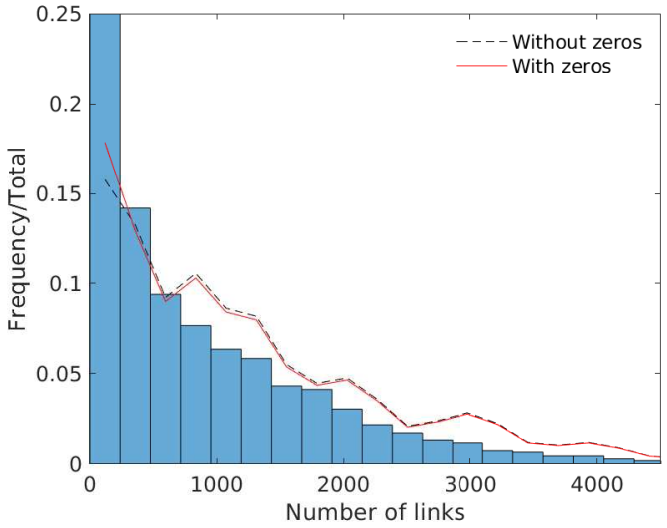


FIGURE 6.1: PPI distributions of number of links. The histogram is for the entire PPI dataset of 24,283 proteins. The black line is the distribution of the spatial proteomics proteins, if all the proteins with no links are removed. The red line is for all 689 proteins in the spatial proteomics dataset.

links compared the the average. This would give us some sense of whether the spatial proteomics proteins are typical or somewhat abnormal, and might give us clues as to how much information we can extract by merging these two data-set. There are 17 spatial proteomics proteins which do not appear in the STRING data-set at all. In Figure 6.1 we show the distributions of the number of links of the entire PPI dataset (the histogram), the distribution of spatial proteomics proteins where we have removed the 17 proteins without any links (black dashed line) and all the spatial proteomics proteins (red solid line). The spatial proteomics proteins tend to have significantly more links that the general distribution of the proteins from the PPI network data. We might expect this to be the case - the 689 proteins will all have shown up strongly in the spatial proteomics data, it is therefore likely that they are highly abundant in general and might then be expected to have a considerable number of known interactions.

6.2.2 XNMF Method for Integrating Spatial Proteomics and PPI network data

XNMF-Biol

For XNMF for spatial proteomics and PPI network (we denote as XNMF-Biol) data we consider the input matrix $\mathbf{V}$ to be the protein abundances per fraction. We define the dimensions to represent the proteins and the samples to be the fractions produced in spatial proteomics experiments. We then have $\mathbf{V} \in \mathbb{R}^{m \times n}$ where m is the number of proteins and n the number of fractions. The matrices to be found by XNMF are $\mathbf{W}_1 \in$

$\mathbb{R}^{m\times r_1}$, $\mathbf{H}_1 \in \mathbb{R}^{r_1\times n}$, $\mathbf{H}_2 \in \mathbb{R}^{r_2\times n}$ and the set matrix of exogenous data is $\mathbf{W}_2 \in \mathbb{R}^{m\times r_2}$. We are then looking to find the three matrices that produce $\mathbf{V} \approx \mathbf{W}_1\mathbf{H}_1 + \mathbf{W}_2\mathbf{H}_2$. We note here that, in comparison to the work in Chapter 5, we could interpret the $\mathbf{V}$ matrix as being transposed. It might be considered more natural to consider the proteins to be equivalent to the stocks and the fractions to be equivalent to time. However, as our external data has no relevance to the fractions our formulation works only if we consider the proteins to be the dimensions in this case.

The interpretation of each matrix needs to be considered. Each column of the $\mathbf{W}_2$ matrix can be viewed as a feature extracted from some outside data source and each element of the column as the weighting of that feature for each protein. The columns of $\mathbf{H}_2$ then are spatial proteomics fractions and each element of the column represents the importance of the relevant feature in $\mathbf{W}_2$ for that fraction. The columns of $\mathbf{W}_1$ can be interpreted as the axes in the protein space that we project onto with the $\mathbf{H}_1$ columns representing the coefficients of those directions for each spatial proteomic fraction.

Methods to generate $\mathbf{W}_2$

While there are many different types of potential external data that could be used, we focus on PPI networks to demonstrate the value of our method and because the relationship between the two data-sets has already been demonstrated [124]. There are many possible ways of producing the $\mathbf{W}_2$ matrix. We considered two main methods, one that extracts features using NMF and one that requires the manual choice of features.

Our first method (for shorthand in this chapter we call it XNMF1) uses standard NMF to reduce the dimensionality of the $\mathbf{V}_{\text{PPI}}$ (the matrix of 1s and 0s that represent whether interactions occur or not) matrix down from m to r_2 and produce the $\mathbf{W}_{\text{PPI}}$ matrix from $\mathbf{V}_{\text{PPI}} \approx \mathbf{W}_{\text{PPI}}\mathbf{H}_{\text{PPI}}$. There might be considerable room for further exploration of how to use NMF to produce a good choice of $\mathbf{W}_{\text{PPI}}$ beyond this method.

For the second method (called XNMF2) we use hand-crafted features to produce the $\mathbf{W}_2$ matrix. As the features are to come from the PPI network a sensible choice would depend on the number of links. Our first feature, column of $\mathbf{W}_2$, is then the total number of links each protein has to other proteins in the dataset, which we normalise by dividing by the number of links that the most connected protein has. The rest of the features (columns of $\mathbf{W}_2$) are the number of links of that protein to the known markers in each of the recorded organelles, normalised by dividing by the number of markers in each organelle.

While we want to focus mainly on these two methods we can also combine the automatic and hand-crafted features into a third method (XNMF3). We do this by concatenating the $\mathbf{W}_2$ matrices produced from the two methods together. There are clearly many other

different methods that could be used to produce features to fill the $\mathbf{W}_2$ columns. Our aim is to demonstrate the potential benefits of using our method with a small number of methods of producing $\mathbf{W}_2$.

Subspace size selection

The optimal choice of subspace size r_1 depends on the aims of the researcher. The higher the dimensions the more separate features will be extracted but more noise will be modelled as well. A method of choosing an optimal r_1 was introduced using minimum description length in Chapter 3 [126]. This subspace size may not be the most useful in terms of visualisations or extracting representations of interest but it is useful to investigate what would be considered the best choice in terms of data compression. For XNMF1 we include $L(\mathbf{W}_2)+L(\mathbf{H}_2)$ into the description length and do a search over the grid space of r_1 and r_2. This addition of the $L(\mathbf{W}_2)$ is only used to guide the choice of the value of r_2 when performing NMF on the PPI network data, once we apply XNMF to the combination of PPI and spatial proteomics datasets this value becomes fixed. For XNMF2 with manual features selected we need to add $L(\mathbf{H}_2)$ to the formulation.

Sparseness

Sparseness is a useful feature for several reasons but perhaps the most useful in this case is for interpretation of the results. We quantify levels of sparsity using the sparseness measure [57]: $\text{sparseness}(\mathbf{x}) = \frac{\sqrt{m}-(\sum|x_i|)/\sqrt{\sum x_i^2}}{\sqrt{m}-1}$ where $\mathbf{x}$ here is a column of $\mathbf{W}_2$ and m the dimensionality. This sparseness measure is between 0 and 1, with 1 meaning that only one element of $\mathbf{x}$ is non-zero (highly sparse) and 0 meaning all elements are equal.

Clustering

A fundamental issue in clustering is that lack of any certainty over what a good cluster actually means. For example, in our case in this chapter, clusters could consist of proteins with different numbers of known PPI interations. Another sensible set of clusters might be those in similar spatial locations - so those in the ER would cluster together, for example. There are methods to encourage the type of clustering that one is interested in [150] and the use of these methods combined with our XNMF might be an interesting piece of future work. We would expect the XNMF method with PPI network data as the external data to produce clusters which do not group the proteins into spatial locations as well as just using spatial proteomics data because the extra information could encourage other forms of clustering. However, these different clusters with the inclusion of PPI data might tell us something interesting especially if there are just relatively small changes in clusters when adding in the PPI data. The proteins that show some movement might well be considered to be those that exist in multiple compartments, for example.

There is also evidence that NMF can produce more stable clustering than some other methods such as using k-means clustering on raw data or on PCA data [65]. It would be interesting therefore to look at the stability of clustering when using: raw data, PCA, NMF and XNMF. To try and make the comparison as fair as possible we use k-means clustering on all the data. To measure the stability of clustering we use the Rand index [111] which measures the similarity between two different clusterings.

The Rand index, in our case, is measured between two different clusterings. We consider applying k-means clustering twice on the same data-set (say the raw spatial proteomics data) with the same number of cluster centres in both. Let $\mathbf{X}$ be one set of clustering and $\mathbf{Y}$ be another. We are then interested in calculating four values from $\mathbf{X}$ and $\mathbf{Y}$. All the values we consider are pairs of proteins compared. a is the number of pairs of proteins that are in the same cluster in $\mathbf{X}$ and also in the same cluster in $\mathbf{Y}$. b is the number of pairs of proteins that are in different clusters in $\mathbf{X}$ and also in different clusters in $\mathbf{Y}$. The values $a+b$ gives the number of pairs of proteins where the two sets of clusterings agree. c is the number of pairs of proteins that are in the same cluster in $\mathbf{X}$ but in different clusters in $\mathbf{Y}$ and d is the number of pairs of proteins in different clusters in $\mathbf{X}$ and the same cluster in $\mathbf{Y}$. So $c+d$ represents the number of pairs of proteins that are differently clustered in the two different clusterings. The Rand index is then given by:

$$R = \frac{a+b}{a+b+c+d} \tag{6.1}$$

where if $R = 1$ then the two clusterings are completely consistent and in $R = 0$ then they are completely inconsistent. We will use the Rand index in two ways: 1) to compare the consistency of clustering formed using the different dimensionality reduction methods and 2) to compare the clusterings with the organelle classes.

6.3 Results and Discussion

We are investigating whether the use of NMF and XNMF can improve the study of spatial proteomics data so we focus on two of the main requirements of data analysis for spatial proteomics [39]: quality of representation and of classification. However, before we look at the representation and classification we want to consider how important the external information is when conducting XNMF and how to choose the parameters r_1 and r_2.

6.3.1 Analysis of the Data and Techniques

We can consider XNMF to be performing $\mathbf{V} \approx \mathbf{X}_1 + \mathbf{X}_2$ where $\mathbf{X}_1 = \mathbf{W}_1\mathbf{H}_1$ and $\mathbf{X}_2 = \mathbf{W}_2\mathbf{H}_2$. The ratio $\frac{\text{mean}(X_2)}{\text{mean}(X_1)}$ gives us some idea of the importance of the external information to the representation formed. For $r_1 \approx r_2$ and $r_1 > r_2$ we would expect to see a value less than one because $\mathbf{W}_1$ can vary so should always be able to fit the data better than the constant $\mathbf{W}_2$ matrix. It is valuable to pay attention to this measure because it gives us some idea of how important the external data is, if the ratio falls very low then it is likely that the external data does not hold much value, conversely if, in future work, a type of external data was used with a high ratio it would imply that the type of external data could provide a lot of information. However, it may also be the case that with a high ratio the information provided by the external data might just be replacing information that could have been provided without the external data.

In the left plot of Figure 6.2 we show the $\frac{\text{mean}(X_2)}{\text{mean}(X_1)}$ against r_1 for both XNMF1 and XNMF2, we use $r_2 = 6$ for XNMF1 so it is directly comparable to XNMF2. We have shifted the position on the graph of each point slightly to the left and right so that it is easier to see the differences. As r_1 increases the importance of the external data decreases because the increasing r_1 can fit the data better - in fact it may be overfitting to the data and modelling noise. It is also interesting to note that XNMF1 and XNMF2 produce very similar results, it is not clear from this plot which we should favour. The right plot of Figure 6.2 displays the importance of the external data for a varying r_2 with $r_1 = 2, 3, 5$. As r_2 increases we see a clear increase in the importance of the $\mathbf{X}_2$ matrix to the overall reconstruction. Both of these effects are as we expect, but it is worth noting that the importance of the $\mathbf{X}_2$ reconstruction never becomes very high, in fact it does not reach even 20% of the total, even when $r_1 = 2$ and $r_2 = 12$. It may mean that we cannot extract a lot of information from the external data.

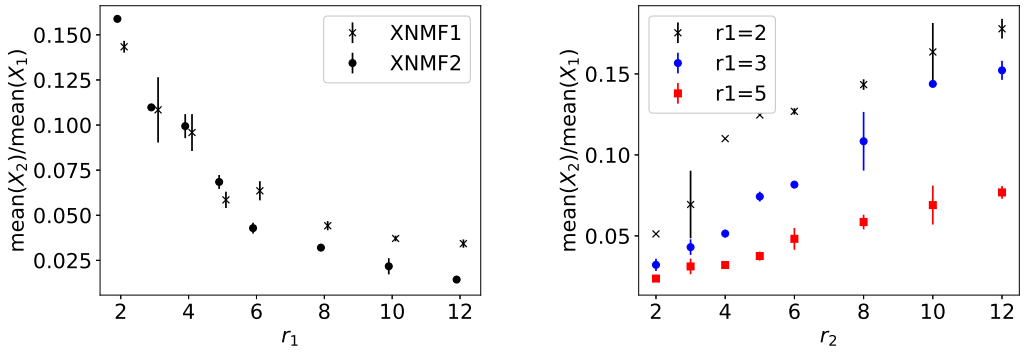


FIGURE 6.2: Left) Relative importance of the external data for XNMF1 (with $r_2 = 6$) and XNMF2 when r_1 increases. Right) Relative importance of the external data for the automatic features for increasing r_2 for three values of r_1. All points shown include three repeats with the standard deviation shown as an errorbar.

For standard versions of XNMF the only parameters that need to be selected are r_1 and r_2. The optimal choice of r_1 and r_2 can be chosen using the MDL method [126] and results for this are shown in Figure 6.3. The left plot shows the optimal choice of r_1 using the six manually chosen features (XNMF2). The right plot shows the optimal choice of r_1 for different choices of r_2. Both methods produce a choice of $r_1 = 2$ although $r_1 = 3$ produces a similar result. This implies we can maximise the compression of the data using an r_1 value of two or three. We emphasise here that this is the choice of r_1 which produces the best compression of the data - depending on the aims of the researcher there may be different requirements for the dimensionality reduction, such as picking out certain features, which may require different values of r_1. This does though imply that there is a low dimensional subspace on which the data sits and that projecting onto that subspace should result in retaining much of the information from the original data. This is interesting in that (presumably mostly for ease of visual inspection) a lot of spatial proteomics analysis uses a projection subspace size of two [39], which implies they are making a reasonable choice by projecting onto two dimensions. The right plot also implies that as r_2 increases the total description length also increases. For the aim of maximising the noise compression it may be better to use a very low r_2 or even no r_2 at all, however, noise compression is not the only aim when analysing the data. Including the external data may not be optimal from a MDL perspective but it might improve the representation.

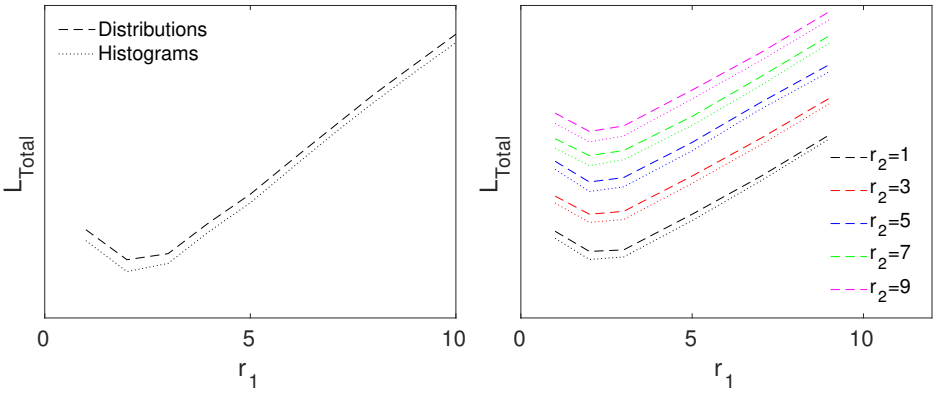


FIGURE 6.3: Left) The optimal choice of $r_1 = 2$ with the manual features (XNMF2) in $\mathbf{W}_2$ is found by minimising the description length. Right) The optimal choice of $r_1 = 2$ for the automatically chosen features (XNMF1), shown for the different values of r_2.

XNMF includes external data into the NMF formulation which should therefore produce a reconstructed matrix that is closer to the original. We would therefore expect to see a reduction in the final XNMF error $||\mathbf{V} - \mathbf{W}_1\mathbf{H}_1 - \mathbf{W}_2\mathbf{H}_2||^2_{\text{Fro}}$ compared to standard NMF. In the left plot of Figure 6.4 we show the final errors for NMF, XNMF1, XNMF2 and XNMFcontrol for increasing r_1. The XNMF version called XNMFcontrol uses a $\mathbf{W}_2$ matrix with values drawn from a uniform ditribution to demonstrate that the improvement in error is not due purely to the increase in number of matrix elements

that can be used to recreate the data. Both XNMF1 and XNMF2 perform better than standard NMF as well as the control XNMF (which produces very similar results to standard NMF). In the right plot of Figure 6.4 we show plots of how the error (taken in comparison to the error when $r_2 = 2$ to make the plot easier to interpret) falls for increasing r_2 when performing XNMF1. As r_2 increases we see an improved error but for higher levels of r_1 the change is small - which is what we expect. At high values of r_1 the model will overfit to noise so external data will provide no significant benefit in terms of error reduction.

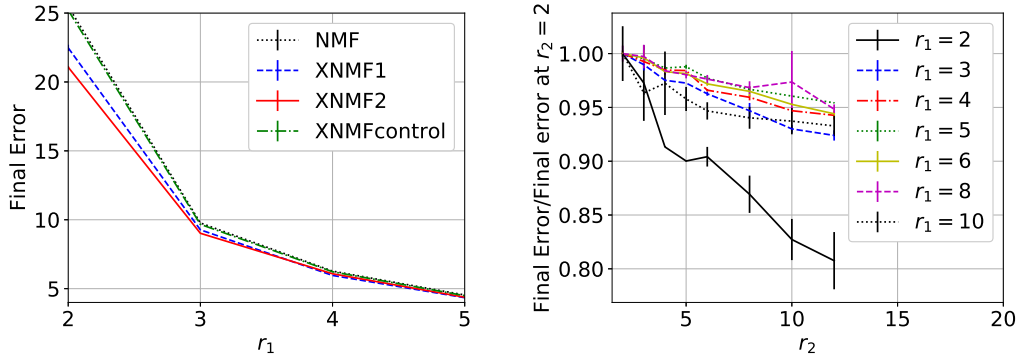


FIGURE 6.4: a) Final errors for NMF, XNMF1, XNMF2 and XNMFcontrol against increasing r_1, demonstrating that XNMF produces a reduction in error compared to standard NMF. b) Final errors (as a fraction of the error at $r_2 = 2$) against r_2 which demonstrates the increased value of the external data as the dimensionality of r_2 is increased.

We might hope that the outside information would help with producing a better representation of the data when the data becomes increasingly corrupted. This we can test by adding increasing amounts of Gaussian distributed noise to the original data matrix. In Figure 6.5a we show the true error against noise level for all the different methods and in Figure 6.5b we show a zoomed in version of the same plot. There is a small decrease in ability to recreate the true result (as opposed to the noise added data). NMF actually gets the closest to the true result, although the result is minor. This may be a feature of marginal overfitting when using XNMF - the features of the $\mathbf{W}_2$ matrix are produced by using the corrupted data so the slight increase in error is, perhaps, to be expected. But it does not suggest that XNMF provides any advantage over standard NMF when trying to extract clean data from noisy data, at least for these particular data-sets and methods.

6.3.2 Representation of the data

The interpretation of the four matrices, $\mathbf{W}_1$, $\mathbf{W}_2$, $\mathbf{H}_1$ and $\mathbf{H}_2$, was discussed in the previous section. The most interesting matrix here is $\mathbf{W}_1$ as it is the representation of

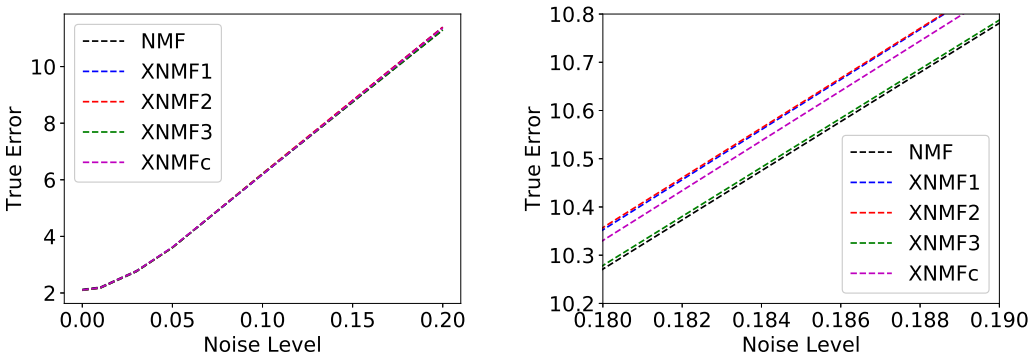


FIGURE 6.5: The effect of added noise on the different methods. NMF produces a very marginally better result that the XNMF versions (lower is better in this plot). The right plot is a zoomed in version of the left plot.

the proteins and should have the most biological interest.

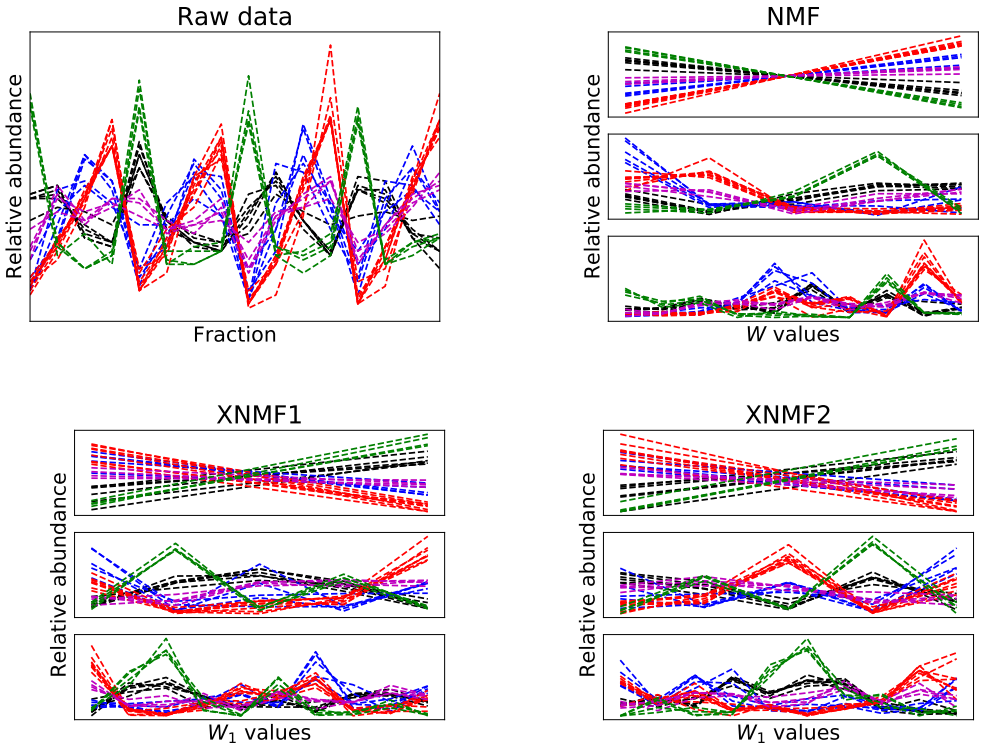


FIGURE 6.6: Top left) The fractional abundances for the raw data for the original marker proteins. Other three plots) The rows of $\mathbf{W}_1$ for the original markers with $r_1 = 2, 5, 10$ for NMF, XNMF1 and XNMF2. At low r_1 values interpretation is much easier than for higher numbers of dimensions.

We will consider two way of looking at the $\mathbf{W}_1$ matrix. The first is as a dimensionality reduced version of the spatial proteomics abundances which we show in Figure 6.6. Here we are looking at the individual proteins and their weights in the $\mathbf{W}_1$ space, so each drawn line is a row of $\mathbf{W}_1$. The top left plot shows the raw data for the original marker

proteins with different colours representing proteins from different organelles. The top right plot show dimensionality reduced representations of the organelle markers using NMF with r values of $r = 2, 5, 10$ (from top to bottom). The bottom two plots are equivalents for XNMF1 (with $r_2 = 6$) and XNMF2. It is of particular note how well separated out the different marker groups are when $r = 2$ for NMF. In fact from these two dimensions alone we can see that all the marker proteins can be separated into their specific organelles. This tallies with our estimate of the optimal subspace size of $r = 2$ from using MDL [126]. The dimensionality reduced version is significantly easier to interpret and analyse than the original raw data. The effect of the external data in XNMF1 and XNMF2 appears to be to blur the separation between the different markers.

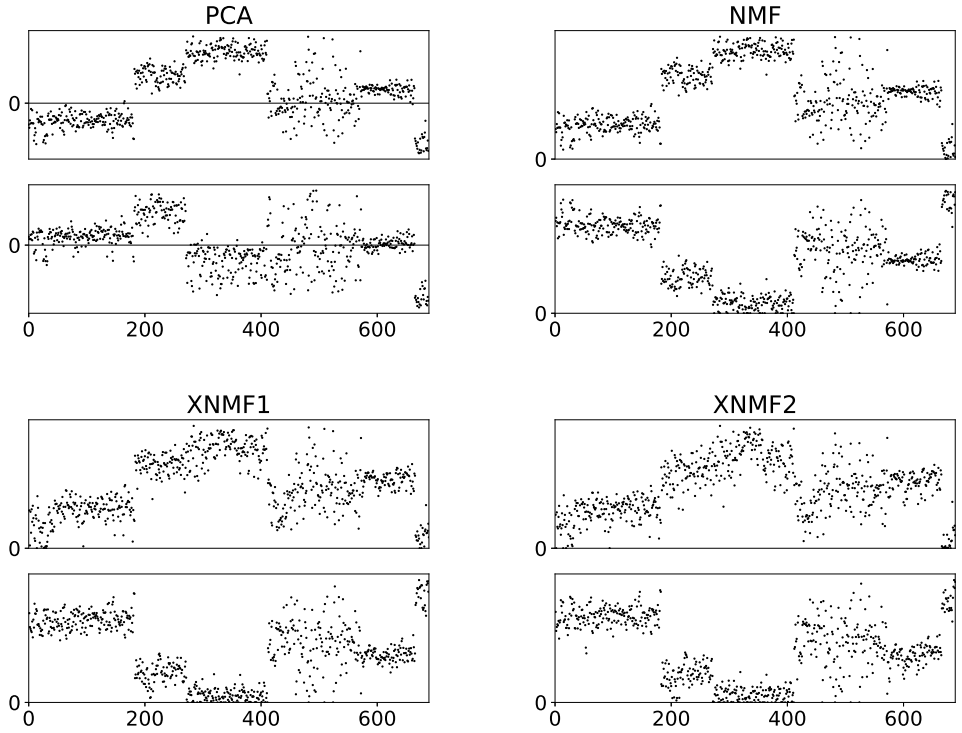


FIGURE 6.7: Plots, for $r_1 = 2$ of the columns of $\mathbf{W}_1$ for NMF, XNMF1 and XNMF2 along with equivalent plots for PCA. Each subplot represents one column of $\mathbf{W}_1$ and each dot is the value in that dimension for one protein.

The second way of looking at $\mathbf{W}_1$ we want to consider is observing each of the columns of $\mathbf{W}_1$ (in contrast to considering the rows of $\mathbf{W}_1$ as we did in Figure 6.6). So each dot is now the value of one protein for that column of $\mathbf{W}_1$ and each plot represents a different column of $\mathbf{W}_1$. We also show equivalent results for PCA as a lot of dimensionality reduction in spatial proteomics is conducted using PCA [39]. We show results for $r_1 = 2$ in Figure 6.7 and with $r_1 = 6$ in Figure 6.8. We see that there are clear groups of proteins in each of the $\mathbf{W}_1$ columns but that the data is highly noisy and challenging to interpret. There are several comments to be made. As has been mentioned before the $\mathbf{V}$

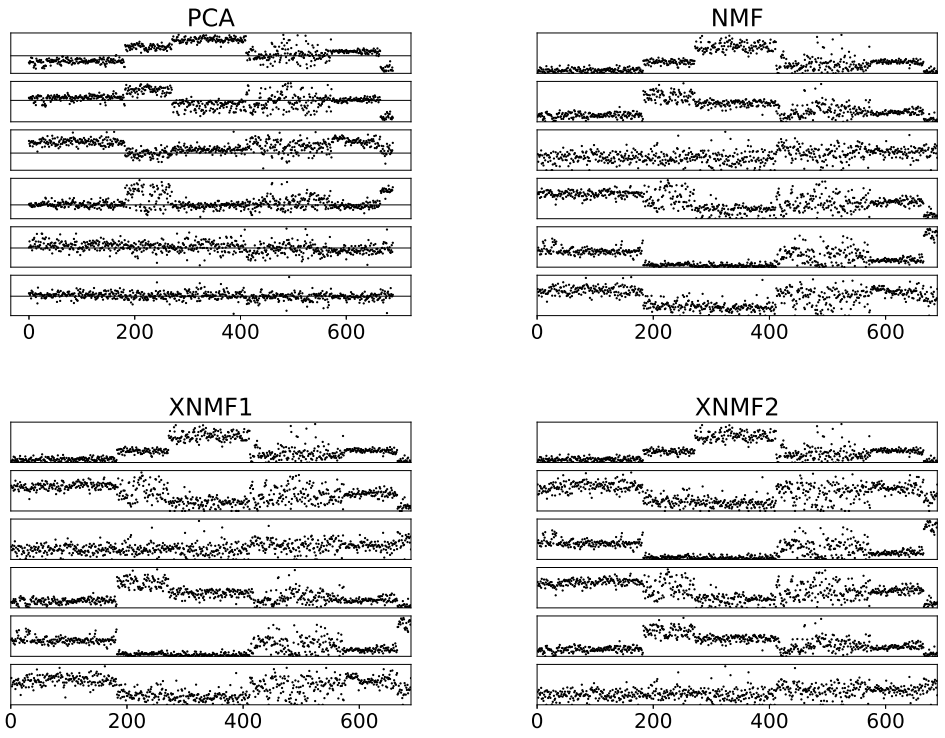


FIGURE 6.8: The equivalent plots as Figure 6.7 but with $r_1 = 6$

matrix is not random - the proteins are effectively lined up in organelle order, so all the known markers from one organelle are next to one another. Therefore we expect to see structure in the data. These plots give us some indication of how similar the proteins are to each other. We can see that the proteins from 1 to approximately 180 are relatively similar in values, for both $r_1 = 2$ and $r_1 = 6$. Conversely, from around protein 400 to 580 there seems to be little relationship between the proteins. There are also other interesting features such as in Figure 6.8 from around protein 180 to 400 virtually all the proteins are around zero in some dimensions (accepting some noise) but the same proteins appear to have two structures in other dimensions where from around 180 to 270 they are very similar and then from 270 to 400 they hold a different structure. It is possible these sort of structures in the data tell us something interesting about the underlying biology, but that would be for future research.

Visualisation is an essential tool [39] for spatial proteomics analytics. In Figure 6.9 we use $r = 3$ to produce a representation and then take two of the columns of the $\mathbf{W}_1$ matrices and plot them as a two dimensional graph - such that each data-point represents one protein. The data separates out into clusters in all the plots but is least separated in the combined version. It is interesting to note what changes occur from the standard NMF (top left) to different versions of XNMF. In the manual XNMF2 we see the blue points shifting towards the yellow which may have significance in that it may

imply there are significant PPI links between the two groups. The choice of $r = 3$ made here is because it provides plots which are highly comparable between all the methods, hence we can easily compare how the clusters in XNMF1 and XNMF2 are less tightly bound, and overlap more with other clusters, than in PCA or NMF.

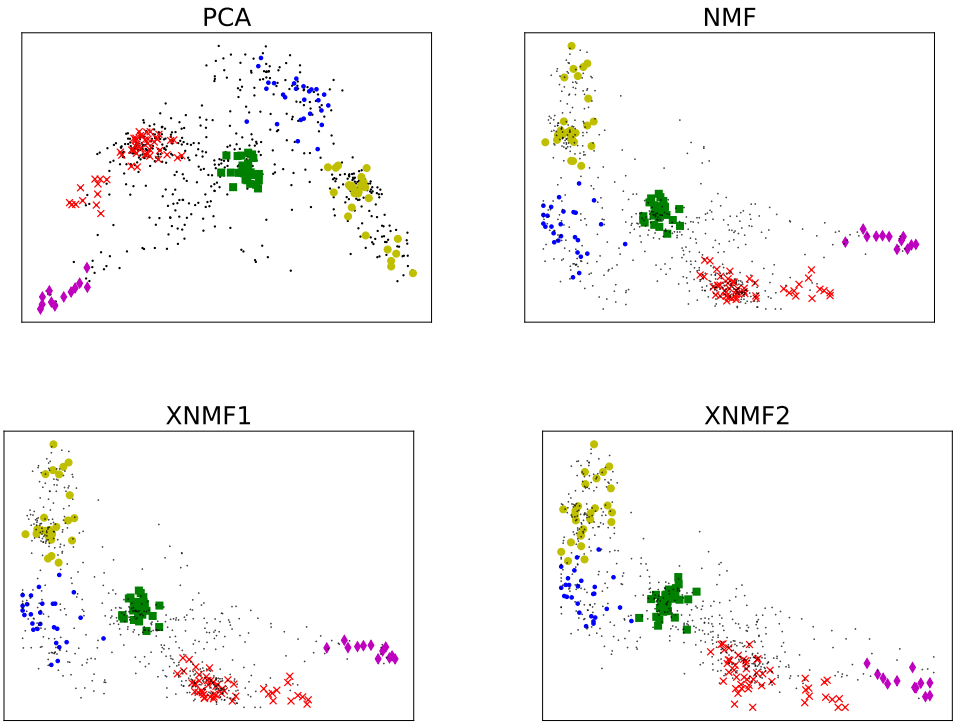


FIGURE 6.9: Visualisations in 2-D of the proteins using PCA, NMF, XNMF1 and XNMF2. We used $r_1 = 3$ and then plotted two of the components. The marker proteins are coloured points. All versions give a good representation of the data. Some of the changes between NMF and the XNMF versions might tell us something of interest about those proteins.

A potential advantage of using XNMF over NMF could be an increased level of sparsity of the factorised matrices. The left plot of Figure 6.10 shows the average sparseness of $\mathbf{W}_1$ as r_2 increases. For each value of r_2 we measured the sparseness of each column of $\mathbf{W}_2$ and averaged it over the r_1 columns of $\mathbf{W}_1$, but as r_1 can also be increased we then averaged over the different values of r_1 to find our final average sparseness value. We repeated the process three times to get the mean and standard deviation we have then plotted. Increasing r_2 does appear to result in marginally higher levels of sparseness in the $\mathbf{W}_1$ columns - which is a potentially useful result as increasing sparseness has considerable value especially in terms of interpretability. However, the magnitude of the change is relatively small. The right plot of Figure 6.10 shows the effect of increasing r_2 on the sparseness of the $\mathbf{W}_2$ columns. For each r_2 value we take the mean sparseness across the columns and plot it along with the standard deviation. There is a clear

increase in the average sparsity as r_2 increases. For this plot we show the mean and standard deviation of the sparseness of the columns for each r_2 value with three repeats.

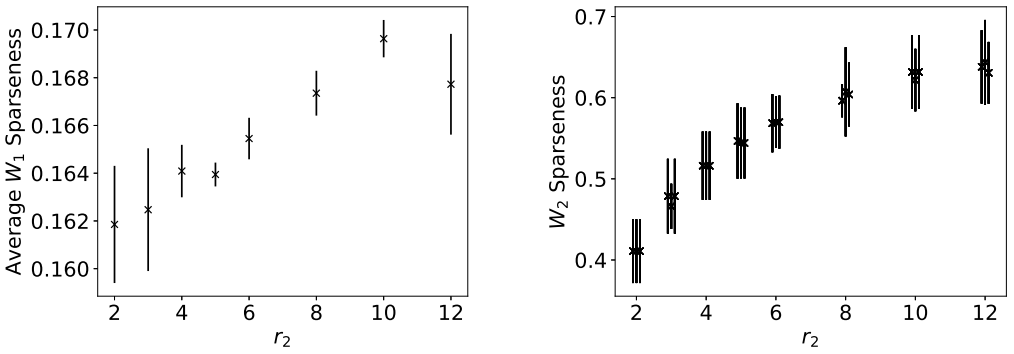


FIGURE 6.10: Left) The change in average sparseness of the $\mathbf{W}_1$ matrices as r_2 increases. Right) The average and standard deviation of the sparseness measure for the $\mathbf{W}_2$ columns as r_2 increases.

6.3.3 Clustering and Classification

We want to consider how NMF and XNMF effect the quality of both clustering and classification. To test the quality of the clustering we will use the Rand index, discussed in the previous section, in two ways. First we will use it to compare how well clustering works in comparison to the real labels. Here the labels are considered as a set of clusters which can then be compared to clusters formed using k-means clustering on raw data as well as dimensionality reduced data using PCA, NMF, and XNMF. The higher the value, the better the clustering algorithm matches the real labels. In Figure 6.11 the Rand index is shown for repeats of the k-means clustering algorithm on different dimensionality reduction methods. We ran ten repeats of the NMF and XNMFs to produce the different points and ran k-means clustering fifteen times for each of the repeats to give a mean and standard deviation. There is little evidence here that using NMF or XNMF produces considerably better results than PCA, although the results using NMF are slightly higher but it is probably not significant. It is of interest that the XNMF control version produces reasonable results. This is probably because we are clustering on $\mathbf{W}_1$ and the addition of the random $\mathbf{W}_2$ matrix does not significantly effect $\mathbf{W}_1$, meaning that the $\mathbf{W}_1$ from the XNMF control version will be very similar to $\mathbf{W}$ from NMF.

The second way we want to measure the clustering is to consider the similarity of the clusters to themselves. Here the clusters could be very different to the labels but if they are consistent they will get high values. In Figure 6.12 we show results for repeated implementations of the k-means clustering. The k-means cluster is applied twice to the raw data, different versions of NMF/XNMF and PCA all with $r_1 = 5$, allowing the Rand

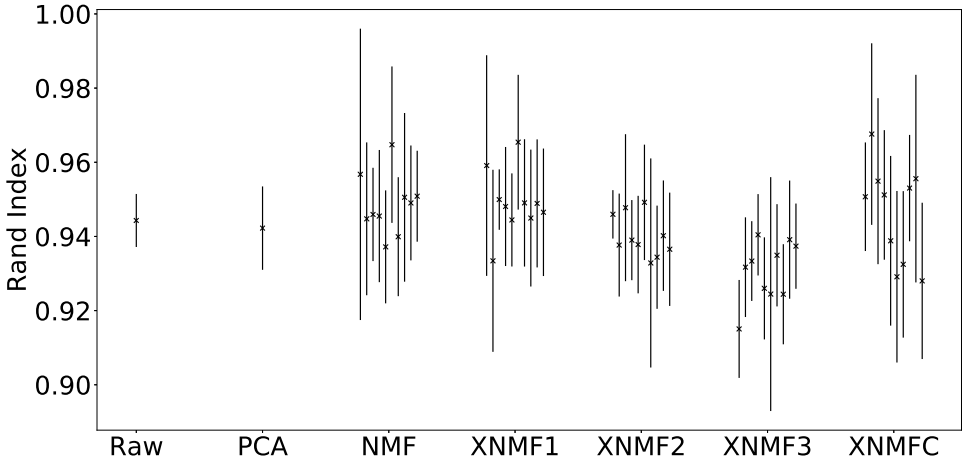


FIGURE 6.11: Quality of the clustering, compared to the real labels. A higher value means that the clustering produces a result closer to the actual labels.

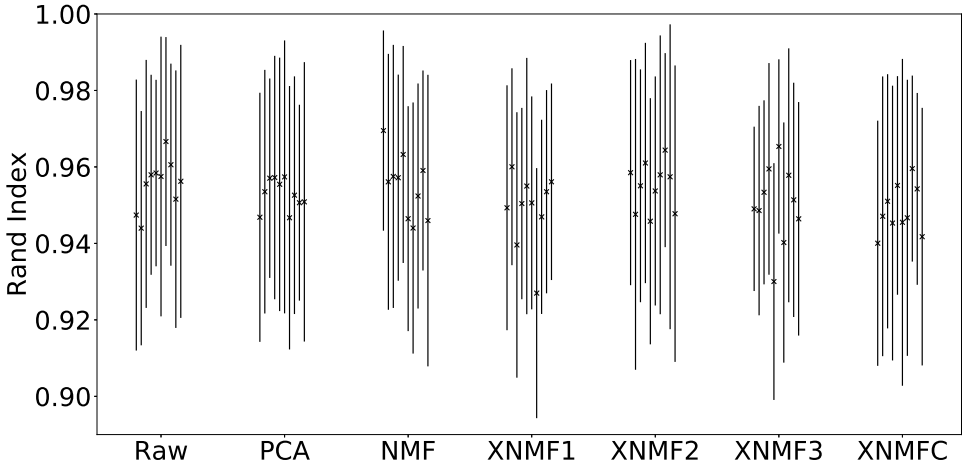


FIGURE 6.12: The second method of measuring consistency of clustering this time comparing across multiple runs of the dimensionality reduction methods. We see no improvement in the consistency of clustering using these techniques.

index to be calculated each time between the two clusterings. This process is repeated twenty times and an average Rand index with standard deviation is measured. We produced ten points for each of the ten different repeats of the NMF/XNMF methods with the errorbars being the standard deviation across the twenty repeats of the k-means clustering algorithm, there is no notable differences in output. We also show repeats for the raw data and the PCA data which should not show differences across the repeats as they both produce the same results. This demonstrates that most of the variation is coming from the k-means clustering algorithm.

The main purpose of spatial proteomics experiments is to classify proteins into the correct organelles. We therefore are interested in whether our XNMF method could produce improved classification accuracy. It is fairly common practice to test unsupervised learning techniques by applying a linear classifier onto the reduced representation to test its ability to effectively represent the data [110]. We therefore apply a linear support vector machine (SVM) to various different representations of the data and examine the results.

In Figure 6.13 (top) we show the results of using a random half of the extended marker set as training and the other half as testing, we repeated 1000 times with different random partitions and show the average and standard deviation for the different methods. In the bottom plot we show the results of training on only the original marker set and testing on the total extended set. We do appear to get a better classification accuracy with the XNMF versions when training using just the original marker set and testing on the extended markers. It is potentially useful to utilise XNMF for improved classification here. However, we get worse results when using XNMF on the extended markers where PCA and NMF both do significantly better than our XNMF variants.

Part of the challenge of drawing conclusions from these results of clustering and classification is that we get generally very good results using all methods, including the raw data, making it difficult to discriminate between the methods. We do see promising results when working from the original markers alone though where we get significantly better results with NMF and XNMF than with PCA or with raw data.

6.4 Summary

In this chapter we have demonstrated that NMF is an effective method to use for the dimensional reduction of spatial proteomics data and that XNMF can be used to add external data into the formulation. As with many unsupervised learning methods, demonstrating conclusively that the method works better than alternatives is challenging.

There are impressive results using standard NMF, with $r = 2$ we see a complete separation of the marker proteins into different compartments, which tallies with the rank estimated using the minimum description length method [126]. Without negative values the columns of the $\mathbf{W}_1$ matrix should be easier to interpret. The 2-d visualisations effectively separate out the marker proteins. Clusters formed do appear to be slightly more consistent when using NMF or XNMF than PCA, but the differences are small and may be within the margin of error. There also may be a signficant advantage of using the NMF or XNMF methods in terms of classification accuracy compared to PCA

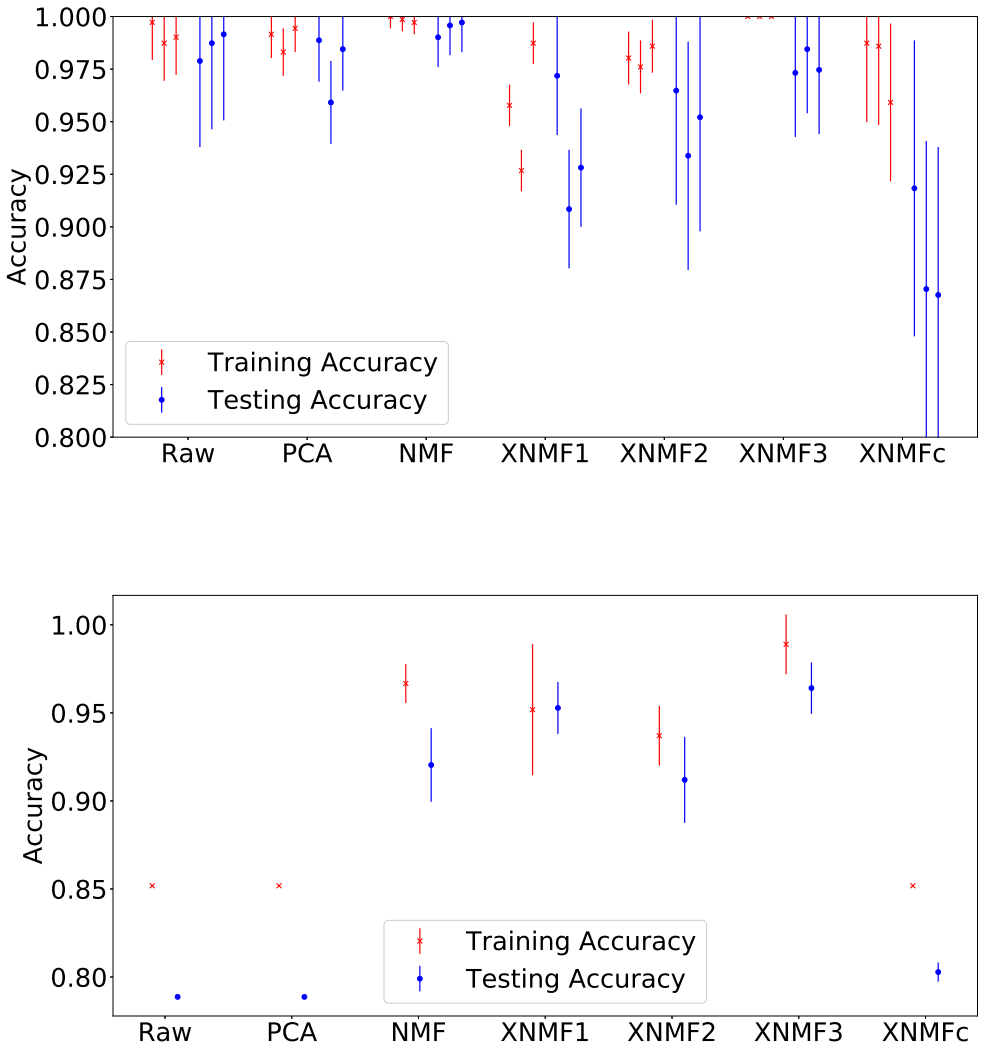


FIGURE 6.13: Top) Classification accuracies using partitions of the extended marker set as training set and another partition as the testing set. With use of concatenated $\mathbf{W}_1$ with $\mathbf{W}_2$. Bottom) Classification accuracies using the original marker data as training and the extended marker set as the testing set.

especially when dealing with small amounts of training data such as the original marker proteins.

There are several potential advantages that we claim for XNMF over NMF, we improve the reproduction quality - with less overfitting than doing so by increasing the r_1 value. With increasing r_2 we find an increasing level of sparseness of the $\mathbf{W}_1$ matrix which should lead to a representation which is easier to interpret. XNMF appears to push some of the marker proteins closer together in the 2-d visualisations - potentially suggesting they might be proteins of interest - such as translocalising (existing in more than one compartment). There, potentially, could be a significant improvement

in generalisation accuracy when using XNMF and training on small data-set. While we have not demonstrated drastic improvements using XNMF there are signs that it might be a useful method with further research with other proteomics data-sets (such as the Drosophila data-set we used in previous work [124, 128]) or different types of biological data.

Chapter 7

A Framework for Performing Variants of Non-Negative Matrix Factorisation using Constrained Autoencoders

In this chapter we explore the use of an autoencoder to perform various versions of NMF. We will consider whether an autoencoder can effectively be used to perform standard NMF and variants of NMF. We are particularly interested in producing a formulation that is flexible enough to cope with a range of disparate datasets. This chapter also feeds into Chapter 8 providing the ideas behind using an autoencoder to perform NMF which then enables us to perform probabilistic NMF.

7.1 Introduction

As the work in this thesis is primarily concerned with unsupervised learning, and specifically non-negative constraints in unsupervised learning we will explore some of the relevant literature regarding neural networks and autoencoder; the application of non-negative constraints in neural networks; how neural networks can be used to perform NMF; and will conclude this section by providing a motivation for the use of autoencoders to perform NMF.

7.1.1 Neural Networks and Autoencoders

A variety of versions of neural networks have recently achieved impressive results in supervised learning, especially convolutional neural networks [69] and recurrent neural networks [54]. There has been somewhat less focus on unsupervised learning in neural networks where autoencoders [115, 52] are one of the key methods.

Autoencoders are a type of unsupervised neural network with a target output the same as its input. The network has some type of constriction which, it is hoped, forces the network to find an efficient encoding of the input data [52]. We will discuss the mathematics behind autoencoders in more detail in Section 7.2.

7.1.2 Non-Negative Constraints in Neural Networks

A key motivation for the application of non-negative constraints into neural networks is to improve interpretability. The activation functions in a neural nework are non-negative if either sigmoids or rectified linear units (ReLU) [98] are applied but the weights of the network, usually, are allowed to be all real numbers. There has been limited significant work published on imposing non-negativity on the weights of a neural network. One such study found imposing non-negativity on the weights resulted in a worse test error but claimed that the network weights were easier to interpret and understand [19].

An one-hidden-layer autoencoder with non-negative weights was introduced by Lemme et al. with the weight matrix kept the same for the encoding and decoding parts [75, 76]. They utilise a sigmoidal activation function with the intrinsic plasticity mechanism [132] which automatically updates the parameters of the sigmoid. This method allows the autoencoder to be directly compared to NMF but has limitations especially in that it ties the weights of the encoder to those of the decoder and uses a sigmoidal activation function. They also use a standard squared error objective function. While their method is effective it does not allow much flexibility of approach, for example they cannot easily increase the depth of their network under their formulation. A neural network approach with non-negativity constraints was also used with an application to audio problems, on which they find improved source separation compared to standard NMF [123], however they do not impose non-negativity on the weights of the network directly, instead imposing sparsity on $\mathbf{H}$.

While some other work has been completed on the use of nonnegativity constraints in autoencoders [56, 99, 6, 4, 5] these have tended to focus on what happens to an autoencoder when a non-negativity constraint is applied to it rather than using autoencoders to specifically perform NMF. They all claim improved interpretability and some claim

that they find more sparse solutions. These claims tend to be made on specific datasets with, presumably, highly tuned methods applied which effectively work on these particular problems.

7.1.3 Using Neural Networks to perform NMF

There has been little work conducted which specifically focuses on using neural networks to actually perform NMF. One example of this was a proposed method which uses a recurrent neural (RNN) to perform NMF [22]. They produce a Lagrangian formulation of NMF and use the RNN to find the Lagrange multipliers. As previously mentioned, the work of Lemme et al. [75, 76] does find a weight matrix and latent activation functions that can be interpreted as performing NMF.

It is also of interest that NMF has been used to find the parameters of neural networks. One paper used semi-NMF and NMF to optimise the layers of a deep neural network without directly using backpropagation [116]. While we do not attempt this, the idea of using NMF in this way to find the parameters of the network might be of future interest.

7.1.4 Motivation

In recent year neural networks have come to dominant many fields of machine learning due, mainly, to the impressive results they have produced. NMF is still mostly conducted using matrix factorisation techniques such as the multiplicative updates. In this chapter we want to explore whether using neural networks to perform NMF has any advantages over more standard methods. To this end we utilise several variations of the autoencoder to peform NMF and analyse whether using an autoencoder for NMF has value.

7.2 Methodology

7.2.1 Basic AE-NMF

A basic autoencoder is shown in Figure 7.1, the mathematics of the autoencoder, if certain constraints are imposed, is the same as for NMF. We consider an input data-matrix $\mathbf{V} \in \mathbb{R}^{m \times n}$ with m dimensions and n data-points. If we then consider one data-point $\mathbf{v}_i \in \mathbb{R}^{m \times 1}$, in NMF we want to produce $\mathbf{x}_i \in \mathbb{R}^{m \times 1}$ where $\mathbf{x}_i = \mathbf{W}\mathbf{h}_i$ and $\mathbf{x}_i \approx \mathbf{v}_i$. For a basic autoencoder with one hidden layer the weight matrix linking the input to the hidden layer is $\mathbf{W}_1 \in \mathbb{R}^{r \times m}$ where r is the number of neurons in the hidden layer. We then have the output of the hidden layer $\mathbf{h}_i = \sigma_1(\mathbf{W}_1\mathbf{v}_i)$ where σ_1 is some

non-linear function that operates element-wise and $\mathbf{h}_i \in \mathbb{R}^{r\times 1}$. The final layer then has a weight matrix $\mathbf{W}_f \in \mathbb{R}^{m\times r}$ and produces output $\mathbf{x}_i = \sigma_f(\mathbf{W}_f\mathbf{h}_i)$. If the activation function σ_f is the identity, and both $\mathbf{W}_f$ and $\mathbf{h}_i$ are non-negative we have an autoencoder which can be interpreted as performing NMF.

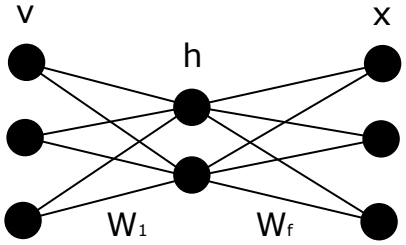


FIGURE 7.1: A basic autoencoder to perform NMF. The weights of $\mathbf{W}_f$ must all be non-negative, the activation function σ_f which operates on the output neurons should be the identity and the activation function, σ_1, that produces $\mathbf{h}$ must be non-negative. The number of neurons in the hidden layer is the subspace size.

For the basic autoencoder to perform standard NMF the constraints that must be met are:

1. The activation function σ_1 must produce a non-negative output.
2. The activation function σ_f should be the identity.
3. The weights of $\mathbf{W}_f$ must be non-negative.
4. The number of hidden units in the hidden layer must be the same as the desired subspace size.

If these conditions are met then the autoencoder will produce a matrix $\mathbf{W}_f$ which is equivalent to the $\mathbf{W}$ from NMF and a set of dimensionally reduced points $\mathbf{h}_i \quad \forall i \in \{1, ..., n\}$ which, combined together, are equivalent to the $\mathbf{H}$ from NMF. We call this autoencoder AE-NMF. This explanation of basic NMF adds nothing of substance to the previous work discussed in the previous section, except for the manner we have presented it. The contribution we are adding in this chapter is below here where we demonstrate how this more flexible framework we are setting out can be used in a variety of different ways.

Within these constraints there is considerable flexibility which enables us to explore a range of choices when performing AE-NMF. These variations fall into two main categories, those that: 1) improve performance and 2) perform different versions of NMF.

Performance here includes: finding a lower error $F = \frac{1}{2}||\mathbf{V} - \mathbf{WH}||^2_{\text{Fro}}$; sparser solutions; a more interpretable representation; faster computation; lower computational requirements; better consistency of results across multiple runs. Different versions of NMF

could include methods such as XNMF [125], seen in Chapter 5, or probabilistic versions of NMF (see Chapter 8). In the rest of this chapter we will consider variations to AE-NMF which may be able to achieve something from either of these two categories.

7.2.2 Variations to AE-NMF

A potential advantage of AE-NMF over more standard forms is the ease with which we can adjust a range of parameters and hyper-parameters (by which we mean the choice of, say, the activation function itself rather than parameters associated with it). We first consider a few relatively small changes which we will review in turn.

The Activation Function

The choice of activation function of the hidden layer is restricted in that it must be non-negative, ruling out options such as the hyperbolic tangent. The most obvious two choices are the sigmoid and rectified linear unit (ReLU). Many modern neural networks use ReLU rather than sigmoids as the activation function of the neurons as they often appear to produce better results [98].

Choice of Optimiser

There are many optimisation choices for performing on a neural network. We consider two of the most common: gradient descent and the Adam optimiser [66].

Method of Imposing Non-negativity on $\mathbf{W}_f$

We consider two options to impose non-negativity on $\mathbf{W}_f$: projection to zero and the gradient descent form of multiplicative updates [72, 42].

The projection to zero method is simple and fast. For all the elements of $\mathbf{W}_f$ the projection method performs $W'_{i,j} = \max(W_{i,j} + \Delta W_{i,j}, 0)$, where $W'_{i,j}$ is the updated element and $\Delta W_{i,j}$ update amount.

For the second method, using multiplicative updates [72], when we implemented them to update the weights of the $\mathbf{W}_f$ matrix we got poor results. It is possible that the problem is that when we are updating the weights of the first layer ($\mathbf{W}_1$) we use a static learning rate that is kept too small. As the learning rate of $\mathbf{W}_f$ is adaptive (see Chapter 5) it is possible that they are scaled too high so that the final layer weights were being updated too much compared to the other layer, leading to poor outcomes. To compensate for this we added an additional learning rate, λ:

$$\mathbf{W} \leftarrow \mathbf{W} - \lambda \frac{[\mathbf{W}]}{[\mathbf{W}\mathbf{H}\mathbf{H}^T]} \circ \nabla_W F. \tag{7.1}$$

The addition of λ still guarantees the monotonic decrease [73] of the objective function but it will be slower.

We will now consider larger variations that warrant more attention: depth of the network; alterations to the objective function; online versus batch approaches; the use of an autoencoder for performing XNMF.

7.2.2.1 Network Depth

Deep neural networks often appear to achieve far better performance than shallower networks [71, 93] even though a neural network with only one hidden layer can be a universal function approximator [24, 55]. An optimal solution (or many optimal solutions) should exist to any NMF problem and it should be possible to find them by a fairly shallow AE-NMF with only two hidden layers. We require two hidden layers in this AE-NMF formulation because we must constrain the number of units (to r units) in the penultimate layer as that is the size of our subspace and for the universal function approximator the number of units cannot be constrained in this way.

It is worth considering whether the apparent benefit of depth in certain neural networks might be true for AE-NMF as well and may enable us to reach better solutions. The constraint here is that the depth must all come before the final layer - we cannot add layers after the constriction otherwise we cannot recreate NMF correctly. However, we can alter the previous layers considerably. In Figure 7.2 we present a visualisation of a deeper network with three hidden layers and with $r = 2$.

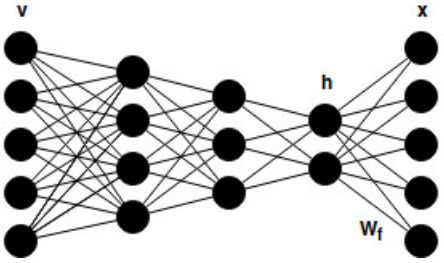


FIGURE 7.2: A deep auto-encoder for performing NMF. We can add as many layers as we want before the constriction (which has two neurons) but, to perform NMF, we must have just one set of weights after the constriction.

7.2.2.2 Alterations to Objective Function

A potentially significant benefit of performing AE-NMF rather than standard NMF is the relative ease of altering the objective function. The example investigated in this work was an attempt to make the objective function "more convex".

A common problem in neural networks and in standard NMF is that the objective function is non-convex. The importance of the non-convexity of either NMF or neural networks is unclear, for example, if there are multiple minima but they all have the same error and provide a similar interpretation then the non-convexity might not be important. For example, people have claimed to demonstrate that even with non-convex functions the local minima which dominate the error space for a deep network all have similar results to the global minimum [64]. If this is the case for NMF then the solutions found should be reasonable and it is not particularly important that the method fails to find a global solution. However, it is unclear if this applies to NMF and there might be advantages in trying to use objective functions which provide a "more convex" error space.

We focus on using techniques from Lo et al. [82, 83, 84, 85]. The idea is that you should adapt the objective function to remove (or reduce) the existence of local minima and saddle points.

Convexification from the point of view of Lo et al. uses a risk averting error (RAE) and normalised risk averting error (NRAE). A common objective function for an MLP is the mean squared error (MSE):

$$Q(\mathbf{W}) = \frac{1}{n}\sum_{i=1}^{n} ||\mathbf{v}_i - \hat{\mathbf{v}}_i||^2 \tag{7.2}$$

where n is the number of data-points, $\mathbf{v}_i$ is the target output and $\hat{\mathbf{v}}_i$ is the predicted output. Now the RAE is given by

$$J_\lambda(\mathbf{W}) \equiv \sum_{i=1}^{n} \exp\left(\lambda||\mathbf{v}_i - \hat{\mathbf{v}}_i||^2\right) \tag{7.3}$$

where λ is called the risk sensitivity index. The larger is λ the larger the convex region, however, if λ is large or $||\mathbf{v}_i - \hat{\mathbf{v}}_i||^2$ is large then J_λ will cause a numerical overflow as the numbers become too large to manage.

To counter these numbers getting too large Lo et al. introduced the normalised risk averting error (NRAE):

$$C_\lambda(\mathbf{W}) \equiv \frac{1}{\lambda} \ln \left[\frac{1}{n} J_\lambda(\mathbf{W}) \right]. \tag{7.4}$$

While the NRAE does prevent the production of too large numbers it does not immediately prevent the previous problem - J_λ still need to be computed before the natural log can be applied and that will overload the computer. Lo et al. [83] therefore introduce some new terms to correct for this problem. We now have:

$$C_\lambda(\mathbf{W}) = \frac{1}{\lambda} \ln \frac{1}{n} + ||\epsilon_M(\mathbf{W})||^2 + \frac{1}{\lambda} \ln \Big[\sum_{i=1}^{n} \eta_i(\mathbf{W}) \Big] \tag{7.5}$$

where

$$\epsilon_i(\mathbf{W}) \equiv \mathbf{v}_i - \hat{\mathbf{v}}_i(\mathbf{W}), \tag{7.6}$$

$$\epsilon_M(\mathbf{W}) \equiv \max_{i \in [1,\ldots,n]} ||\epsilon_i(\mathbf{W})||^2, \tag{7.7}$$

$$\eta_i(\mathbf{W}) \equiv \exp(\lambda(||\epsilon_i(\mathbf{W})||^2 - \epsilon_M(\mathbf{W})||^2. \tag{7.8}$$

We use automatic differentiation built into PyTorch but we also compared the gradients found with those of the derivative of C_λ:

$$\frac{\partial C_\lambda}{\partial w_i} = \frac{1}{\lambda J_\lambda} \frac{\partial J_\lambda}{\partial w_i} \tag{7.9}$$

$$= \frac{-2 \sum_{k=1}^{n} \eta_k \epsilon_k^T \frac{\partial \hat{v}_k}{\partial w_i}}{\sum_{k=1}^{n} \eta_k}, \tag{7.10}$$

which produces the same result. Therefore we continue to use the automatic differentiation method through the network.

While we investigate altering the objective function in this particular manner using the NRAE and RAE there may be many other objective functions of interest such as the MDL approach in Chapter 4. In the next section we show the results from the application of the NRAE and RAE objective functions in the AE-NMF framework. Perhaps more importantly, we demonstrate the ease of investigating new objective functions using the AE-NMF framework.

7.2.2.3 Online versus Batch

Many NMF techniques such as multiplicative updates [72] or hierarchical alternating least squares [42] are batch processes and are not easily converted into an online form. Online updates is a useful feature under several circumstances especially if you are working in areas where the data continuously comes in or if the data-set are too large to hold in your memory at one time. Using AE-NMF it is relatively straightforward to adapt to an online approach. In fact, much of the neural network background code that exists is designed to mostly work in an online (or mini-batch) manner, with methods such as stochastic gradient descent.

7.2.2.4 Autoencoder for XNMF

In Chapter 5 [125] we provided a novel method for including external data into the NMF formulation, called XNMF. We can perform the same method using AE-NMF, which we then call AE-XNMF. In Figure 7.3 we display the layout of AE-XNMF. The black solid lines represent all the weights that can be updated while the red dot-dashed lines are those weights which are fixed.

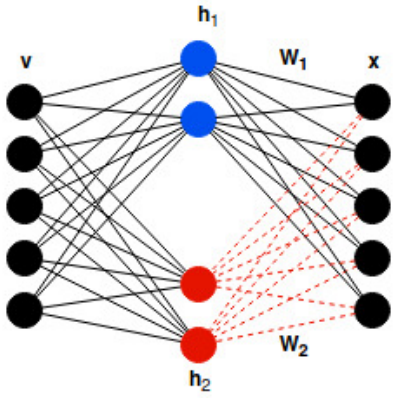


FIGURE 7.3: An autoenoder with one hidden layer designed to perform XNMF. The black lines are all weights that must be learnt while the red lines represent the external subspace and are constant.

In AE-XNMF the final layer produces $\mathbf{V}_p = \mathbf{W}_1\mathbf{H}_1 + \mathbf{W}_2\mathbf{H}_2$ where $\mathbf{W}_1$ are the weights that can vary, $\mathbf{W}_2$ the set weights, while $\mathbf{H}_1$ and $\mathbf{H}_2$ are the activations of the previous layer. This formulation is identical to XNMF but has the same potential variations outlined above that may enable improved versions of XNMF.

7.2.3 Transfer Learning for AE-NMF

Transfer learning in neural networks involves using a previously trained network, or part of a previously trained network, and using it for a new problem [109, 107]. The use of transfer learning in AE-NMF may provide several advantages over training a network from scratch including: improving the speed of training; producing more consistent solutions over repeated runs which also might make comparisons between techniques easier; reducing overfitting; and producing more generalised results. In Figure 7.4 we display our method of performing transfer learning for AE-NMF.

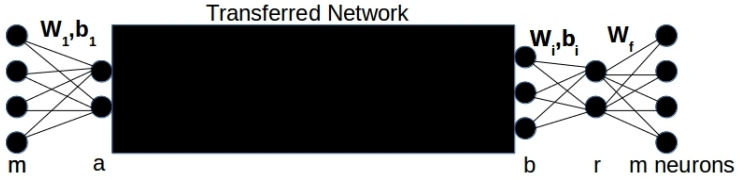


FIGURE 7.4: A transfer learning layout for an AE-NMF. We have to learn the weights and biases labelled in the diagram fresh each time we reuse the network. Re-learning the first and i^{th} layers allow us to take any dimensional space and apply this method. The final layer allows us to choose our subspace size and vary the output size.

The transferred network is found in some manner in a previous step - as we hope it will uncover some form of structure in data it is not necessary to specify the method used. Supervised or unsupervised learning methods might both work if they produce a sensible function between the input and output.

All the weights and biases in the transferred network remain the same. We have to update $\mathbf{W}_1$, $\mathbf{b}_1$, $\mathbf{W}_i$, $\mathbf{b}_i$ and $\mathbf{W}_f$, where we do not specify which layer i is as the depth of the transferred network does not need to be constrained. The structure of the transfer learning part of the AE-NMF is also not heavily constrained as all the NMF constraints occur in the final layer. There are a neurons at the entrance to the transferred network and b at the exit. To adapt the data to the particular transferred network requires that $\mathbf{W}_1 \in \mathbb{R}^{m\times a}$, $\mathbf{b}_1 \in \mathbb{R}^{1\times a}$, $\mathbf{W}_i \in \mathbb{R}^{b\times r}$ and $\mathbf{b}_i \in \mathbb{R}^{1\times r}$.

Part of the logic behind transfer learning is that deep neural networks learn a representation characteristic of the problem domain with simpler patterns in earlier layers and more complex forms as the depth increases [155]. Therefore some transfer learning methods take the early layers, which for images tend to form general patterns like straight lines and curves, and hold them constant whilst altering the later layers [151]. Our method partially does that but also has a first layer which alters the dimensionality of the data to match that of the transferred network.

7.3 Results and Discussion

In this chapter we are exploring the use of autoencoders for performing NMF, we are not planning on finding optimal solutions to particular data-sets. Instead we will investigate the themes we discussed in the previous section and see if we can draw conclusions about the value, or otherwise, of using autoencoders to perform NMF. We test these methods and variations on the same three data-sets we have regularly used throughout this thesis (images of faces, FTSE 100 and genetic data) and also include a reduced (to speed up computational time) version of MNIST with the digits 1, 2 and 6 included.

7.3.1 Minor Choices

First we consider the relatively minor choices discussed in the previous section that must be made to conduct NMF using an autoencoder: choice of activation function, update rules for the network and methods of imposing non-negativity.

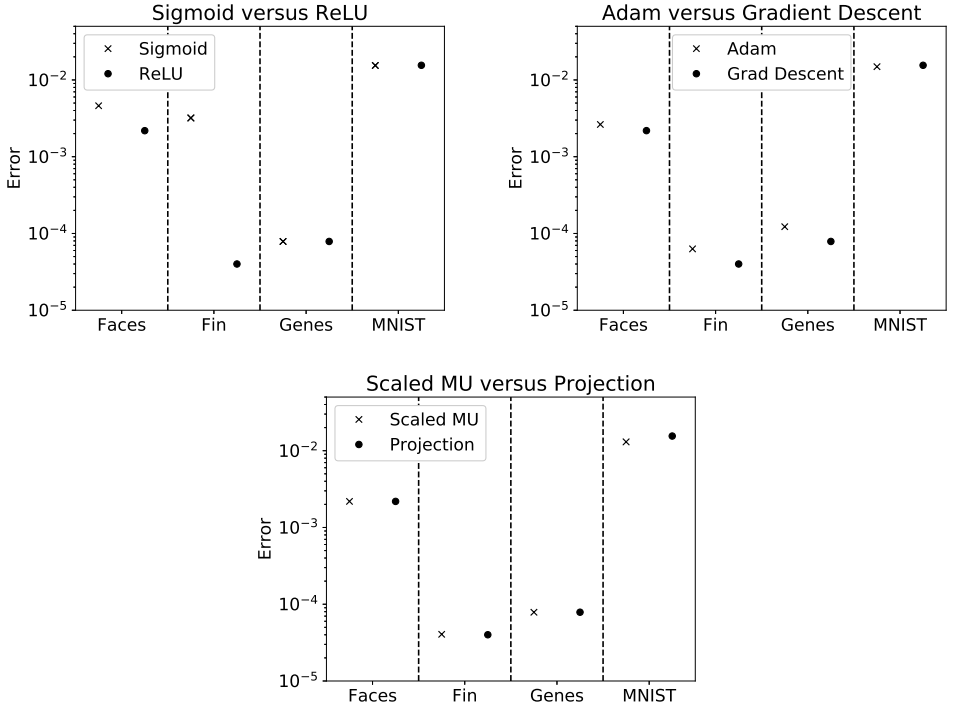


FIGURE 7.5: (Top left) Final errors for the sigmoid and ReLU activation functions. (Top right) Final erors for the Adam and gradient descent methods of updating the network. (Bottom) Final errors of the scaled MU and projection methods of keeping $\mathbf{W}$ non-negative.

We empirically tested these three choices and show the results in Figure 7.5. We ran repeats of these tests but the results are almost identical with errorbars that are not possible to see. The first plot is of using a sigmoid or ReLU activation function for the

production of $\mathbf{h}_i$. We find better results using the ReLU but only very marginally for the genomic and MNIST data-sets. For the faces data-set the differences are substantial (note that we use a log scale so all four data-sets are visible). The FTSE 100 data (labelled at Fin for financial) gives drastically different results. Here the network with the sigmoid activation function is failing to find an appropriate solution for some unknown reason. We did not spend much time investigating this failure as the improvements caused by using ReLU have already been noted [98].

The second plot of Figure 7.5 shows the result of updating the network using the ADAM optimiser [66] or standard gradient descent methods. There is not a significant difference between the two methods but, perhaps surprisingly, the gradient descent tends to do a little better.

The final plot shows the effect of using the scaled multiplicative updates or the projection to zero method. There is very little difference between these methods, throughout most of the rest of the work we use the projection method as it is computationally faster and does not require any choice of learning rate.

Throughout the rest of this chapter we have mostly used the ReLU, gradient descent and projection methods due to the results found here. However, we also note that improved tuning or different data-sets might have resulted in different outcomes. What is fairly clear though is that, with the exception of the sigmoid versus ReLU for the faces and FTSE 100 data-sets the differences in outcome are relatively small, and therefore it is unlikely that it makes a considerable difference to any final conclusions. In addition, as we are exploring the use of AE-NMF rather than trying to specifically find optimal solutions the tests performed here should be sufficient. For different data-sets these conclusions might be different, which adds weight to our argument that our method, being flexible, should enable improvements over more rigid methods.

7.3.2 Online Approaches

One of the potential benefits of conducting AE-NMF rather than standard NMF is that performing NMF in an online manner is easier in AE-NMF. In the left plot of Figure 7.6 we see results of the online approach using both Adam and gradient descent where Adam does better for all four data-sets. In the right plot we demonstrate the difference in final error between a batch and an online approach, the batch method provides better final errors but the online approach still gets to good results which effectively recreate the data.

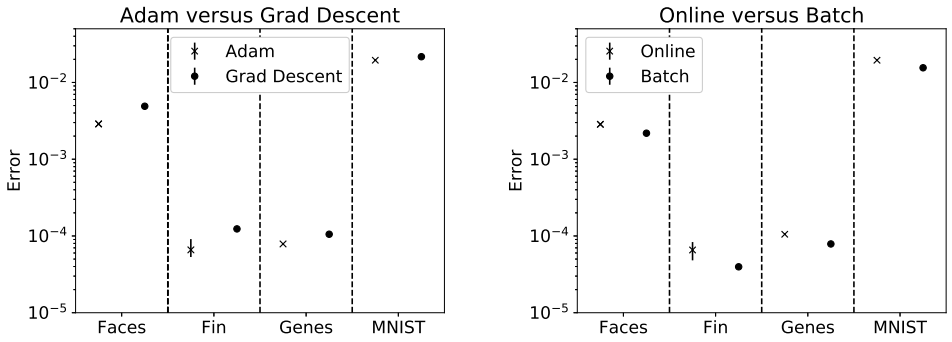


FIGURE 7.6: Left) Final errors for two versions (grad descent and Adam) of the online algorithm, both use the projection method to keep the weights non-negative. Right) A comparison of final errors for the online and batch methods.

7.3.3 Applying a more Convex Objective Function

In Figure 7.7 we apply the RAE and NRAE approaches of Lo et al. We attempted two methods, one using the NRAE alone and one using NRAE followed by RAE. Neither end up producing results which can match or better using the standard Frobenius norm objective function. These results do not tally with the empirical evidence of improvement in final error found by Lo et al. [82, 83, 84, 85]. It may be the case that on some data-sets the RAE or NRAE approaches do work effectively, in which case using AE-NMF with these altered objective functions would be a simple and effective strategy. While, on these datasets, we do not find an improvement in performance it does demonstrate the flexibility of approach of using AE-NMF over standard NMF.

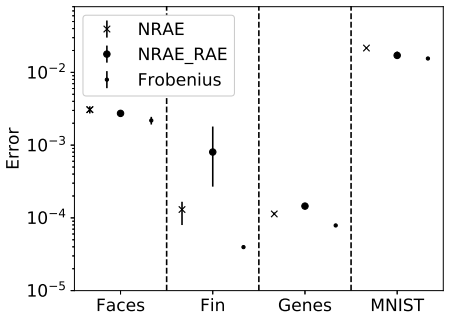


FIGURE 7.7: A comparison of final squared error found by the NRAE, NRAE-RAE and normal objective functions. We find no improvement using the RAE or NRAE over a standard Frobenius norm objective function.

7.3.4 Deep AE-NMF

In Figure 7.8 we show results of testing the network with two and six hidden layers and compare it to standard NMF. In the top left plot we see the results, the two autoencoders

perform much worse than standard NMF. The reason is that for both of these networks some of the final layers are being turned off, which has the effect of reducing the effective dimensionality further, we show the number of neurons being turned off in the top right plot for different values of r. When we correct for this effect in the bottom plot and compare the results of the autoencoders with their effective r-value we see almost identical results between standard NMF and the autoencoder versions. Perhaps the most likely reason for this is that both the autoencoders and NMF are finding a result that is close to the global minimum but we definitely cannot draw a conclusion here that using a deep version of AE-NMF provides any clear benefit.

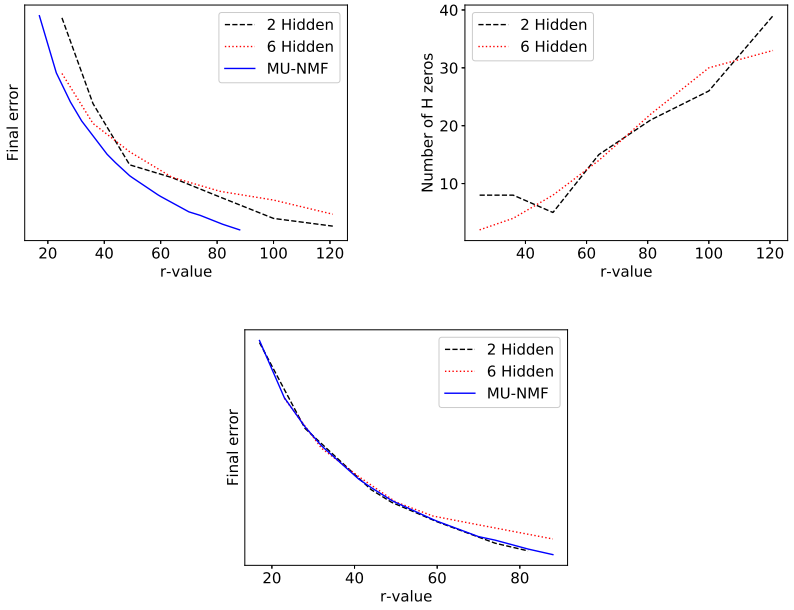


FIGURE 7.8: a) Final errors for different starting r-values for two types of deep AE-NMF and standard NMF. b) The increasing number of H-zeros for increasing starting r-value for the AE-NMF networks with different depths. c) The final errors for different starting r-values once we factor out the H-zeros, we see no difference between the AE-NMF and standard NMF errors.

7.3.5 XNMF using AE-NMF

We demonstrate here the further flexibility of AE-NMF by its simple alteration to produce similar results to those of XNMF [125]. We use the same FTSE 100 data as used in Chapter 5, apply AE-XNMF for a range of $r-$values and compare with XNMF. In Figure 7.9 we show that AE-XNMF and XNMF produce very similar results across the range of r_1 values. There are three repeats shown on these plots for both XNMF and AE-NMF but they produce such similar results it is hard to see the errorbars, both AE-NMF and XNMF produce highly consistent results. It is interesting that AE-XNMF actually seems to produce a better solution than XNMF at low r values, which we would

not have expected since we are holding $\mathbf{W}_f$ non-negative by setting negative values to zero. It is possible that XNMF is getting trapped in local minima or saddlepoints and that AE-XNMF is able to escape. However, at the optimal r value (see Chapter 3) of around 8-10 the results are almost identical.

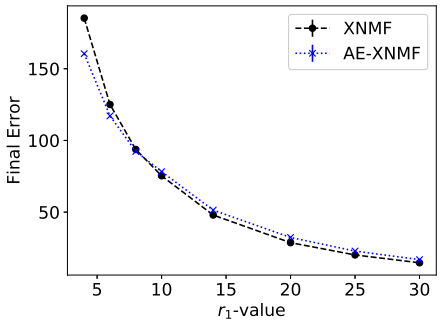


FIGURE 7.9: Final errors against r_1 for XNMF and AE-XNMF. The results are similar for most values of r but at the low end AE-XNMF appears to do better than XNMF, and at the other end the reverse appears to be true.

7.3.6 Transfer Learning in AE-NMF

A benefit of transfer learning could be that it enables the relationships between features found in one related type of data to be transferred to another. A standard explanation of this is in images where straight lines and curves are highly transferrable between many different types of images so a model which extracts these features from images should be useful for a wide range of image data-sets. There are several potential advantages to this method, one being much faster learning on the final network as you do not have to update as many weights. Perhaps of even more importance is that this form of transfer learning can enable small data-sets to be used when training from a randomly initialised network might be impossible.

We tested this potential benefit of transfer learning by training our transfer network on the faces data-set and testing on MNIST-reduced data. We performed three versions of this method: (Figure 7.10 one with transfer learning (black solid line), one with transfer learning but with random weights (blue dashed line) and one from scratch (red dotted line). We performed five repeats for each of the three versions. The results are promising in that our transfer learning method finds the lowest error and the best minima. Interestingly the randomised transferred network outperforms the network trained from scratch - implying that the network trained from scratch is getting stuck in very poor local minima or saddle-points.

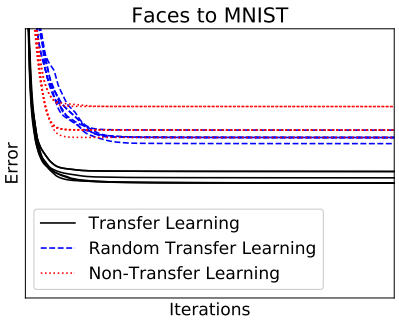


FIGURE 7.10: Transfer learning from the faces data-set to the MNIST-reduced data-set showing significant improvements in performance over learning from scratch.

We performed the same process with the financial data. Instead of using a different data-set we took the FTSE-100 data but from a previous time period. This may not work well as the features extracted (columns of $\mathbf{W}$) will be over a very different period and may not transfer well. We trained the original network and selected the network with the lowest error to be the transferred network and then tested our final network five times using the transferred network. The results are in Figure 7.11. We do not see any improvement with the transferred network which may be because the network trained from scratch has no problem in finding a reasonable solution. The transferred network also may not be aligned well with the second data-set and the features extracted from the first data-set may not be relevant.

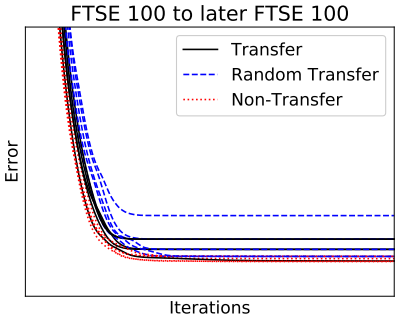


FIGURE 7.11: Transfer learning for the financial data with the transferred network learnt with data from a previous time period and tested on a later period. We see no improvement - in fact running from scratch seems to provide an improvement in accuracy.

7.4 Summary

Standard matrix factorisation methods for performing NMF such as multiplicative updates [72] appear to be highly effective at finding good solutions. However, they lack

flexibility and can suffer in terms of computational resources required for large data-sets. AE-NMF is a highly flexible approach which enables the application of multiple methods to attempt to find a good solution for a particular data-set. In addition, for any new data-point, $\mathbf{v}_i$ we can find its latent representation, $\mathbf{h}_i$, by passing it through the AE-NMF network.

The application of AE-NMF enables different objective functions to be imposed, easy application of an online approach, the flexibility over a range of hyper-parameters (such as update methods or optimization schemes) and adjustments to different methods such as XNMF. This overall flexibility allows for experimentation on a data-set until desired results are achieved.

We found comparable performance to standard NMF and to XNMF using our autoencoder method. However, we did not find any evidence that AE-NMF could improve on NMF. It is possible that NMF applied to these data-sets finds very good solutions so we are unlikely to surpass them.

In the next chapter we will investigate using a variational autoencoder to perform probabilistic NMF, that work is based upon the principles outlined in this chapter of using an autoencoder to perform standard NMF.

Chapter 8

A Variational Autoencoder for Probabilistic Non-Negative Matrix Factorisation

We introduce and demonstrate the variational autoencoder for probabilistic non-negative matrix factorisation (PAE-NMF). We design a network which can perform NMF and add in aspects of a VAE to make the coefficients of the latent space probabilistic. By restricting the weights in the final layer of the network to be non-negative and using the non-negative Weibull distribution we produce a probabilistic form of NMF which allows us to generate new data and find a probability distribution that effectively links the latent and input variables. We demonstrate the effectiveness of PAE-NMF on three heterogeneous datasets: images, financial time series and genomic.

8.1 Introduction

8.1.1 Using Autoencoders for NMF

Several authors have studied the addition of extra constraints to an autoencoder to perform NMF [76, 6, 56] and we explored these idea in Chapter 7. These methods show some potential advantages over standard NMF including the implicit creation of the $\mathbf{H}$ matrix and straightforward adaptation to online techniques. For completeness we recreate the diagram of an autoencoder for performing NMF from Chapter 7 in Figure 8.1.

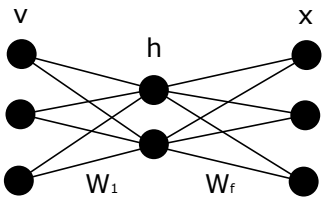


FIGURE 8.1: Diagram of an autoencoder designed to perform NMF. The weights of the final layer, $\mathbf{W}_f$, become the directions of the subspace, with the outputs of the hidden layer, $\mathbf{h}$, as the coefficients in that new subspace. The activation function that produces $\mathbf{h}$ must be non-negative as must all the elements of $\mathbf{W}_f$.

8.1.2 Variational Autoencoders

NMF and autoencoders both produce a lower dimensional representation of some input data. However, neither produce a probability model, just a deterministic mapping. It is also not obvious how to generate new data from these lower dimensional spaces. Generative models solve both of these problems, enabling a probability distribution to be found linking the input and latent spaces whilst enabling new data to be created.

One of the most popular recent generative model is the variational autoencoder (VAE) [67, 113]. A VAE is a probabilistic model which utilises the autoencoder framework of a neural network to find the probabilistic mappings from the input to the latent layers and on to the output layer. Unlike a standard autoencoder the VAE finds a distribution between the latent and seen variables, which also enables the production of new data by sampling from the latent distributions.

8.1.3 Probabilistic Non-Negative Matrix Factorisation

Several authors have presented versions of NMF with a probabilistic element [105, 17, 119] which involve the use of various sampling techniques to estimate posteriors. Other work has been done using hidden Markov models to produce a probabilistic NMF [94] and utilising probabilistic NMF to perform topic modelling [87]. However, to our knowledge no one has utilised the ideas behind the VAE to perform NMF.

Although probabilistic methods for NMF have been developed, even a full Bayesian framework still faces the problem that, for the vast majority of problems where NMF is used, we have little idea about what is the appropriate prior. We would therefore be forced to do model selection or introduce hyperparameters and perform inference (maximum likelihood or Bayesian) based on the evidence. However, as the posterior in such cases is unlikely to be analytic this is likely to involve highly time consuming

Monte Carlo. In doing so we would expect to get results close to those we obtain using PAE-NMF. However, for machine learning algorithms to be of value they must be practical. Our approach, following a minimum description length methodology, provides a principled method for achieving automatic regularisation. Because it fits within the framework of deep learning it is relatively straightforward and quick to implement (using software such as Keras or PyTorch with built-in automatic differentiation, fast gradient descent algorithms, and GPU support). In addition, our approach provides a considerable degree of flexibility (e.g. in continuous updating or including exogenous data), which we believe might be much more complicated to achieve in a fully probabilistic approach.

The next section lays out the key alterations needed to a VAE to allow it to perform NMF and is our main contribution in this chapter.

8.2 PAE-NMF

The model proposed in this chapter provides advantages both to VAEs and to NMF. For VAEs, by forcing a non-negative latent space we inherit many of the beneficial properties of NMF; namely we find representations that tend to be sparse and often capture a parts based representation of the objects being represented. For NMF we introduce a probabilisitic representation of the vectors $\mathbf{h}$ which models the uncertainty in the parameters of the model due to the limited data.

8.2.1 Ideas Behind PAE-NMF

Kingma and Welling [67] proposed the VAE for which the aim is to perform inference where the latent variables have intractable posteriors and the data-sets are too large to easily manage. Two of their contributions are showing that the "reparameterization trick" allows for use of standard stochastic gradient descent methods through the autoencoder and that the intractable posterior can be estimated.

In a standard autoencoder we take some data point $\mathbf{v} \in \mathbb{R}^m$ and run it through a neural network to produce a latent variable $\mathbf{z} = f(\mathbf{v}) \in \mathbb{R}^r$ where $r < m$ and f is some non-linear element-wise function produced by the neural network. This is the encoding part of the network. We then run $\mathbf{z}$ through another neural network to produce an output $\hat{\mathbf{v}} = g(\mathbf{z}) \in \mathbb{R}^m$, which is the decoding part of the network. The hope is that due to $r < m$ the latent variables will contain real structure from the data.

A standard VAE differs in that instead of encoding a deterministic variable z we find a mean, μ, and variance, σ^2 of a Gaussian distribution. If we want to generate new data we can then sample directly from this distribution to get z and then run z through the decoding part of the network to produce our new generated data. As well as sampling from a distribution, VAEs also differ in that they require a different objective function.

PAE-NMF utilises the same objective function as a standard VAE, so for each data-point, we are minimising:

$$\begin{aligned} \text{obj} =& \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{v})}\Big(-\log\Big(p_\theta(\mathbf{v}|\mathbf{z})\Big)\Big) + \mathrm{D}_{\mathrm{KL}}(q_\phi(\mathbf{z}|\mathbf{v})||p(\mathbf{z})) && (8.1)\\ \approx& \frac{1}{2\sigma^2}||\mathbf{v}-\hat{\mathbf{v}}||^2 + \mathrm{D}_{\mathrm{KL}}(q_\phi(\mathbf{z}|\mathbf{v})||p(\mathbf{z})) && (8.2) \end{aligned}$$

where D_{KL} is the KL divergence, ϕ and θ represent the parameters of the encoder and decoder, respectively, $\mathbf{v}$ is an input vector with $\hat{\mathbf{v}}$ the reconstructed vector. The first term represents the reconstruction error between the original and recreated data-point. The second term is the KL divergence between our prior expectation, $p(\mathbf{z})$, of the distribution of $\mathbf{z}$ and the representation created by the encoding part of our neural network, $q_\phi(\mathbf{z}|\mathbf{v})$. We can interpret this as a regularisation term, it will prevent much of the probability density being located far from the origin, assuming we select a sensible prior. Another way of looking at it is that it will prevent the distributions for each datapoint being very different from one another as they are forced to remain close to the prior distribution.

This objective function can also be interpreted as the amount of information needed to communicate the data [68]. We can think of vectors in latent space as code words. Thus to communicate our data, we can send a code word and an error term (as the errors fall in a more concentrated distribution than the original message they can be communicated more accurately). The log-probability term can be interpreted as the message length for the errors. By associating a probability with the code words we reduce the length of the message needed to communicate the latent variables (intuitively we can think of sending the latent variables with a smaller number of significant figures). The KL divergence (or relative entropy) measures the length of code to communicate a latent variable with probability distribution $q_\phi(\mathbf{z}|\mathbf{v})$. Thus by minimising the objective function we learn a model that extracts all the useful information from the data (i.e. information that allows the data to be compressed), but will not over-fit the data.

8.2.2 Structure of the PAE-NMF

The structure of our PAE-NMF is given in Figure 8.2. The input is fed through an encoding neural network which produces two vectors $\mathbf{k}$ and $\boldsymbol{\lambda}$, which are the parameters of the $q_\phi(\mathbf{h}|\mathbf{x})$ distribution. The latent vector for that data-point, $\mathbf{h}$, is then created by sampling from that distribution. However, this causes a problem when training the network because backpropagation requires the ability to differentiate through the entire network. In other words, during backpropagation we need to be able to find $\frac{\partial h_i}{\partial k_i}$ and $\frac{\partial h_i}{\partial \lambda_i}$, where the i refers to a dimension. If we sample from the distribution we cannot perform these derivatives. In variational autoencoders this problem is removed by the "reparameterization trick" [67] which pushes the stochasticity to an input node, which does not need to be differentiated through, rather than in the middle of the network. A standard variational autoencoder uses Gaussian distributions. For a univariate Gaussian the trick is to turn the latent variable, $h \sim q(h|x) = \mathcal{N}(\mu, \sigma^2)$, which cannot be differentiated through, into $h = \mu + \sigma\epsilon$ where $\epsilon \sim \mathcal{N}(0, 1)$. The parameters μ and σ are both deterministic and can be differentiated through and the stochasticity is added from outside.

We cannot use the Gaussian distribution because we need our h_i terms to be non-negative to fulfill the requirement of NMF. There are several choices for probability distributions which produce only non-negative samples, we use the Weibull distribution for reasons detailed later in this section. We can sample from the Weibull distribution using its inverse cumulative distribution and the input of a uniform distribution at an input node. In Figure 8.2 we display this method of imposing stochasticity from an input node through $\boldsymbol{\epsilon}$. Similarly to using a standard autoencoder for performing NMF the same restrictions, such as $\mathbf{W}_f$ being forced to stay non-negative, apply to the PAE-NMF.

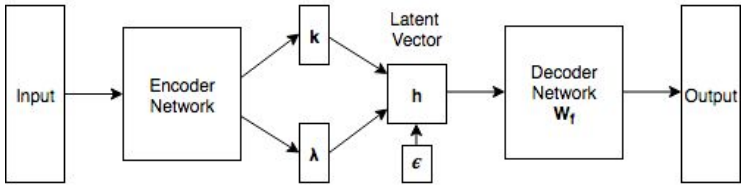


FIGURE 8.2: General PAE-NMF with stochasticity provided by the input vector $\boldsymbol{\epsilon}$.

8.2.3 Details of the PAE-NMF

In this paper we have utilised the Weibull distribution which has a probability density function (PDF) of

$$f(x) = \begin{cases} \frac{k}{\lambda}\left(\frac{x}{\lambda}\right)^{k-1} \exp\left(-(x/\lambda)^k\right) & \text{if } x \geq 0 \\ 0 & \text{if } x < 0 \end{cases}$$

with parameters k and λ. The Weibull distribution satisfies our requirements: that the PDF is zero below $x = 0$, falls towards 0 as x becomes large and is flexible enough to enable various shapes of distribution. For each data point there will be r Weibull distributions generated, one for each of the subspace dimensions.

To perform PAE-NMF we need an analytical form of the KL divergence, so we can differentiate the objective function, and a way of extracting samples using some outside form of stochasticity. The KL divergence between two Weibull distribution is given by [9]

$$\begin{aligned} D_{KL}(F_1||F_2) &= \int_0^\infty f_1(x|k_1,\lambda_1) \log\left(\frac{f_1(x|k_1,\lambda_1)}{f_2(x|k_2,\lambda_2)}\right) dx \\ &= \log\left(\frac{k_1}{\lambda_1^{k_1}}\right) - \log\left(\frac{k_2}{\lambda_2^{k_2}}\right) + (k_1 - k_2)\left[\log(\lambda_1) - \frac{\gamma}{k_1}\right] \\ &\quad + \left(\frac{\lambda_1}{\lambda_2}\right)^{k_2} \Gamma\left(\frac{k_2}{k_1} + 1\right) - 1 \end{aligned} \tag{8.3}$$

where $\gamma \approx 0.5772$ is the Euler-Mascheroni constant and Γ is the gamma function.

In other situations using NMF with probability distributions the gamma distribution has been used [126] (see Chapters 3 and 4. The reason that we have chosen the Weibull distribution is that while it is possible to apply variations on the reparameterization trick to the gamma function [35] it is simpler to use the Weibull distribution and use inverse transform sampling. To sample from the Weibull distribution all we need is the inverse cumulative function, $C^{-1}(\epsilon) = \lambda(-\ln(\epsilon))^{1/k}$. We generate a uniform random variable ϵ at an input node and then sample from the Weibull distribution with λ and k by $C^{-1}(\epsilon)$. So, refering to Figure 8.2, we now have $\epsilon \sim \mathcal{U}(0,1)$ and each of the dimensions of $\mathbf{z}$ are found by $z_i = C_i^{-1}(\epsilon_i)$.

It is worth considering exactly how and where PAE-NMF differs from standard NMF. In Figure 8.1 we see that, in terms of NMF, it is fairly unimportant what occurs before the constriction layer in terms of the outputs of interest ($\mathbf{W}$ and $\mathbf{H}$). The aim of the design before that part is to allow the network to find the best possible representation that gets the lowest value of the objective function. There are effectively two parts which make this a probabilistic method: the choice of objective function and the fact that we sample from the distribution. Without those two parts the link between the distribution

parameters and $\mathbf{h}$ would just be a non-linear function which might do better or worse than any other choice but would not make this a probabilistic method.

8.2.4 Methodology

We now have the structure of our network in Figure 8.2 and the distribution (Weibull) that we are using. The basic flow through the network with one hidden layer, for an input datapoint $\mathbf{v}$, looks like:

$$\boldsymbol{\lambda} = f(\mathbf{W}_{\boldsymbol{\lambda}}\mathbf{v}), \quad \boldsymbol{k} = g(\mathbf{W}_{\mathbf{k}}\mathbf{v}), \quad \mathbf{h} = C^{-1}_{\boldsymbol{\lambda},\mathbf{k}}(\boldsymbol{\epsilon}), \quad \hat{\mathbf{v}} = \mathbf{W}_f\mathbf{h} \tag{8.4}$$

where f and g are non-linear functions that work element-wise. The inverse cumulative function, $C^{-1}_{\boldsymbol{\lambda},\mathbf{k}}$, also works element-wise.

There are a range of choices to make for this network, to demonstrate the value of our technique we have kept the choices as simple as possible. We use rectified linear units (ReLU) as the activation function as these are probably the most popular current method [98]. To keep the network simple we have only used one hidden layer. We update the weights and biases using gradient descent. We keep the $\mathbf{W}_f$ values non-negative by setting any negative terms to zero after updating the weights. We use the whole data-set in each batch, the same as is used in standard NMF. We use a prior of $k = 1$ and $\lambda = 1$. We calculate the subspace size individually for each data-set using the method of [126] or pick values manually depending on what we are trying to demonstrate. The learning rates are chosen by trial and error.

To demonstrate the use of our technique we have tested it on the same three heterogeneous data-sets as we have used throughout this thesis, they are displayed in Table 8.1. The faces data-set, are a group of 19×19 grey-scale images of faces. The Genes data-set is the 5000 gene expressions of 38 samples from a leukaemia data-set and the FTSE 100 data is the share price of 94 different companies over 1305 days.

8.3 Results and Discussion

First we demonstrate that PAE-NMF will produce a reasonable recreation of the original data. We would expect the accuracy of the reconstruction to be worse than for standard NMF because we impose the KL divergence term and we sample from the distribution

TABLE 8.1: Data-sets names, the type of data, the number of dimensions, m, number of data-points, n and the source of the data.

Name	**Type**	m	n	**Source**
Faces	Image	361	2429	http://cbcl.mit.edu/software-datasets /FaceData2.html
Genes	Biological	5000	38	http://www.broadinstitute.org/cgi-bin /cancer/datasets.cgi
FTSE 100	Financial	1305	94	Bloomberg information terminal

rather than taking the latent outputs deterministically. These two impositions should help to reduce overfitting resulting in the higher reconstruction error.

In Figure 8.3 we show recreations of the original data for the three data-sets from Table 8.1. The faces plots show five original images chosen at random above the recreated versions (with $r = 81$). The bottom left plot shows nine stocks from the FTSE 100 with the original data as a black dashed line and the recreated results as a red dotted line ($r = 9$). The results here follow the trend well and appear to be ignoring some of the noise in the data. In the bottom right plot we show 1000 elements of the recreated matrix versus the equivalent elements ($r = 3$). There is significant noise in this data, but there is also a clear trend. The black line shows what the results would be if they were recreated perfectly.

Secondly, we want to look at the $\mathbf{W}_f$ matrices (the weights of the final layer). In NMF the columns of $\mathbf{W}$ represent the dimensions of the new subspace we are projecting onto. In many circumstances we hope these will produce interpretable results, which is one of the key features of NMF. In Figure 8.4 we demonstrate the $\mathbf{W}_f$ matrices for the faces data-set, where each of the 81 columns of $\mathbf{W}_f$ has been converted into a 19×19 image (left) and the FTSE 100 data-set (right) where the nine columns are shown. We can see that the weights of the final layer does do what we expect in that these results are similar to those found using standard NMF [72]. The faces data-set produces a representation which can be considered to be parts of a face. The FTSE 100 data-set can be viewed as showing certain trends in the stock market.

We now want to consider empirically the effect of the sampling and KL divergence term during the training process. In Figure 8.5 we show the distributions of the latent space of one randomly chosen datapoint from the faces data-set. We use $r = 9$ so that the distributions are easier to inspect. The left and right set of plots show results with and without the KL divergence term respectively. The black and blue dashed lines show samples extracted deterministically from the median (the median due to the ease of extracting the median from the cumulative distribution) of the distributions and

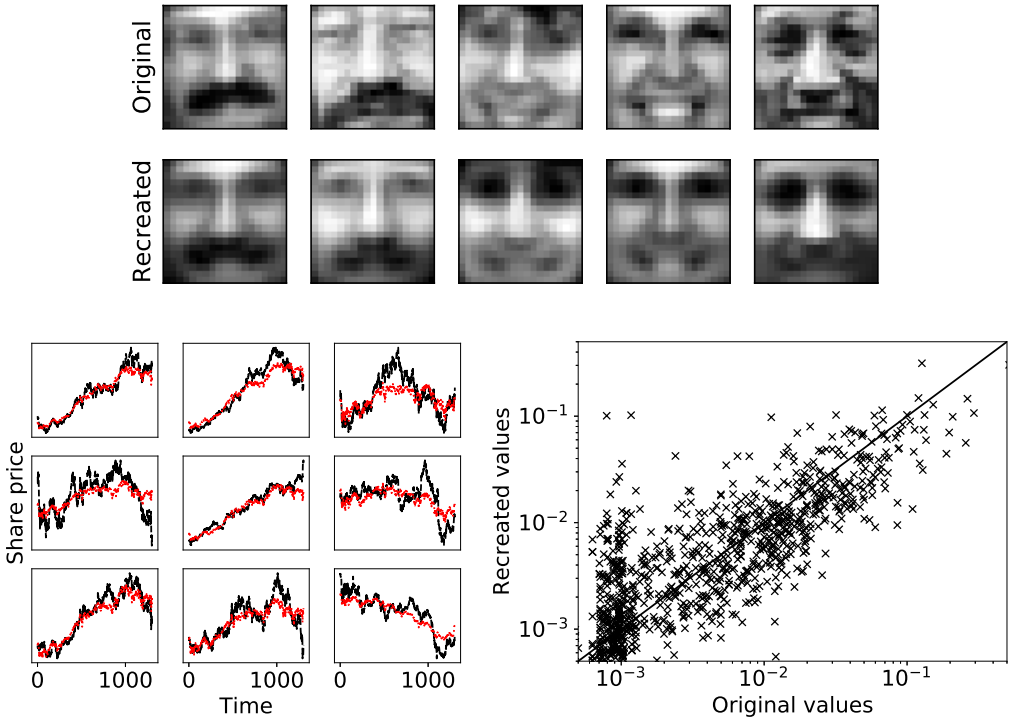


FIGURE 8.3: (Top) Top five are original faces with the equivalent recreated faces below. (Bottom left) Nine stocks with original values (black dashed) and recreated values (red dotted). (Bottom right) 1000 recreated elements plotted against the equivalent original.

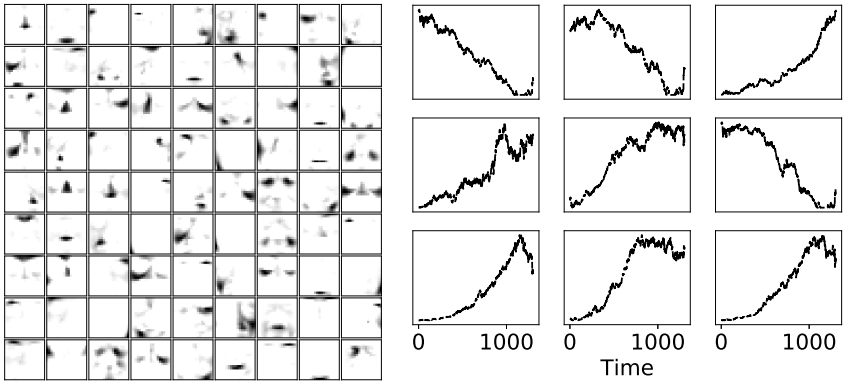


FIGURE 8.4: (Left) Each small image is one of the 81 reshaped columns of $\mathbf{W}_f$ for the faces data-set. The features we see are very similar to what you get in standard NMF. (Right) Each plot is a column of $\mathbf{W}_f$ for the FTSE 100 data-set.

sampled from the distribution respectively. The inclusion of the KL divergence term has several effects. First, it reduces the scale of the distributions so that the values assigned to the distributions are significantly lower in the left hand plot. This has the effect of making the distributions closer together in space. The imposition of randomness into the PAE-NMF through the uniform random variable $\boldsymbol{\epsilon}$ has the effect of reducing the variance of the distributions. When there is a KL divergence term we can see that the

distributions follow fairly close to the prior. However, once the stochasticity is added this is no longer tenable as the wide spread of data we are sampling from becomes harder to learn effectively from. When there is no KL divergence term the effect is to tighten up the distribution so that the spread is as small as possible. This then means we are approaching a point, which in fact would return us towards standard NMF. The requirement for both the KL divergence term and the stochasticity during training means that we do get a proper distribution and the results are prevented from simply reducing towards a standard NMF formulation.

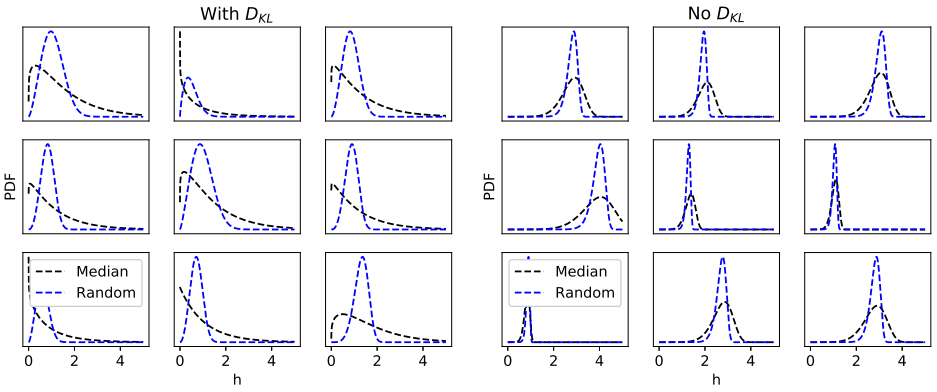


FIGURE 8.5: The distributions of **h** for one data-point from the faces data-set with $r = 9$. (Left) These plots show the distributions when we include the D_{KL} term. (Right) The D_{KL} term is not applied during training. The black dashed line shows results when we train the network deterministically using the median value of the distribution and the blue line is when we trained with random samples.

A potentially valuable effect of PAE-NMF is that we would expect to see similar distributions for similar data-points. In Figure 8.6 we can see the distributions of the **h** vectors for two pairs of faces, each pair is similar to the other one in the pair and very different from the other pair. Next to them we plot the distributions. The distributions for the two faces on the left are the black dashed lines and the distributions for the two faces on the right are given by the red dotted lines. There is very clear similarity in distribution between the similar faces, and significant differences between the dissimilar pairs.

Finally, we want to discuss the generation of new data using this model. We show results for the faces and FTSE 100 data-sets in Figure 8.7. For the faces we use a low $r = 9$ value and show four random faces (left column), with the median being the **h** drawn from the middle of the inverse cumulative distribution. The three image plots on the right are then drawn randomly from the distributions. Four stocks from the FTSE 100 data are shown on the right of the figure. The solid lines are the original data with the dotted lines representing sampled data. This ability to directly sample from the

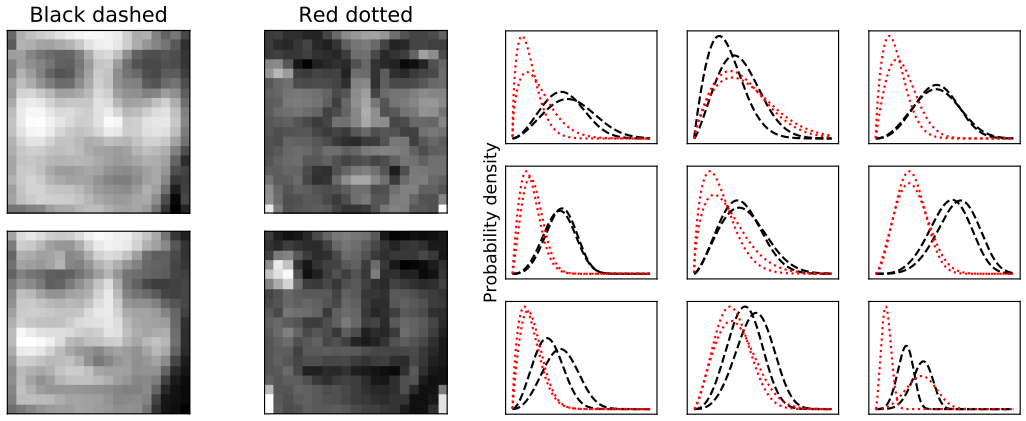


FIGURE 8.6: The left images (top and bottom) are similar to one another and we plot their distributions as black dashed lines in the plots to the right. The right images are very different to the left, we plot their distributions as red dotted lines. The similar images have very similar distributions for these data-points.

distributions and produce plausible new data-points is one of the most useful features of using PAE-NMF over standard NMF.

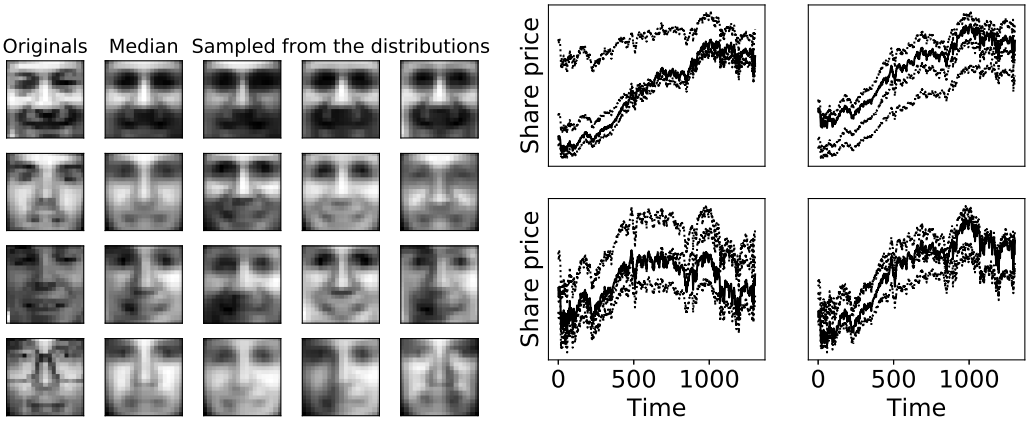


FIGURE 8.7: (Left) Sampling from the distributions of the faces data-set with $r = 9$. Four original faces are on the left column, with faces drawn deterministically from the centre of the distribution next to them and three sampled faces along the next three columns. (Right) Sampling from the FTSE 100 data-set with $r = 9$ for four different stocks. The solid black line is the real data and the dotted lines show three sampled versions.

8.4 Summary

We have demonstrated a novel method of probabilistic NMF using a variational autoencoder. This model extends NMF by providing us with uncertainties on our $\mathbf{h}$ vectors and allowing us to sample new data. The advantage over a VAE is that we should see

improved interpretability due to the sparse and parts based nature of the representation formed, especially in that we can interpret the $\mathbf{W}_f$ layer as the dimensions of the projected subspace.

Our method extracts the useful information from the data without over-fitting due to the combination of the log-probability with the KL divergence. While the log-probability works to minimise the error, the KL divergence term acts to prevent the model over-fitting by forcing the distribution $q_\phi(\mathbf{h}|\mathbf{x})$ to remain close to its prior distribution. This provides a principled regularisation mechanism. Other alternatives for probabilistic NMF still require the choice of an appropriate prior. This could be done using model selection through the evidence term, but this would require computing the full posterior, which is likely to require techniques such as Markov Chain Monte Carlo and could be prohibitively time consuming.

Another advantage of this approach is that as the objective function measures the description length, we could use it to select the appropriate size for the latent space. This would provide an alternative to the approach demonstrated in Chapter 3. However, as the distribution $q_\phi(\mathbf{h}|\mathbf{x})$ provides a self-regularisation the size of the latent space is likely to be less critical.

Chapter 9

Conclusions and Future Work

9.1 Conclusions

This thesis addresses the important problems of model selection, inclusion of external data, use of neural network approaches and probabilistic formulation of the linear dimensionality technique, non-negative matrix factorisation, which has found a wide range of application in data analysis. In this section we bring together the key problems we considered along with the methods used to solve them and how effectively they worked.

In Chapter 3 we demonstrated a method which provides a solution for one of the issues facing NMF [42], the selection of the subspace size r. By applying a principled trade-off between complexity and the accuracy of the representation using MDL we developed a technique to assess the optimal size of the subspace. We showed that this method finds the correct subspace size on synthetic data and on real data we produced empirical evidence that several desirable requirements are fulfilled. The selection of the subspace size is important because with a good choice of r there should be less over, or under, -fitting of the data.

Another potential problem of overfitting in NMF is that the factorised matrices themselves may become highly complex and end up modelling noise. While this effect can be countered by applying regularisation, it is a fairly unprincipled approach and still requires choices about how much regularisation, and what type, to apply. In Chapter 4 we proposed using a similar MDL approach as in Chapter 3 but with MDL applied as the objective function. The minimisation of this MDL objective function should allow for an automatic and principled trade-off between the complexity of the model and the accuracy of the reconstruction. We produced a method to conduct NMF using an MDL objective function and demonstrated promising results on real and semi-synthetic data.

The second theme we examined was how to include fixed external data into the NMF formulation (Chapter 5). We can consider this external data to be prior knowledge we are trying to inject into our problem domain. To solve this issue we created a novel method, called XNMF. We designed a new objective function to include the external data, created a new extended version of the multiplicative update method [72] and proved that it was guaranteed to monotonically reduce the new objective function. We demonstrated the potential of this method to produce an improved representation on FTSE 100 data by showing that it finds a lower error and can build tighter clusters than standard NMF.

Building upon XNMF, in Chapter 6, we demonstrated that the method can also be applied to biological data and has the potential to provide additional information and representations. We considered spatial protoemics data with external information coming from a PPI network. We focussed on representation and clustering/classification and demonstrated that there were some advantages in using XNMF especially in identifying outlier proteins (that may exist in multiple compartments) and in classification. Perhaps more importantly we demonstrated that XNMF can integrate biological data which opens up a wide realm of possible valuable future directions of research.

In Chapter 7 we explored the use of autoencoders to perform NMF (AE-NMF). We laid out some of the reasons why using an autoencoder to perform NMF might be a fruitful avenue of research and outlined the constraints that have to be imposed on an autoencoder so that it provides the benefits of NMF. We demonstrated that an autoenoder can reproduce similar results to standard NMF which is a necessary (but not sufficient) condition for it to be useful in the NMF domain. We then looked at how using AE-NMF might provide some advantages over standard NMF. We showed that AE-NMF can perform methods such as XNMF and that we could use AE-NMF to test different objective functions, such as the risk averting error [83].

One of the reasons for using an autoencoder to perform NMF is it opens up a range of other methods that can be applied to NMF from the neural network world. In Chapter 8 we investigated how one of these recent techiques, the variational autoencoder, could be applied to perform a probabilistic form of NMF. This probabilistic version has several advantages over a standard deterministic NMF including providing a level of uncertainty on the representation of each data-point in the latent space and allowing us to sample new data-points from the distribution. We produced a design for this network and a distribution (the Weibull) which enables us to perform probabilistic NMF. We then tested the method on three heterogenous data-sets and demonstrated that it produces a regularised version of NMF and that we can successfully draw new data-points from the distributions found.

9.2 Future Work

It might be informative to reproduce our method of rank selection using MDL with other distributions such as the Weibull distribution used in Chapter 8. We would then be able to compare the results found with the different distributions, if they are similar then it adds weight to the method, if different then it casts doubt on the quality of the technique. It would also be interesting to investigate empirically when and how the optimal choice of r changes between the histogram method and the distribution method for different data-sets. This would be of particular interest to see what happens when the distributions fit the data poorly and how quickly the optimal values of r change. This would give us some sense of when the method begins to fail and how robust it is to data which does not fit the distributions well.

Similarly it might be useful to test other non-negative distributions when using MDL as the objective function. It would also be valuable to find a computationally efficient method to perform discrete optimisation using the histograms alone as this would enable a direct comparison between the distribution and histogram methods. This comparison would be especially useful as it might give us some insight into how well the method is working, if the results from both are similar we could be more confident that this method has value. There would also be a considerable advantage to finding a better method for minimising the objective function of the parametric version than our gradient descent method as we took a very simple approach which could doubtlessly be significantly improved. One option would be to use the AE-NMF formulation and utilise improved optimisation schemes as there are a suite of built in methods for use with neural networks.

A potentially interesting future extension of XNMF would be to relax the non-negativity constraint on the $\mathbf{H}_2$ matrix. One method would be to allow any element of $\mathbf{H}_2$ to be negative as well as positive, which would be straight-forward to do in the AE-XNMF formulation, which is, incidently, another potential advantage of the autoencoder method of performing NMF. Another technique would be to have a forced negative and forced positive matrix, such that XNMF would become: $\mathbf{V} \approx \mathbf{W}_1\mathbf{H}_1 + \mathbf{W}_2\mathbf{H}_2 - \mathbf{W}_2\mathbf{H}_3$, where all the matrices are forced to be non-negative so that all elements of $\mathbf{H}_3$ are effectively negative. We would use the same $\mathbf{W}_2$ matrix for the final two matrix multiplications, again something that would be easy to do with the AE-NMF. This would maintain the benefits of XNMF in terms of parts based and sparse solutions but would enable negative values to be used. Another option would be to allow negative external data, i.e. allowing $\mathbf{W}_2$ to be negative. While this would allow a broader range of external data to be used it might reduce to interpretability of the $\mathbf{W}_1$ and $\mathbf{H}_1$ matrices, but it would be worth investigating.

One of the issues with performing XNMF is how to select and preprocess the exogeneous data. In Chapter 5 we briefly mentioned that there are many other types of macro-economic data that could be considered, such as values of indices from other stock markets. Also the questions of how to normalise these external values and how to manage the differences in frequency of the data are still open.

Continuing with XNMF, there may be other data-sets that might be usefully analysed using this method. We investigated one small problem in the biological world in Chapter 6 but there will be many other open problems in that highly complex and integrated domain where both reducing the complexity of data while integrating multiple data-sets would be valuable. Also the best way to create and normalise the external data, $\mathbf{W}_2$, would be useful to investigate.

It is also possible that XNMF could be accomplished by concatenating the $\mathbf{W}_1$ and $\mathbf{W}_2$ matrices. Standard NMF would then be performed on the combined matrix but the elements associated with $\mathbf{W}_2$ would not be updated. This may produce the same results as our XNMF method but that would need to be checked and analysed as would a proof of monotonic reduction of the objective function.

There is a potentially wide range of future work that could be done for AE-NMF (Chapter 7). We have already discussed how the AE-NMF formulation could be useful for both extensions to XNMF and with the MDL objective function. It would be interesting to take one data-set of interest and focus on using all the methods available to produce the best representation possible, seeing what changes might help in a systematic way and if this can result in a better set of results compared to NMF. We would also be interested in making a comparison of computational speed and complexity for different forms of AE-NMF. If AE-NMF could be made to run faster than NMF, or with less memory requirement, then it would be of considerable value, especially if it would enable NMF to be applied to very large data-sets.

One way that AE-NMF might be of value is to attempt to measure the ability of different NMF methods to generalise. As the $\mathbf{W}$ matrix (weights of the final layer) is produced as part of the AE-NMF process we only need to find the weights of the previous layers to be able to use AE-NMF to get some estimate of the ability of the method to generalise. Either we could use the network without any alteration, or we could update the earlier layer weights which produce the $\mathbf{H}$ matrix. Either method could give us some understanding of how well the NMF method could be applied to unseen matrices.

There is considerable scope for further investigations using PAE-NMF (Chapter 8) or in combining PAE-NMF with other methods. To give some sense of the robustness of PAE-NMF, using the gamma distribution rather than the Weibull would be interesting.

We would hope that it does not make much difference what the specific choice of distribution is, if we find similar results with different distributions it implies this is the case, conversely, if we found very different results it would call into question the method itself.

A particularly interesting future technique would be to perform a form of probabilistic XNMF, called PAE-XNMF, which is a merger of the work from Chapters 5 and 8. This would give some sense of the uncertainty and would allow for the extraction of samples from the distribution while including external data. We have created this method and implemented it but have not yet completed the analysis which we are doing with a colleague and did not feel it was ready to include in this thesis.

One frustration in developing new NMF methods is the lack of ground truth or a standard test suite to make comparisons between techniques easier. This is intrinsically difficult because what is considered a good solution might be dependent on the application. Nevertheless, the creation and curation of a test suite for NMF may be necessary to enable better comparisons and would be a valuable piece of further work in this field.

Appendix A

Cross Validation for Rank Selection in NMF

A.1 Cross-Validation

A.1.1 Theory and Methodology

Many supervised learning techniques use cross-validation to test their models and prevent over-fitting. In cross-validation the model is trained on "held-in" data and tested on "held-out" data. In supervised classification learning the result desired is the label of an unknown data point. In NMF the desired outcome is both the $\mathbf{W}$ matrix, which is produced from all the data points and the column of $\mathbf{H}$ which represents the unknown data point in the new space. If there is a reasonably large number of data points then $\mathbf{W}$ can be found using the held-in data with reasonable confidence that it would be similar to that $\mathbf{W}$ if all the data was used, similarly to the production of the model in the supervised case. However, in the supervised case the model would then be applied to the test data and the quality of the outcome assessed. With NMF there is no straight-forward method for assessing what the missing columns of $\mathbf{H}$ would be to then apply and compare to the associated columns of the original data matrix $\mathbf{V}$.

Kanagal and Sindhwani [63] assessed three different types of cross-validation. The first is a standard type of cross-validation. The second is based on work by Owen and Perry [104] and the third is a method they introduce themselves.

1) **Basic holdout mechanism**. The input matrix is split up into k training and testing sets, as for standard k-fold cross-validation: $\mathbf{V} = [\mathbf{V}_{\text{train}}, \mathbf{V}_{\text{test}}]$. Then $\mathbf{W}_{\text{train}}$ and $\mathbf{H}_{\text{train}}$ are found from the training data for each fold. The objective function for the

training data: $F_{\text{train}} = ||\mathbf{V}_{\text{train}} - \mathbf{W}_{\text{train}}\mathbf{H}_{\text{train}}||^2_{\text{Fro}}$ is the analogue for training error, or classification error, in supervised machine learning and will decrease as r increases. The objective function for the held-out set is: $F_{\text{test}} = ||\mathbf{V}_{\text{test}} - \mathbf{W}_{\text{train}}\mathbf{H}_{\text{test}}||^2_{\text{Fro}}$. So $\mathbf{W}_{\text{train}}$ is reused but $\mathbf{H}_{\text{test}}$ must be found using $\mathbf{V}_{\text{test}}$. A reduced version of Algorithm 1 which removes the third step which updates $\mathbf{W}$ is applied to find the $\mathbf{H}$ terms which best fit the objective function for the held-out set. The obvious problem with this method is that $\mathbf{H}_{\text{test}}$ is trained using $\mathbf{V}_{\text{test}}$, unlike in cross-validation in supervised learning where the held-out testing set is not trained in the presence of the relevant labels. Here if the NMF algorithm works effectively the mechanism is highly susceptible to over-fitting.

2) **Bi-Cross-Validation** [104]

Here the dimensions (rows of the matrix) are divided into h groups and columns into l groups, combinations of which give each fold, for a total of $(h \times l)$ folds. One of the folds is the hold-out group while the others are available for training. This is equivalent to splitting the matrix $\mathbf{V}$ into four smaller matrices: $\mathbf{V} = \begin{pmatrix} \mathbf{A} & \mathbf{B} \\ \mathbf{C} & \mathbf{D} \end{pmatrix}$ where $\mathbf{A}$ contains the holdout elements for testing while $\mathbf{B}$, $\mathbf{C}$ and $\mathbf{D}$ contain the hold-in elements available for training. The aim of this type of cross-validation is to find $F_A = ||\mathbf{A} - \mathbf{W}_A\mathbf{H}_A||^2_{\text{Fro}}$ which is the objective function for the held-out elements. Their technique uses $\mathbf{B}$, $\mathbf{C}$ and $\mathbf{D}$ together to find $\mathbf{W}_A$ and $\mathbf{H}_A$ without using $\mathbf{A}$.

First $\mathbf{W}_D$ and $\mathbf{H}_D$ are found using NMF with the objective function $F_D = ||\mathbf{D} - \mathbf{W}_D\mathbf{H}_D||^2_{\text{Fro}}$. Next $\mathbf{H}_C = \text{argmin}_H||\mathbf{C} - \mathbf{W}_D\mathbf{H}||^2_{\text{Fro}}$ is calculated. This $\mathbf{H}_C$ is the best (within the limits of NMF) choice to minimise the objective function when using $\mathbf{W}_D$. Essentially this step does the same as the basic hold-out mechanism discussed above but for the input matrix $\mathbf{V}_{CD} = [\mathbf{C}, \mathbf{D}]$. The next step is to assign $\mathbf{H}_A = \mathbf{H}_C$ so that now we have an approximation for $\mathbf{H}_A$. The reasoning for this step is that if we consider $\mathbf{V}_{AC} = \left[\begin{smallmatrix} \mathbf{A} \\ \mathbf{C} \end{smallmatrix}\right]$ it is clear that any $\mathbf{H}$ that approximates for $\mathbf{C}$ in the new space must also approximate $\mathbf{A}$ because $\mathbf{H}$ provides coefficients for the data point in the new space, and $\mathbf{H} \in \mathcal{R}^{(r \times n)}$ which has no dependence on the dimension m which is different between $\mathbf{A}$ and $\mathbf{C}$. So the assumption here is that as $\mathbf{A}$ and $\mathbf{C}$ are from the same data points they will share the same $\mathbf{H}$, or that $\mathbf{H}_C$ is a reasonable approximation for $\mathbf{H}_A$.

Next $\mathbf{W}_A$ is found in an analogous manner. So $\mathbf{W}_B = \text{argmin}_W||\mathbf{B} - \mathbf{W}\mathbf{H}_D||$ and then $\mathbf{W}_A$ is set to $\mathbf{W}_B$. The logic is similar as for finding $\mathbf{H}_A$: $\mathbf{D}$ and $\mathbf{B}$ must share the same $\mathbf{H}$ but different $\mathbf{W}$. So $\mathbf{W}_B$ is optimised when using $\mathbf{H}_D$. If $\mathbf{V}_{AB} = [\mathbf{A}, \mathbf{B}]$ the $\mathbf{A}$ and $\mathbf{B}$ must share a $\mathbf{W}$, as $\mathbf{W}$ represents the new subspace which must be shared by $\mathbf{A}$ and $\mathbf{B}$. So we set $\mathbf{W}_A = \mathbf{W}_B$.

Now we have approximations of $\mathbf{W}_A$ and $\mathbf{H}_A$ and can calculate the held-out error F_A which is the analogue to the testing error, or generalisation error, in supervised learning.

At no point has $\mathbf{W}_A$ or $\mathbf{H}_A$ been trained on $\mathbf{A}$ so the error should not over-fit. This method is a form of data-imputation.

3) **Cross-validation using a weighted NMF** [63]. A weighted NMF (introduced in Section 2.6) minimises the objective function:

$$F = \text{argmin}_{\mathbf{W}\geq 0,\mathbf{H}\geq 0}||\mathbf{S}\odot(\mathbf{V}-\mathbf{WH})||^2 \tag{A.1}$$

where $\odot$ is the Hadamard product and $\mathbf{S}$ is a matrix of weights which can emphasise the importance of parts of the data matrix over other parts. In this type of cross-validation $\mathbf{S}$ is a binary matrix where zeros correspond to the held-out elements and ones to the held-in elements. Simple modifications to the MU rules produce update rules for weighted NMF [53, 91]:

$$W = W\odot\frac{(\sqrt{S}\odot V)H^T}{\left(\sqrt{S}\odot(WH)\right)H^T} \qquad H = H\odot\frac{W^T(\sqrt{S}\odot V)}{W^T\left(\sqrt{S}\odot(WH)\right)}. \tag{A.2}$$

The left out cells can be either random (Wold holdouts) or more systematic as in the Bi-Cross-Validation technique (Gabriel hold-outs). The held-in (classification) error is then the error on the elements with a weight of one and the held-out (generalisation) error is the error on the cells with a weighting of zero.

A.1.2 Results and Analysis

The three types of cross validation were implemented using modified versions of MU and Hoyer's NMF code. Gabriel hold-outs were used for the third type to make it a fair comparison with the other two methods. These were then applied to the Faces dataset with $m = 361$ and an unknown "true" r: there is no ground truth here.

The results in Figure A.1 demonstrate the unsatisfactory performance of the three methods. The error bars record the standard deviation of the different folds. The first cross-validation method (red line) appears to show over-fitting with results considerably below the other methods. It also shows no turning point or plateau and therefore no clear choice for r. The second type of cross-validation (green line) also fails to show any turning point. The third type of cross-validation was run for nine (in total) and four folds as done in the paper by Kanagal [63]. The third method with nine folds shows no obvious turning points but with four folds does at around $r = 100$. The turning point is not ideal at the error falls again as r becomes large. While turning points do appear as the number of folds decreases they can also move around for different numbers of folds.

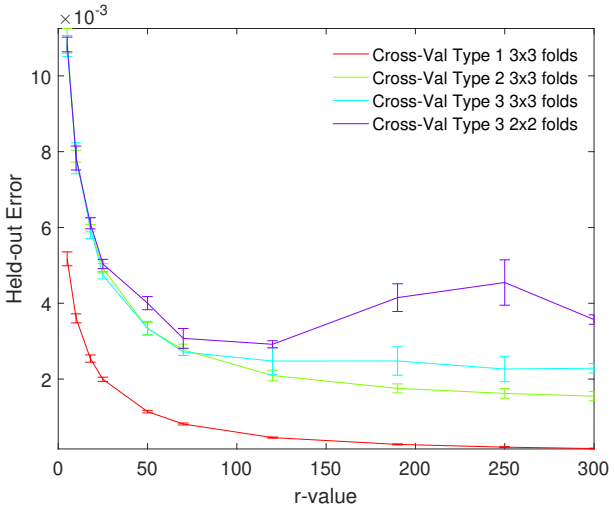


FIGURE A.1: The held-out Gabriel error for the three cross-validation methods with a total of nine folds, with the addition of a fourth plot for four folds for the third technique. These cross-validation methods do not show clear turning points which would give a reasonable choice for r.

This seems to shift the selection of r onto the selection of fold number, which is not obviously a significant improvement. The cross-validation techniques produce interesting results which may be of value to assessing sensible options for r but cannot, on their own, provide a strong guide for how to choose r.

Appendix B

Derivation of Equations for MDL as an Objective Function

B.1 MDL as an Objective Function using the Distributions

We want to be able to minimise the objective function, f, with respect to $\mathbf{W}$ and $\mathbf{H}$ therefore we need to be able to differentiate f with respect to $\mathbf{W}$ and $\mathbf{H}$. We can then either follow some form of gradient descent.

The probability densities $\rho(\mathbf{X})$ are easy to calculate once you have the parameters (α_W, α_H, β_W, β_H, σ, μ). To get these parameters in the first place requires computer power. But more importantly it requires all the values to be binned and then a distribution set to the binned results. Once we have that original distribution we can actually go back to using the X values as non-binned, if we are happy to use the approximation that the data points are no longer in their bin centres, we can then consider the probability densities to be continuous variables. We can then differentiate f and follow some form of gradient descent.

B.1.1 Updates of the objective function

So start with our objective function:

$$\begin{aligned} f &= L_E + L_W + L_H \\ &= -\left[\sum_{b=1}^{B} n_b^{(E)} \log_2 P(E_b) + \sum_{b=1}^{B} n_b^{(W)} \log_2 P(W_b) + \sum_{b=1}^{B} n_b^{(H)} \log_2 P(H_b)\right]. \end{aligned} \tag{B.1}$$

If we are considering distributions we may be able to make this simpler by removing the bins and instead just using values. So Eq (B.1) becomes:

$$f = -\left[\sum_{i=1}^{m}\sum_{j=1}^{n}\log_2 P(E_{ij}) + \sum_{i=1}^{m}\sum_{j=1}^{r}\log_2 P(W_{ij}) + \sum_{i=1}^{r}\sum_{j=1}^{n}\log_2 P(H_{ij})\right]. \tag{B.2}$$

The errors will be modelled with a Gaussian distribution while the **W**s and **H**s with a gamma distribution. Our probabilities for the gamma distribution look like:

$$P(W_{il}) = \frac{\Delta_W \beta^\alpha}{\Gamma(\alpha)} W_{il}^{\alpha-1} e^{-\beta W_{il}} \tag{B.3}$$

where Δ_W is the precision required of the data.

We now consider how the change in the element W_{il} effects the objective functions, f. W_{il} will have an effect on L_W and L_E but not on L_H.

First we will consider the effect of changes in W_{il} on L_W. The only part of L_W that will be effected by shifts in W_{il} is $\log_2(P(W_{il}))$. We need to differentiate L_W with respect to W_{il}. Let $u = P(W_{il})$, then $L_W = -\log_2 u$ and $\frac{\partial L_W}{\partial u} = -\frac{1}{u\ln(2)}$ and using the chain rule: $\frac{\partial L_W}{\partial W_{il}} = \frac{\partial L_W}{\partial u}\frac{\partial u}{\partial W_{il}}$ therefore

$$\frac{\partial L_W}{\partial W_{il}} = -\frac{1}{P(W_{il})\ln(2)}\frac{\partial P(W_{il})}{\partial W_{il}}$$

where, if we let $A = \frac{\Delta_W \beta^\alpha}{\Gamma(\alpha)}$,

$$\begin{aligned}\frac{\partial P(W_{il})}{\partial W_{il}} &= \frac{\partial\left[A W_{il}^{\alpha-1} e^{-\beta W il}\right]}{\partial W_{il}} \\ &= A\left[(\alpha-1)W_{il}^{\alpha-2}e^{-\beta W_{il}} - \beta W_{il}^{\alpha-1}e^{-\beta W_{il}}\right] \\ &= A e^{-\beta W il} W_{il}^{\alpha-2}\left[(\alpha-1) - \beta W_{il}\right]\end{aligned}$$

and then, with a bit of cancelling down

$$\frac{\partial L_W}{\partial W_{il}} = -\frac{W_{il}^{-1}\left[(\alpha-1)-\beta W_{il}\right]}{\ln(2)} = -\frac{\left[(\alpha-1)W_{il}^{-1}-\beta\right]}{\ln(2)} \tag{B.4}$$

Note the problem here about what to do as W_{il} tends towards zero. The whole solution will blow up, so need to be very wary of that happening - this is why we keep the elements of $\mathbf{W}$ larger than zero.

The L_W term will also be effected by changes in W_{il}. So we need to find the solution to $\frac{\partial L_E}{\partial W_{i,j}}$.For the Gaussian distribution terms we have

$$P(E_{ij}) = \frac{\Delta_E}{\sqrt{2\sigma^2\pi}} e^{-\frac{(E_{ij}-\mu)^2}{2\sigma^2}}. \tag{B.5}$$

Now we start with:

$$L_E = -\sum_{i=1}^{m}\sum_{j=1}^{n} \log_2(P(E_{ij})).$$

but let us consider just one component, E_{ij}, so that $L_{Eij} = -\log_2(P(E_{ij}))$.

We know that $E_{ij} = V_{ij} - \sum_{k=1}^{r} W_{ik}H_{kj}$ so we want:

$$\frac{\partial L_{Eij}}{\partial W_{il}} = \frac{\partial L_{Eij}}{\partial P(E_{ij})}\frac{\partial P(E_{ij})}{\partial E_{ij}}\frac{\partial E_{ij}}{\partial W_{il}} \tag{B.6}$$

So we can go through each part of Eq (B.6) in turn to find our final answer:

$$\frac{\partial L_{Eij}}{\partial P(E_{ij})} = -\frac{1}{\ln(2)P(E_{ij})} \tag{B.7}$$

and

$$\frac{\partial P(E_{ij})}{\partial E_{ij}} = -\frac{\Delta_E}{\sqrt{2\sigma^2\pi}}\frac{(E_{ij}-\mu)}{\sigma^2} e^{\frac{-(E_{ij}-\mu)^2}{2\sigma^2}} \tag{B.8}$$

and

$$\frac{\partial E_{ij}}{\partial W_{il}} = -H_{lj} \tag{B.9}$$

so finally:

$$\frac{\partial L_{Eij}}{\partial W_{il}} = -\frac{(E_{ij} - \mu)H_{lj}}{\ln(2)\sigma^2}. \tag{B.10}$$

Now for each change in W_{il} there is a change in the whole i^{th} row of $\mathbf{E}$ (but no changes in the other rows) so the final equation is:

$$\frac{\partial L_E}{\partial W_{il}} = -\frac{1}{\ln(2)\sigma^2}\sum_{j=1}^{n}(E_{ij} - \mu)H_{lj}. \tag{B.11}$$

We then can see the final solution:

$$\frac{\partial f}{\partial W_{il}} = -\frac{[(\alpha - 1)W_{il}^{-1} - \beta]}{\ln(2)} - \frac{1}{\ln(2)\sigma^2}\sum_{j=1}^{n}[(E_{ij} - \mu)H_{lj}] \tag{B.12}$$

So the equivalent for the H needs to be found. First let:

$$P(H_{lj}) = \frac{\Delta_H b^a}{\Gamma(a)}H_{lj}^{a-1}e^{-bx} \tag{B.13}$$

where a and b are parameters for the gamma distribution equivalent to α and β for the W terms. The we have:

$$\frac{\partial L_H}{\partial H_{lj}} = -\frac{\left[(a - 1)H_{lj}^{-1} - b\right]}{\ln(2)} \tag{B.14}$$

and

$$\frac{\partial L_E}{\partial H_{lj}} = -\frac{1}{\ln(2)\sigma^2}\sum_{i=1}^{m}(E_{ij}-\mu)W_{il}. \tag{B.15}$$

Then the final solution for the the H terms is:

$$\frac{\partial f}{\partial H_{lj}} = -\frac{[(a-1)H_{lj}^{-1}-b]}{\ln(2)} - \frac{1}{\ln(2)\sigma^2}\sum_{i=1}^{m}\left[(E_{ij}-\mu)W_{il}\right]. \tag{B.16}$$

Appendix C

Proof of Monotonic Reduction of XNMF

C.1 Introduction

We follow almost entirely the scheme of Lee and Seung [73] to prove that our algorithm will monotonically reduce the objective function for our XNMF formulation but extend it to include the extra matrices that we add.

We have an objective function

$$f = \frac{1}{2}||\mathbf{V} - \mathbf{W}_1\mathbf{H}_1 - \mathbf{W}_2\mathbf{H}_2||^2_{\mathrm{Fro}}. \tag{C.1}$$

with three unknown matrices $\mathbf{W}_1$, $\mathbf{H}_1$ and $\mathbf{H}_2$. We propose an extension to the multiplicative update rule that should monotonically reduce the objective function.

Theorem C.1. *The Frobenius norm* $||\boldsymbol{V} - \boldsymbol{W}_1\boldsymbol{H}_1 - \boldsymbol{W}_2\boldsymbol{H}_2||^2_{Fro}$ *will monotonically decreases under the updates:*

$$\boldsymbol{W}_1 \leftarrow \boldsymbol{W}_1 \odot \frac{\left[\boldsymbol{V}\boldsymbol{H}_1^T\right]}{\left[\boldsymbol{W}_1\boldsymbol{H}_1\boldsymbol{H}_1^T + \boldsymbol{W}_2\boldsymbol{H}_2\boldsymbol{H}_1^T\right]}, \tag{C.2}$$

$$\boldsymbol{H}_1 \leftarrow \boldsymbol{H}_1 \odot \frac{\left[\boldsymbol{W}_1^T\boldsymbol{V}\right]}{\left[\boldsymbol{W}_1^T\boldsymbol{W}_1\boldsymbol{H}_1 + \boldsymbol{W}_1^T\boldsymbol{W}_2\boldsymbol{H}_2\right]} \tag{C.3}$$

and

$$\boldsymbol{H}_2 \leftarrow \boldsymbol{H}_2 \odot \frac{\left[\boldsymbol{W}_2^T\boldsymbol{V}\right]}{\left[\boldsymbol{W}_2^T\boldsymbol{W}_2\boldsymbol{H}_2 + \boldsymbol{W}_2^T\boldsymbol{W}_1\boldsymbol{H}_1\right]}. \tag{C.4}$$

where $\odot$ is element-wise multiplication and $\frac{[]}{[]}$ is element-wise division. So for components we would have: $W_{ij}^{(1)} \leftarrow W_{ij} \frac{(\mathbf{V}\mathbf{H}_1^T)_{ij}}{(\mathbf{W}_1\mathbf{H}_1\mathbf{H}_1^T + \mathbf{W}_2\mathbf{H}_2\mathbf{H}_1^T)_{ij}}$. Our aim then is to prove that these multiplicative updates monotonically reduce the objective function.

C.2 Proof of convergence

These proofs are from Lee and Seung [73] with a small extension that we have done. In this proof we consider just changes to $\mathbf{H}_1$, in particular looking at the column vector $\mathbf{h}_1$ and then each component of that vector $h_{(1)a}$. As we are updating $\mathbf{h}_1$ we consider updates $\mathbf{h}_1^t$, where t is the iteration number and $\mathbf{h}_1^t$ is a fixed point of $\mathbf{h}_1$. We also define $\mathbf{v}$ is a column vector of $\mathbf{V}$. A proof for $\mathbf{h}_1$ suffices for all of the rest of the columns that make up $\mathbf{H}$ and for both $\mathbf{W}_1$ and $\mathbf{H}_2$.

Definition C.2. $G(\mathbf{h}, \mathbf{h}')$ is an auxiliary function for $F(\mathbf{h})$ if the conditions

$$G(\mathbf{h}, \mathbf{h}') \geq F(\mathbf{h}), \qquad G(\mathbf{h}, \mathbf{h}) = F(\mathbf{h}) \tag{C.5}$$

are satisfied.

This definition is important because of the next lemma.

Lemma C.3. *If G is an auxiliary function, then F is nonincreasing under the update*

$$\boldsymbol{h}^{t+1} = argmin_{\boldsymbol{h}} G(\boldsymbol{h}, \boldsymbol{h}^t) \tag{C.6}$$

Proof. $F(\mathbf{h}^{t+1}) \leq G(\mathbf{h}^{t+1}, \mathbf{h}^t) \leq G(\mathbf{h}^t, \mathbf{h}^t) = F(\mathbf{h}^t)$ □

Definition C.2 and lemma C.3 are both taken almost exactly from Lee and Seung 2001. I want to expand slightly upon what they specify. The first definition specifies what an auxiliary function in this context is: it is a function that is always larger than F except at point (or points?) $\mathbf{h} = \mathbf{h}'$ where they are equal. The lemma then proves that if a function G is an auxiliary function of F then decreasing G will guarantee not to increase F, which is the monotonic condition we want. An extra note is that, I believe, minimising G may push h past the minimum point of F, but it will still reduce the value of F, never increase it.

They also note that $F(\mathbf{h}^{t+1}) = F(\mathbf{h}^t)$ only if $\mathbf{h}^t$ is a local minimum of $G(\mathbf{h}, \mathbf{h}^t)$. This then also implies that $\nabla F(\mathbf{h}^t) = 0$ therefore we are at a minimum (local) of F. So if we

update G we eventually get to a minimum of F. We note that this is true because if $G(\mathbf{h}^{t+1}, \mathbf{h}^t) = G(\mathbf{h}^t, \mathbf{h}^t) = F(\mathbf{h}^t)$ by definition, then the two functions must be touching at that point. If G is at a turning point and F is not a minima, then G must pass under F at some point as it turns towards the horizontal. This would mean G is less than F at that point, contrary to the definitions, therefore F must also be at a local turning point. Therefore iterating through (C.6) must reduce F to a minimum.

It is from lemma C.4 that our proof diverges from that of Lee and Seung 2001 although it follows exactly the same logic. Our notation is $\mathbf{h}^t_{(1)a}$ is the a^{th} component of the vector $\mathbf{h}_1$, which is a column of the matrix $\mathbf{H}_1$ and is found at iteration t of the algorithm, i.e. it gives the values of $\mathbf{h}_1$ at a particular point. Similar notation applies for the other terms.

Lemma C.4. *If $K(h^t)$ is a diagonal matrix*

$$K_{a,b}(h^t) = \delta_{a,b}(\, \boldsymbol{W}_1^T \boldsymbol{W}_1 \boldsymbol{h}_1^t + \, \boldsymbol{W}_1^T \boldsymbol{W}_2 \boldsymbol{h}_2)_a / h^t_{(1)a} \tag{C.7}$$

then

$$G(\boldsymbol{h}_1, \boldsymbol{h}_1^t) = F(\boldsymbol{h}_1^t) + (\boldsymbol{h}_1 - \boldsymbol{h}_1^t)^T \nabla F(\boldsymbol{h}_1^t) + \frac{1}{2}(\boldsymbol{h}_1 - \boldsymbol{h}_1^t)^T K(\boldsymbol{h}_1^t)(\boldsymbol{h}_1 - \boldsymbol{h}_1^t) \tag{C.8}$$

is an auxiliary function for

$$F(\boldsymbol{h}_1) = \frac{1}{2}\,(\boldsymbol{v} - \, \boldsymbol{W}_1 \boldsymbol{h}_1 - \, \boldsymbol{W}_2 \boldsymbol{h}_2)^T\,(\boldsymbol{v} - \, \boldsymbol{W}_1 \boldsymbol{h}_1 - \, \boldsymbol{W}_2 \boldsymbol{h}_2) \tag{C.9}$$

Proof. Since $G(\mathbf{h}_1,\mathbf{h}_1) = F(\mathbf{h}_1)$ is obvious we need to only show that $G(\mathbf{h}_1, \mathbf{h}_1^t) \geq F(\mathbf{h}_1)$. We begin by using a Taylor expansion to find an exact (function is quadratic so Taylor series should be exact not approximation) formulation for $F(\mathbf{h}_1)$:

$$F(\mathbf{h}_1) = F(\mathbf{h}_1^t) + (\mathbf{h}_1 - \mathbf{h}_1^t)^T \nabla F(\mathbf{h}_1^t) + \frac{1}{2}(\mathbf{h}_1 - \mathbf{h}_1^t)^T (\mathbf{W}_1^T \mathbf{W}_1)(\mathbf{h}_1 - \mathbf{h}_1^t). \tag{C.10}$$

This form comes about because the Taylor expansion of $F(\mathbf{h}_1)$ around $\mathbf{h}_1^t$ is given by:

$$F(\mathbf{h}_1) = F(\mathbf{h}_1^t) + (\mathbf{h}_1 - \mathbf{h}_1^t)^T \nabla_{\mathbf{h}_1} F(\mathbf{h}_1^t) + \frac{1}{2}(\mathbf{h}_1 - \mathbf{h}_1^t)^T \nabla_{\mathbf{h}_1} (\nabla_{\mathbf{h}_1} F(\mathbf{h}_1^t))(\mathbf{h}_1 - \mathbf{h}_1^t). \tag{C.11}$$

Then we have:

$$\begin{aligned} F(\mathbf{h}_1) =& \frac{1}{2}\Big[\mathbf{v}^T\mathbf{v} - \mathbf{v}^T\mathbf{W}_1\mathbf{h}_1 - \mathbf{v}^T\mathbf{W}_2\mathbf{h}_2 \\ & - \mathbf{h}_1^T\mathbf{W}_1^T\mathbf{v} + \mathbf{h}_1^T\mathbf{W}_1^T\mathbf{W}_1\mathbf{h}_1 + \mathbf{h}_1^T\mathbf{W}_1^T\mathbf{W}_2\mathbf{h}_2 \\ & - \mathbf{h}_2^T\mathbf{W}_2^T\mathbf{v} + \mathbf{h}_2^T\mathbf{W}_2^T\mathbf{W}_1\mathbf{h}_1 + \mathbf{h}_2^T\mathbf{W}_2^T\mathbf{W}_2\mathbf{h}_2\Big] \end{aligned} \tag{C.12}$$

which differentiating with respect to $\mathbf{h}_1$ gives:

$$\nabla_{\mathbf{h}_1} F(\mathbf{h}_1) = \Big[\mathbf{W}_1^T\mathbf{W}_1\mathbf{h}_1 + \mathbf{W}_1^T\mathbf{W}_2\mathbf{h}_2 - \mathbf{W}_1^T\mathbf{v}\Big] \tag{C.13}$$

and so

$$\nabla_{\mathbf{h}_1}(\nabla_{\mathbf{h}_1} F(\mathbf{h}_1)) = \mathbf{W}_1^T\mathbf{W}_1 \tag{C.14}$$

which then, combined, gives us Equation (C.10).

Comparing Equations (C.8) and (C.10) gives us the same result as Lee and Seung, that $G(\mathbf{h}_1, \mathbf{h}_1^t) \geq F(\mathbf{h}_1)$ is equivalent to

$$0 \leq (\mathbf{h}_1 - \mathbf{h}_1^t)^T\Big[K(\mathbf{h}_1^t) - \mathbf{W}_1^T\mathbf{W}_1\Big](\mathbf{h}_1 - \mathbf{h}_1^t) \tag{C.15}$$

and as Lee and Seung do we consider the matrix:

$$M_{a,b}(\mathbf{h}_1^t) = h_{(1)a}^t(K(\mathbf{h}_1^t) - \mathbf{W}_1^T\mathbf{W}_1)_{a,b}h_{(1)b}^t.$$

We will have proved that G is an auxiliary function of F if can show M is positive semi-definite. We prove this by following the exact same steps as Lee and Seung but with an extra set of terms which make no difference to the conclusion.

$$
\begin{aligned}
&\boldsymbol{\nu}^T \mathbf{M} \boldsymbol{\nu} \\
&= \sum_{a,b} \nu_a M_{a,b} \nu_b \\
&= \sum_{a,b} \Big[h^t_{(1)a} \Big(\big(\mathbf{W}_1^T \mathbf{W}_1 \mathbf{h}_1^t + \mathbf{W}_1 \mathbf{W}_2 \mathbf{h}_2 \big)_a / h^t_{(1)a} \Big)_{a,b} h^t_{(1)b} \nu_a^2 - \nu_a h^t_{(1)a} (\mathbf{W}_1^T \mathbf{W}_1)_{a,b} h^t_{(1)b} \nu_b \Big] \\
&= \sum_{a,b} \Big[(\mathbf{W}_1^T \mathbf{W}_1)_{a,b} h^t_{(1)a} h^t_{(1)b} \nu_a^2 - \nu_a h^t_{(1)a} (\mathbf{W}_1^T \mathbf{W}_1)_{a,b} h^t_{(1)b} \nu_b + (\mathbf{W}_1^T \mathbf{W}_2)_{a,b} h^t_{(2)b} h^t_{(1)a} \nu_a^2 \Big] \\
&= \sum_{a,b} \Big[(\mathbf{W}_1^T \mathbf{W}_1)_{a,b} h^t_{(1)a} h^t_{(1)b} \big(\frac{1}{2} \nu_a^2 + \frac{1}{2} \nu_b^2 - \nu_a \nu_b \big) + (\mathbf{W}_1^T \mathbf{W}_2)_{a,b} h^t_{(2)b} h^t_{(1)a} \nu_a^2 \big] \\
&= \sum_{a,b} \Big[(\mathbf{W}_1^T \mathbf{W}_1)_{a,b} h^t_{(1)a} h^t_{(1)b} \big(\nu_a - \nu_b \big)^2 + (\mathbf{W}_1^T \mathbf{W}_2)_{a,b} h^t_{(2)b} h^t_{(1)a} \nu_a^2 \Big] \\
&\geq 0
\end{aligned} \tag{C.16}
$$

□

Now the final step is to prove that Theorem C.1 is true.

Proof of Theorem C.1. The proof here is almost identical to Lee and Seung except with a different $K(\mathbf{h}^t)$. Replace $G(\mathbf{h}, \mathbf{h}^t)$ in Equation (C.6) with Equation (C.8):

$$
\begin{aligned}
\nabla_{\mathbf{h}_1} G(\mathbf{h}_1, \mathbf{h}_1^t) &= \nabla_{\mathbf{h}_1} \big[\mathbf{h}_1^T \nabla_{\mathbf{h}_1} F(\mathbf{h}_1^t) \big] && \text{(C.17)} \\
&\quad + \frac{1}{2} \nabla_{\mathbf{h}_1} \big[\mathbf{h}_1^T K(\mathbf{h}_1^t) \mathbf{h}_1 - \mathbf{h}_1^{tT} K(\mathbf{h}_1^t) \mathbf{h}_1 - \mathbf{h}_1^T K(\mathbf{h}_1^t) \mathbf{h}_1^t \big] \\
&= \nabla F(\mathbf{h}_1^t) + K(\mathbf{h}_1^t) \mathbf{h}_1 - K(\mathbf{h}_1^t) \mathbf{h}_1^t = 0 && \text{(C.18)}
\end{aligned}
$$

rearranging gives:

$$
\mathbf{h}_1^{t+1} = \mathbf{h}_1^t - K^{-1}(\mathbf{h}_1^t) \nabla F(\mathbf{h}_1^t). \tag{C.19}
$$

If we then consider components explicitly:

$$
\begin{aligned}
h^{t+1}_{(1)a} =& h^t_{(1)a} - \frac{h^t_{(1)a}}{(\mathbf{W}_1^T \mathbf{W}_1 \mathbf{h}_1 + \mathbf{W}_1^T \mathbf{W}_2 \mathbf{h}_2)_a} (\nabla F)_a \\
=& h^t_{(1)a} - \frac{h^t_{(1)a}}{(\mathbf{W}_1^T \mathbf{W}_1 \mathbf{h}_1 + \mathbf{W}_1^T \mathbf{W}_2 \mathbf{h}_2)_a} \big[\mathbf{W}_1^T \mathbf{W}_1 \mathbf{h}_1 + \mathbf{W}_1^T \mathbf{W}_2 \mathbf{h}_2 - \mathbf{W}_1^T \mathbf{v} \big]_a \\
=& h^t_{(1)a} \frac{(\mathbf{W}_1^T \mathbf{v})_a}{(\mathbf{W}_1^T \mathbf{W}_1 \mathbf{h}_1 + \mathbf{W}_1^T \mathbf{W}_2 \mathbf{h}_2)_a}.
\end{aligned} \tag{C.20}
$$

The same argument is true for $\mathbf{h}_2$ and $\mathbf{W}_1$ and therefore we show that the objective function, F, monotonically decreases under our update rules.

□

Appendix D

A Method of Integrating Spatial Proteomics and Protein-Protein Interaction Network Data

This paper was partially completed as part of my previous MSc in Computer Science at the University of Southampton and partially during this PhD. While it has not been included in this thesis due to both its partial completion during another degree and its lack of synergy with the rest of the thesis it is included in this appendix for completeness.

A Method of Integrating Spatial Proteomics and Protein-Protein Interaction Network Data

Steven Squires, Rob Ewing, Adam Prügel-Bennett, and Mahesan Niranjan

University of Southampton, UK
{ses2g14,rob.ewing,apb,mn}@soton.ac.uk

Abstract. The increase in quantity of spatial proteomics data requires a range of analytical techniques to effectively analyse the data. We provide a method of integrating spatial proteomics data together with protein-protein interaction (PPI) networks to enable the extraction of more information. A strong relationship between spatial proteomics and PPI network data was demonstrated. Then a method of converting the PPI network into vectors using spatial proteomics data was explained which allows the integration of the two datasets. The resulting vectors were tested using machine learning techniques and reasonable predictive accuracy was found.

Keywords: Bioinformatics · Spatial proteomics · Machine learning

1 Introduction

Proteins can only perform their function in direct physical contact with other proteins or parts of the cell, therefore knowing the location of a protein (known as spatial proteomics) can aid in understanding its function. It has also been shown that there is a direct connection between diseases and subcellular protein localisation [1], consequently understanding certain diseases and cellular function depends on a reliable and accurate knowledge of protein localisation.

Protein-protein interactions (PPIs) have been studied for many years due, in part, to their importance in understanding cellular function. Interactions between pairs or groups of proteins, or proteins and other parts of the cell, have significant consequences for cell functionality including links to disease [2]. PPI networks chart these known or predicted interactions.

There is considerable interest in combining multiple sources of high throughput biological measurements. Examples include the integration of spatial and temporal patterns of gene expression [3], combining sequence and secondary structure of proteins [4], and the integrated analysis of the transcriptome and proteome [5].

Spatial proteomics and PPI networks should have significant similarities. A pair of proteins can only physically interact if they are in the same spatial location at the same time, hence we would expect that there would be a link between proteins that interact and those that share a spatial location. In principle, accurate PPI networks might be able to predict which proteins co-localise.

Conversely, spatial proteomics cannot on its own specify whether an interaction exists, but if two proteins are in the same compartment it may be more likely due to the increased likelihood that they share a function. In addition, proteins that never exist in the same spatial location cannot directly interact.

There has been work conducted which uses the PPI networks to make predictions on protein localisation. In particular, a recently published paper used PPI networks together with sequence predictors to classify proteins into spatial locations [6]. In contrast, we use the spatial proteomics profiles themselves and integrate them together with the PPI network data. In doing so we propose to aid the development of analytical tools for the analysis of proteomics data.

Our contributions are, first, to demonstrate the strong relationship between spatial proteomics and PPI network data. We then provide a mechanism to integrate the two datasets along with some useful visualisation techniques. We demonstrate that prediction of spatial localization from fractionation profiles can potentially be enhanced by the inclusion of information taken from PPI interactions and that interactions themselves are somewhat predictable from spatial profiles.

This paper is structured as follows: in Section 2 we discuss the spatial proteomics and PPI datasets along with the methods we use; in Section 3 we demonstrate the strong correlation between PPI and spatial proteomics data, the visualisation benefits of our technique, and the predictive power of the datasets. Finally, in Section 4 we provide a brief discussion of our results.

2 Methods

2.1 Spatial Proteomics and PPI Network Datasets

Spatial proteomics data is obtained from experiments which separate the contents of the cell into fractions and measure the relative abundance of each protein within each fraction. Proteins with a similar profile of fractional abundances are believed to occupy the same spatial location [7]. These proteins are then mapped to organelles by using marker proteins with similar profiles whose location is known [8, 9]. The marker proteins tend to be extracted from literature and need to be highly reliable as they set the mapping from profiles to locations. In this study we used two sets of data obtained from Arabidopsis thaliana [10] and Drosophila melanogaster [11]. Spatial proteomics data is generally of a fairly low dimensionality (usually under 10 fractions) and the datasets contain 689 and 888 proteins for Arabidopsis and Drosophila respectively. We consider two marker sets for each organism, the same as the authors use [10, 11]. The first, which we call the original marker set, are those proteins with known location extracted from literature. There are 27 for Arabidopsis and 55 for Drosophila. The authors then use the spatial proteomics datasets to assign previously unknown proteins to an organelle. We use the same assigned proteins which are named as extended marker sets in this paper.

Two PPI datasets were used: STRING [12] and BIOGRID [13]. These datasets have different methodologies to extract PPIs and present the results differently but results we have gained from both, where comparable, are consistent.

2.2 Combining Spatial Proteomics and PPI Network Data

Spatial proteomics data is produced in a format suitable for applying standard machine learning classification techniques as there is a matrix with m dimensions, n datapoints and each datapoint is within a class. In contrast the PPI data is presented as pairs of known interactions and needs to be converted into a form suitable for applying machine learning methods.

We create a simple fixed dimensional representation to capture information held in interaction networks and apply standard machine learning techniques. We do not consider more sophisticated techniques such as graph kernels [14] in this work because the amount of experimental data relating to subcellular measurements is small. Consider three organelles α, β and γ and a protein of interest, A. The PPI data for A is transformed into a three-dimensional vector by calculating the number of interactions between protein A and the marker proteins within α, β and γ respectively. Protein A is then represented as a vector with each dimension associated with an organelle. We normalise the vector by dividing by the number of proteins within each organelle and then dividing each protein individually by the sum of the vectors across each organelle. We then scale up each protein components so the sum across the PPI vector equals one.

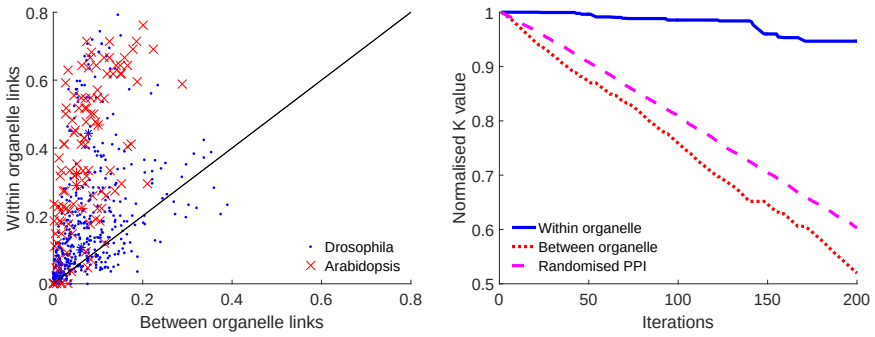
3 Results

3.1 Correlation between Spatial Proteomics and PPI Network Data

For the integration of spatial proteomics and PPI network data to be a valuable analytical technique it should first be demonstrated that they are structurally similar. A key structural similarity is the relationship between the spatial location and the chance of an interaction occurring. In the STRING database approximately 10% and 7% of proteins have links for Drosophila and Arabidopsis respectively; proteins located in the same organelle should, in general, have a higher likelihood of interacting. The ratio of interactions to potential interactions was measured for each protein in the extended marker set. Figure 1(a) shows the fraction of interactions occurring within the same organelle against the fraction to proteins in other organelles. The dots are the proteins for Drosophila and crosses for Arabidopsis, the stars are the averages for each organelle and the black line is the average expected if there was no correlation between the datasets. Any protein above the line has a higher fraction of links to proteins within its organelle than to proteins in other organelles. A majority of proteins have significantly more links within the organelles than between them.

The probability of these results occurring by chance from an uncorrelated PPI network (P-values) was calculated using the hypergeometric distribution.

Within an individual organelle, the probability of the number of interactions observed occurring by chance ranges from $P = 10^{-12}$ to $P = 10^{-404}$. We can reasonably conclude that the marker proteins in the two datasets are correlated.



(a) (b)

Fig. 1. a) The fraction of interactions for each extended marker protein within the organelle is plotted against the fraction of interactions to proteins in different organelles. The black line is the expectation if there is no correlation and the stars show the averages for the organelles. The vast majority of proteins lie above the line. b) The clustering algorithm iteratively removes links, preferentially removing PPI links which connect clusters together rather than those links that are within a cluster. The fall in number of links between proteins within the same organelle (solid line) is slower than those links between proteins in different organelles (dotted line) or a randomised PPI network (dashed line).

If similar clusters of proteins are formed in the PPI networks as in the organelle groupings in spatial proteomics data it would provide further evidence of the correlation between the data types. A clustering algorithm was developed based upon previous work [15, 6] which removes links sequentially based on the structure of the PPI network. The first links to be removed are those considered to be joining separate clusters together. If the structures of the datasets are similar we would expect the number of links between proteins in different organelles to fall off much faster than between proteins within the same organelle. The normalised K value [15] gives a measure of how well connected a group of proteins are, if it falls fast then these proteins are unlikely to be in a cluster together. The average number of links between proteins within the same organelle, K_{in}, and the equivalent number of links between proteins in different organelles, K_{out}, were calculated and normalised. In Figure 1(b) we show that the fall in K_{out} (dotted line) is far faster than for K_{in} (solid line) or for a randomised PPI network (dashed line) demonstrating that the clusters formed in the PPI data are similar to those in spatial proteomics data.

3.2 Visualisation of the Datasets

Visualisation of the spatial proteomics and PPI network data is important for both gaining a qualitative understanding of the quality of the data and for observation of potential patterns.

For spatial proteomics data the ability to classify the proteins depends on differences in the profiles for proteins in different organelles [8]. Part of our contribution is to add the PPI vectors to create vectors with additional dimensions which add extra information to the spatial proteomics data. The average vectors for the original marker proteins for each organelle together with the additional PPI dimensions are in Figure 2(a). The average profile for the PPI dimensions shows peaks at the dimension associated with each organelle (noted by their first letter on the plots). Any protein not showing a significant peak in a PPI dimension while being well classified by the spatial proteomics profile may be of particular biological interest. The organelles shown are the endoplasmic reticulum (ER), mitochondrion (Mito), plasma membrane (PM), Golgi apparatus (Gol) and the vacuole (Vac).

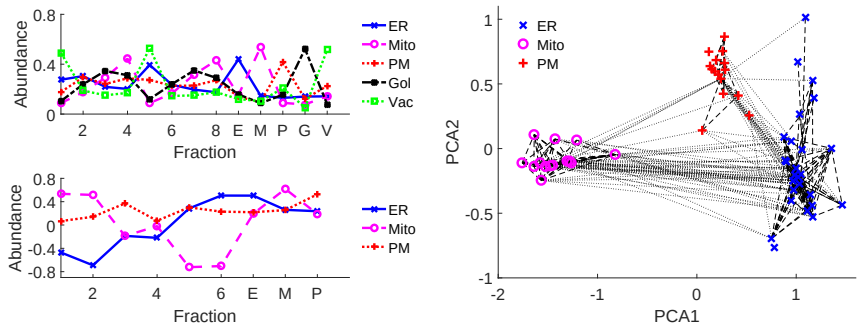
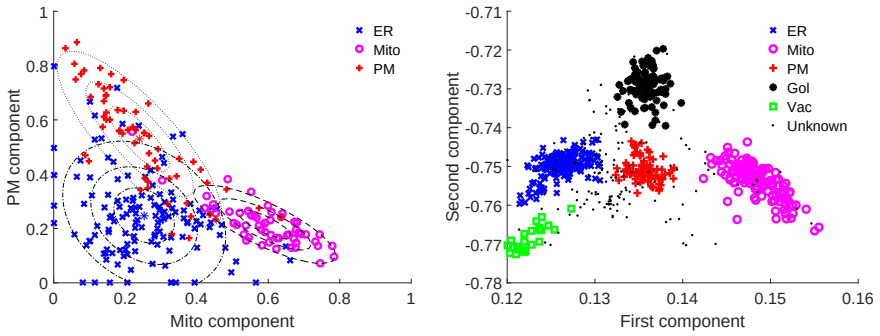


Fig. 2. a) The average profiles for the original marker proteins for Arabidopsis (top) and Drosophila (bottom) are shown. The first eight (for Arabidopsis) and six (Drosophila) fractions are from the spatial proteomics data with the remaining created from the PPI data with labels associated with the first letter of the relevant organelle. b) The original marker protein profiles for Drosophila were projected onto their principal components and PPI network links added. Proteins that have strong connections outside of their organelle can be investigated.

Visualisation of the PPI network data is difficult as the number of links are large. Here, an example of what the network looks like for the small number of original marker proteins for Drosophila is shown in Figure 2(b). The spatial proteomics profiles were projected onto their principal components and known interactions from the STRING database are shown. It may be useful to inspect the links between individual proteins and the markers in this manner to gain insight into how each protein is linked. While there are many links between

proteins in different organelles there are significantly more to proteins within the same organelle. It should also be noted that with only 55 proteins the network plots are already difficult to inspect in this format.

While the PPI networks may be difficult to visualise, the PPI vectors are somewhat easier when two components of the vector are compared. In Figure 3(a) the PM component was plotted against the Mito component for all the Drosophila expanded marker proteins. The PPI vectors were also used to create a Gaussian probabilistic model, with means (shown as stars) and covariances extracted from the vectors. Contours of equal probability are then plotted which aids with understanding the separation and structure of the data. For example, the Mito vectors are much tighter bound than the PM vectors which visually represents that the Mito proteins are more closely connected to other Mito proteins than to proteins in other organelles than the PM proteins are. The structure of the plot is exactly as would be expected with the PM and Mito proteins tending to reside high up the PM and Mito axes respectively and the ER proteins following a fairly isotropic distribution near the origin.



(a) (b)

Fig. 3. a) The PPI vectors for the PM and Mito components for Drosophila marker proteins are plotted. The PM proteins tend to cluster high up the PM axis and the Mito proteins along the Mito axis. The ER proteins are based near the origin. The contours of equal probability are centred on the average of each organelle. b) The application of a Fisher Linear Discriminant to the combined spatial proteomics and PPI vectors for Arabidopsis allows for effective partition of the extended marker proteins into their respective organelles.

We also show the effect of projection using Fisher discriminant directions for the combined vector for Arabidopsis in Figure 3(b). Most of the extended marker proteins are well separated into their organelles, the combined vector is able to effectively partition the proteins into different compartments.

3.3 Predictive Power

We have demonstrated the similarity of the two datasets and some visualisation techniques. Now we will show that there is considerable predictive power in the method of combining datasets

First we show the PPI networks can predict spatial location. As we have converted the PPI network into a vector we can apply standard machine learning techniques. The protein extended marker data was partitioned randomly into two and a support vector machine [16] (SVM) was trained on half of the proteins with the PPI vector as input and the organelles as output classes. The trained SVM was then tested on the remainder of the randomised data. The process, with different random partitions, was then repeated two hundred times. The classification accuracies, sensitivities (true positive rate) and specificities (true negative rates) are shown as boxplots in Figure 4(a). Generally, the SVM trained on the PPI data was able to predict the location of the proteins approximately 70% of the time. The most notable exception was the vacuole where very poor predictions are made. There are only small numbers of vacuole proteins in the extended marker set which is likely to be the reason for the poor predictive ability.

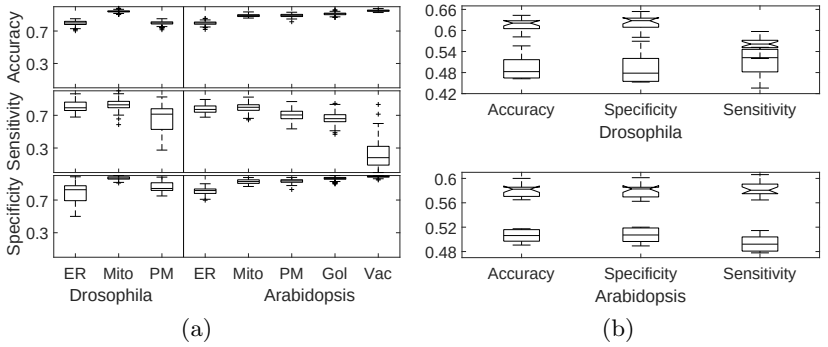


Fig. 4. a) The fraction of extended marker proteins predicted correctly from two hundred runs of an SVM classifier based on the PPI vectors as input and the organelle markers as output classes. The classifier is able to correctly predict the organelle around 70% of the time. b) The spatial proteomics data can make weak predictions on the existence of PPI links. The notched plots show the accuracy, specificity and sensitivity for Drosophila (top) and Arabidopsis (bottom) while the square plots are the results for a randomised PPI network. There is a clear (but faint) signal from the spatial proteomics data.

The inverse problem is more challenging. To attempt to use the spatial proteomics data to estimate whether a PPI exists between two proteins, first each pair of proteins were merged together to create a combined vector by multiplying each component of each vector by all the components of the other. The

six dimensional Drosophila vector, for example, was transformed to a 36 dimensional combined vector. A two-class SVM for interactions and non-interactions was then trained. The new dataset is the combination of all the protein pairs so contains 393,828 protein pairs for Drosophila and 237,016 for Arabidopsis. The data was split into training sets of protein pairs and, as the data is highly skewed towards non-interactions, the sampling was biased to force the training set to contain 50% interactions. The process was repeated twenty times. The accuracy, sensitivity and specificity of the predictions are shown in the notched plots of Figure 4(b). The equivalent SVM was applied to a randomised PPI network and shows what would be expected if there was no signal available (the non-notched plots). While the signal from the real data is small, it is consistently larger than the results from the randomised PPI network and can make some predictions about the PPI network.

4 Discussion

In this paper, we show how sub-cellular proteomics measurements can be combined with information contained in protein-protein interaction networks. Our work shows that there is significant correlation between spatial protein expression in cells and protein interaction information. Using a simple representation of interaction data in a fixed dimensional space, we show that predictions can be made in both directions between spatial proteomics and PPI networks.

There are many potential benefits from using the combined datasets. Differences in classification between the datasets may be of particular interest as it may imply interesting cases such as proteins that exist in multiple compartments or false data that should be re-evaluated. Confidence in conclusions can also be increased if the same conclusion is drawn using two separate datasets. Inspection of the data using some of the visualisation techniques discussed may also be useful for increasing understanding of data quality and building intuition.

References

1. Park, S., Yang, J.S., Shin, Y.E., Park, J., Jang, S.K., Kim, S.: Protein localization as a principal feature of the etiology and comorbidity of genetic diseases. Molecular systems biology 7(1), 494 (2011)
2. Ideker, T., Sharan, R.: Protein networks in disease. Genome research 18(4), 644–652 (2008)
3. Samsonova, A.A., Niranjan, M., Russell, S., Brazma, A.: Prediction of gene expression in embryonic structures of drosophila melanogaster. PLoS computational biology 3(7), e144 (2007)
4. Wieser, D., Niranjan, M.: Remote homology detection using a kernel method that combines sequence and secondary-structure similarity scores. In silico biology 9(3), 89–103 (2009)
5. Gunawardana, Y., Fujiwara, S., Takeda, A., Woo, J., Woelk, C., Niranjan, M.: Outlier detection at the transcriptome-proteome interface. Bioinformatics 31(15), 2530–2536 (2015)

6. Du, P., Wang, L.: Predicting human protein subcellular locations by the ensemble of multiple predictors via protein-protein interaction network with edge clustering coefficients. PloS one 9(1), e86879 (2014)
7. De Duve, C., Beaufay, H.: A short history of tissue fractionation. The Journal of cell biology 91(3), 293 (1981)
8. Gatto, L., Breckels, L.M., Burger, T., Nightingale, D.J., Groen, A.J., Campbell, C., Mulvey, C.M., Christoforou, A., Ferro, M., Lilley, K.S.: A foundation for reliable spatial proteomics data analysis. Molecular & Cellular Proteomics pp. mcp–M113 (2014)
9. Itzhak, D.N., Tyanova, S., Cox, J., Borner, G.H.: Global, quantitative and dynamic mapping of protein subcellular localization. Elife 5, e16950 (2016)
10. Dunkley, T.P., Hester, S., Shadforth, I.P., Runions, J., Weimar, T., Hanton, S.L., Griffin, J.L., Bessant, C., Brandizzi, F., Hawes, C., et al.: Mapping the arabidopsis organelle proteome. Proceedings of the National Academy of Sciences 103(17), 6518–6523 (2006)
11. Tan, D.J., Dvinge, H., Christoforou, A., Bertone, P., Martinez Arias, A., Lilley, K.S.: Mapping organelle proteins and protein complexes in drosophila melanogaster. Journal of proteome research 8(6), 2667–2678 (2009)
12. Jensen, L.J., Kuhn, M., Stark, M., Chaffron, S., Creevey, C., Muller, J., Doerks, T., Julien, P., Roth, A., Simonovic, M., et al.: String 8a global view on proteins and their functional interactions in 630 organisms. Nucleic acids research 37(suppl_1), D412–D416 (2008)
13. Stark, C., Breitkreutz, B.J., Reguly, T., Boucher, L., Breitkreutz, A., Tyers, M.: Biogrid: a general repository for interaction datasets. Nucleic acids research 34(suppl_1), D535–D539 (2006)
14. Kondor, R.I., Lafferty, J.: Diffusion kernels on graphs and other discrete input spaces. In: ICML. vol. 2, pp. 315–322 (2002)
15. Radicchi, F., Castellano, C., Cecconi, F., Loreto, V., Parisi, D.: Defining and identifying communities in networks. Proceedings of the National Academy of Sciences of the United States of America 101(9), 2658–2663 (2004)
16. Vapnik, V.: The nature of statistical learning theory. Springer science & business media (2013)

References

[1] Cbcl face database 1. MIT Center For Biological and Computation Learning, Retrieved Feb 2016. http://www.ai.mit.edu/projects/cbcl.

[2] Cancer program datasets. Broad Institute, Retrieved Feb 2016. http://www.broadinstitute.org/cgi-bin/cancer/datasets.cgi.

[3] Ariana Anderson, Pamela K Douglas, Wesley T Kerr, Virginia S Haynes, Alan L Yuille, Jianwen Xie, Ying Nian Wu, Jesse A Brown, and Mark S Cohen. Non-negative matrix factorization of multimodal mri, fmri and phenotypic data reveals differential changes in default mode subnetworks in adhd. *NeuroImage*, 102:207–219, 2014.

[4] Babajide O Ayinde and Jacek M Zurada. Deep learning of constrained autoencoders for enhanced understanding of data. *IEEE Transactions on Neural Networks and Learning Systems*, 2017.

[5] Babajide O Ayinde and Jacek M Zurada. Nonredundant sparse feature extraction using autoencoders with receptive fields clustering. *Neural Networks*, 93:99–109, 2017.

[6] Babajide O Ayinde, Ehsan Hosseini-Asl, and Jacek M Zurada. Visualizing and understanding nonnegativity constrained sparse autoencoder in deep learning. In *International Conference on Artificial Intelligence and Soft Computing*, pages 3–14. Springer, 2016.

[7] AS Babu and SK Reddy. Exchange rate forecasting using arima, neural network and fuzzy neuron. *Journal of Stock & Forex Trading*, 3(4):1–5, 2015.

[8] Andrew Barron, Jorma Rissanen, and Bin Yu. The minimum description length principle in coding and modeling. *Information Theory, IEEE Transactions on*, 44 (6):2743–2760, 1998.

[9] Christian Bauckhage. Computing the kullback-leibler divergence between two generalized gamma distributions. *arXiv preprint arXiv:1401.6853*, 2014.

[10] David M Blei, Perry R Cook, and Matthew Hoffman. Bayesian nonparametric matrix factorization for recorded music. In *Proceedings of the 27th International Conference on Machine Learning (ICML-10)*, pages 439–446, 2010.

[11] Sanja Brdar, Vladimir Crnojevic, and Blaz Zupan. Integrative clustering by non-negative matrix factorization can reveal coherent functional groups from gene profile data. *Biomedical and Health Informatics, IEEE Journal of*, 19(2):698–708, 2015.

[12] J-P Brunet, P Tamayo, T R Golub, and J P Mesirov. Metagenes and molecular pattern discovery using matrix factorization. *Proceedings of the national academy of sciences*, 101(12):4164–4169, 2004.

[13] Ioan Buciu, Nikos Nikolaidis, and Ioannis Pitas. Nonnegative matrix factorization in polynomial feature space. *Neural Networks, IEEE Transactions on*, 19(6):1090–1100, 2008.

[14] Wray Buntine. Variational extensions to em and multinomial pca. In *European Conference on Machine Learning*, pages 23–34. Springer, 2002.

[15] Deng Cai, Xiaofei He, Xiaoyun Wu, and Jiawei Han. Non-negative matrix factorization on manifold. In *Data Mining, 2008. ICDM'08. Eighth IEEE International Conference on*, pages 63–72. IEEE, 2008.

[16] Deng Cai, Xiaofei He, Jiawei Han, and Thomas S Huang. Graph regularized nonnegative matrix factorization for data representation. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 33(8):1548–1560, 2011.

[17] AT Cegmil. Bayesian inference in non-negative matrix factorization models. *Computational Intelligence and Neuroscience*, 2008.

[18] Girish Chandrashekar and Ferat Sahin. A survey on feature selection methods. *Computers & Electrical Engineering*, 40(1):16–28, 2014.

[19] Jan Chorowski and Jacek M Zurada. Learning understandable neural networks with nonnegative weight constraints. *Neural Networks and Learning Systems, IEEE Transactions on*, 26(1):62–69, 2015.

[20] Pierre Comon. Independent component analysis, a new concept? *Signal processing*, 36(3):287–314, 1994.

[21] Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Machine learning*, 20(3):273–297, 1995.

[22] Giovanni Costantini, Renzo Perfetti, and Massimiliano Todisco. Recurrent neural network for approximate nonnegative matrix factorization. *Neurocomputing*, 138: 238–247, 2014.

[23] John P Cunningham and Zoubin Ghahramani. Linear dimensionality reduction: survey, insights, and generalizations. *Journal of Machine Learning Research*, 16 (1):2859–2900, 2015.

[24] George Cybenko. Approximation by superpositions of a sigmoidal function. *Mathematics of control, signals and systems*, 2(4):303–314, 1989.

[25] Christian De Duve and Henri Beaufay. A short history of tissue fractionation. *The Journal of cell biology*, 91(3):293, 1981.

[26] Ruairí de Fréin, Konstantinos Drakakis, and Scott Rickard. Portfolio diversification using subspace factorizations. In *Information Sciences and Systems, 2008. CISS 2008. 42nd Annual Conference on*, pages 1075–1080. IEEE, 2008.

[27] Ruairí de Fréin, Konstantinos Drakakis, Scott Rickard, and Andrzej Cichocki. Analysis of financial data using non-negative matrix factorization. In *International Mathematical Forum*, volume 3, pages 1853–1870. Journals of Hikari Ltd, 2008.

[28] Arthur P Dempster, Nan M Laird, and Donald B Rubin. Maximum likelihood from incomplete data via the em algorithm. *Journal of the royal statistical society. Series B (methodological)*, pages 1–38, 1977.

[29] K Devarajan. Nonnegative matrix factorization—a new paradigm for large-scale biological data analysis. In *Proceedings of the Joint Statistical Meetings. Joint Statistical Meetings*, pages 6–10, 2006.

[30] Karthik Devarajan. Nonnegative matrix factorization: an analytical and interpretive tool in computational biology. *PLoS Comput Biol*, 4(7):e1000029, 2008.

[31] Chris Ding, Xiaofeng He, and Horst D Simon. On the equivalence of nonnegative matrix factorization and spectral clustering. In *Proceedings of the 2005 SIAM International Conference on Data Mining*, pages 606–610. SIAM, 2005.

[32] Chris Ding, Tao Li, and Wei Peng. On the equivalence between non-negative matrix factorization and probabilistic latent semantic indexing. *Computational Statistics & Data Analysis*, 52(8):3913–3927, 2008.

[33] Pufeng Du and Lusheng Wang. Predicting human protein subcellular locations by the ensemble of multiple predictors via protein-protein interaction network with edge clustering coefficients. *PloS one*, 9(1):e86879, 2014.

[34] Tom PJ Dunkley, Svenja Hester, Ian P Shadforth, John Runions, Thilo Weimar, Sally L Hanton, Julian L Griffin, Conrad Bessant, Federica Brandizzi, Chris Hawes, et al. Mapping the arabidopsis organelle proteome. *Proceedings of the National Academy of Sciences*, 103(17):6518–6523, 2006.

[35] Michael Figurnov, Shakir Mohamed, and Andriy Mnih. Implicit reparameterization gradients. *arXiv preprint arXiv:1805.08498*, 2018.

[36] Ronald A Fisher. The use of multiple measurements in taxonomic problems. *Annals of eugenics*, 7(2):179–188, 1936.

[37] Paul Fogel, Douglas M Hawkins, Chris Beecher, George Luta, and S Stanley Young. A tale of two matrix factorizations. *The American Statistician*, 67(4): 207–218, 2013.

[38] Laurent Gatto, Juan Antonio Vizcaíno, Henning Hermjakob, Wolfgang Huber, and Kathryn S Lilley. Organelle proteomics experimental designs and analysis. *Proteomics*, 10(22):3957–3969, 2010.

[39] Laurent Gatto, Lisa M Breckels, Thomas Burger, Daniel JH Nightingale, Arnoud J Groen, Callum Campbell, Claire M Mulvey, Andy Christoforou, Myriam Ferro, and Kathryn S Lilley. A foundation for reliable spatial proteomics data analysis. *Molecular & Cellular Proteomics*, pages mcp–M113, 2014.

[40] Eric Gaussier and Cyril Goutte. Relation between plsa and nmf and implications. In *Proceedings of the 28th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 601–602. ACM, 2005.

[41] Nicolas Gillis. *Nonnegative Matrix Factorization Complexity, Algorithms and Applications*. PhD thesis, Universite catholique de Louvain, 2011.

[42] Nicolas Gillis. The why and how of nonnegative matrix factorization. *Regularization, Optimization, Kernels, and Support Vector Machines*, 12:257, 2014.

[43] Nicolas Gillis. Introduction to nonnegative matrix factorization. *arXiv preprint arXiv:1703.00663*, 2017.

[44] Nicolas Gillis and François Glineur. Accelerated multiplicative updates and hierarchical als algorithms for nonnegative matrix factorization. *Neural Computation*, 24(4):1085–1105, 2012.

[45] Vladimir Gligorijević and Nataša Pržulj. Methods for biological data integration: perspectives and challenges. *Journal of the Royal Society Interface*, 12(112): 20150571, 2015.

[46] Todd R Golub, Donna K Slonim, Pablo Tamayo, Christine Huard, Michelle Gaasenbeek, Jill P Mesirov, Hilary Coller, Mignon L Loh, James R Downing, Mark A Caligiuri, et al. Molecular classification of cancer: class discovery and class prediction by gene expression monitoring. *science*, 286(5439):531–537, 1999.

[47] Liyun Gong and Asoke K Nandi. An enhanced initialization method for non-negative matrix factorization. In *Machine Learning for Signal Processing (MLSP), 2013 IEEE International Workshop on*, pages 1–6. IEEE, 2013.

[48] Quanquan Gu and Jie Zhou. Neighborhood preserving nonnegative matrix factorization. In *BMVC*, pages 1–10, 2009.

[49] Naiyang Guan, Dacheng Tao, Zhigang Luo, and Bo Yuan. Online nonnegative matrix factorization with robust stochastic approximation. *IEEE Transactions on Neural Networks and Learning Systems*, 23(7):1087–1099, 2012.

[50] Yawwani Gunawardana, Shuhei Fujiwara, Akiko Takeda, Jeongmin Woo, Christopher Woelk, and Mahesan Niranjan. Outlier detection at the transcriptome-proteome interface. *Bioinformatics*, 31(15):2530–2536, 2015.

[51] Mithun Das Gupta and Jing Xiao. Non-negative matrix factorization as a feature selection tool for maximum margin classifiers. In *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*, pages 2841–2848. IEEE, 2011.

[52] Geoffrey E Hinton and Ruslan R Salakhutdinov. Reducing the dimensionality of data with neural networks. *science*, 313(5786):504–507, 2006.

[53] ND Ho, P Van Dooren, and V Blondel. Weighted nonnegative matrix factorization and face feature extraction. *submitted to Image and Vision Computing*, 2007.

[54] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[55] Kurt Hornik. Approximation capabilities of multilayer feedforward networks. *Neural networks*, 4(2):251–257, 1991.

[56] Ehsan Hosseini-Asl, Jacek M Zurada, and Olfa Nasraoui. Deep learning of part-based representation of data using sparse autoencoders with nonnegativity constraints. *IEEE transactions on neural networks and learning systems*, 27(12): 2486–2498, 2016.

[57] Patrik O Hoyer. Non-negative matrix factorization with sparseness constraints. *The Journal of Machine Learning Research*, 5:1457–1469, 2004.

[58] Trey Ideker and Roded Sharan. Protein networks in disease. *Genome research*, 18 (4):644–652, 2008.

[59] Lars J Jensen, Michael Kuhn, Manuel Stark, Samuel Chaffron, Chris Creevey, Jean Muller, Tobias Doerks, Philippe Julien, Alexander Roth, Milan Simonovic, et al. String 8—a global view on proteins and their functional interactions in 630 organisms. *Nucleic acids research*, 37(suppl_1):D412–D416, 2008.

[60] Yuan Wang Yunde Jia and Changbo Hu Matthew Turk. Fisher non-negative matrix factorization for learning local features. In *Proc. Asian Conf. on Comp. Vision*, pages 27–30. Citeseer, 2004.

[61] Zhilong Jia, Xiang Zhang, Naiyang Guan, Xiaochen Bo, Michael R Barnes, and Zhigang Luo. Gene ranking of rna-seq data via discriminant non-negative matrix factorization. *PloS one*, 10(9):e0137782, 2015.

[62] Luis O Jimenez and David A Landgrebe. Supervised classification in high-dimensional space: geometrical, statistical, and asymptotical properties of multivariate data. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, 28(1):39–54, 1998.

[63] Bhargav Kanagal and Vikas Sindhwani. Rank selection in low-rank matrix approximations: A study of cross-validation for nmfs. *reconstruction*, 1:10, 2010.

[64] Kenji Kawaguchi. Deep learning without poor local minima. In *Advances in Neural Information Processing Systems*, pages 586–594, 2016.

[65] Jingu Kim and Haesun Park. Sparse nonnegative matrix factorization for clustering. 2008.

[66] Diederik Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[67] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

[68] Diederik P Kingma, Shakir Mohamed, Danilo Jimenez Rezende, and Max Welling. Semi-supervised learning with deep generative models. In *Advances in Neural Information Processing Systems*, pages 3581–3589, 2014.

[69] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[70] Steve Lawrence, C Lee Giles, Ah Chung Tsoi, and Andrew D Back. Face recognition: A convolutional neural-network approach. *IEEE transactions on neural networks*, 8(1):98–113, 1997.

[71] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *Nature*, 521 (7553):436–444, 2015.

[72] Daniel D Lee and H Sebastian Seung. Learning the parts of objects by non-negative matrix factorization. *Nature*, 401(6755):788–791, 1999.

[73] Daniel D Lee and H Sebastian Seung. Algorithms for non-negative matrix factorization. In *Advances in neural information processing systems*, pages 556–562, 2001.

[74] Te-Won Lee. Independent component analysis. In *Independent component analysis*, pages 27–66. Springer, 1998.

[75] Andre Lemme, René Felix Reinhart, and Jochen Jakob Steil. Efficient online learning of a non-negative sparse autoencoder. In *ESANN*, 2010.

[76] Andre Lemme, René Felix Reinhart, and Jochen Jakob Steil. Online learning and generalization of parts-based image representations by non-negative sparse autoencoders. *Neural Networks*, 33:194–203, 2012.

[77] Yifeng Li and Alioune Ngom. Versatile sparse matrix factorization: Theory and applications. *Neurocomputing*, 145:23–29, 2014.

[78] Chih-Jen Lin. Projected gradient methods for nonnegative matrix factorization. *Neural computation*, 19(10):2756–2779, 2007.

[79] Zachary C Lipton. The mythos of model interpretability. *arXiv preprint arXiv:1606.03490*, 2016.

[80] Haifeng Liu, Zhaohui Wu, Xuelong Li, Deng Cai, and Thomas S Huang. Constrained nonnegative matrix factorization for image representation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 34(7):1299–1311, 2012.

[81] Ji Liu, Jun Liu, Peter Wonka, and Jieping Ye. Sparse non-negative tensor factorization using columnwise coordinate descent. *Pattern Recognition*, 45(1):649–656, 2012.

[82] James T Lo and Devasis Bassu. An adaptive method of training multilayer perceptrons. In *Neural Networks, 2001. Proceedings. IJCNN'01. International Joint Conference on*, volume 3, pages 2013–2018. IEEE, 2001.

[83] James Ting-Ho Lo. Convexification for data fitting. *Journal of global optimization*, 46(2):307–315, 2010.

[84] James Ting-Ho Lo, Yichuan Gui, and Yun Peng. Overcoming the local-minimum problem in training multilayer perceptrons with the nrae training method. In *International Symposium on Neural Networks*, pages 440–447. Springer, 2012.

[85] James Ting-Ho Lo, Yichuan Gui, and Yun Peng. Training deep neural networks with gradual deconvexification. In *Neural Networks (IJCNN), 2016 International Joint Conference on*, pages 1000–1007. IEEE, 2016.

[86] Jiawei Luo, Gen Xiang, and Chu Pan. Discovery of microrna and transcription factors co-regulatory modules by integrating multiple types of genomic data. *IEEE transactions on nanobioscience*, 16(1):51–59, 2017.

[87] Minnan Luo, Feiping Nie, Xiaojun Chang, Yi Yang, Alexander G Hauptmann, and Qinghua Zheng. Probabilistic non-negative matrix factorization and its robust extensions for topic modeling. In *AAAI*, pages 2308–2314, 2017.

[88] David JC MacKay. *Information theory, inference and learning algorithms*. Cambridge university press, 2003.

[89] James MacQueen et al. Some methods for classification and analysis of multivariate observations. In *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability*, volume 1, pages 281–297. Oakland, CA, USA, 1967.

[90] Julien Mairal, Francis Bach, Jean Ponce, and Guillermo Sapiro. Online learning for matrix factorization and sparse coding. *The Journal of Machine Learning Research*, 11:19–60, 2010.

[91] Yun Mao and Lawrence K Saul. Modeling distances in large-scale networks by matrix factorization. In *Proceedings of the 4th ACM SIGCOMM conference on Internet measurement*, pages 278–287. ACM, 2004.

[92] Geoffrey J McLachlan and Kaye E Basford. *Mixture models: Inference and applications to clustering*, volume 84. Marcel Dekker, 1988.

[93] Hrushikesh Mhaskar, Qianli Liao, and Tomaso A Poggio. When and why are deep networks better than shallow ones? In *AAAI*, pages 2343–2349, 2017.

[94] Nasser Mohammadiha, W Bastiaan Kleijn, and Arne Leijon. Gamma hidden markov model as a probabilistic nonnegative matrix factorization. In *Signal Processing Conference (EUSIPCO), 2013 Proceedings of the 21st European*, pages 1–5. IEEE, 2013.

[95] Michael Möller, Ernie Esser, Stanley Osher, Guillermo Sapiro, and Jack Xin. A convex model for matrix factorization and dimensionality reduction on physical space and its application to blind hyperspectral unmixing. Technical report, MINNESOTA UNIV MINNEAPOLIS INST FOR MATHEMATICS AND ITS APPLICATIONS, 2010.

[96] Luis Montesdeoca and Mahesan Niranjan. Extending the feature set of a data-driven artificial neural network model of pricing financial options. In *Computational Intelligence (SSCI), 2016 IEEE Symposium Series on*, pages 1–6. IEEE, 2016.

[97] Lucian Morgos. Non-negative factorization for clustering of microarray data. *International Journal of Computers Communications & Control*, 9(1):16–23, 2014.

[98] Vinod Nair and Geoffrey E Hinton. Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th international conference on machine learning (ICML-10)*, pages 807–814, 2010.

[99] Tu Dinh Nguyen, Truyen Tran, Dinh Phung, and Svetha Venkatesh. Learning parts-based representations with nonnegative restricted boltzmann machine. In *Asian Conference on Machine Learning*, pages 133–148, 2013.

[100] Mahesan Niranjan. Sequential tracking in pricing financial options using model based and neural network approaches. In *Advances in neural information processing systems*, pages 960–966, 1997.

[101] Mohamed F Omran. Nonlinear dependence and conditional heteroscedasticity in stock returns: Uk evidence. *Applied Economics Letters*, 4(10):647–650, 1997.

[102] Le Ou-Yang, Dao-Qing Dai, and Xiao-Fei Zhang. Protein complex detection via weighted ensemble clustering based on bayesian nonnegative matrix factorization. *PloS one*, 8(5):e62158, 2013.

[103] Le Ou-Yang, Dao-Qing Dai, Xiao-Li Li, Min Wu, Xiao-Fei Zhang, and Peng Yang. Detecting temporal protein complexes from dynamic protein-protein interaction networks. *BMC bioinformatics*, 15(1):335, 2014.

[104] Art B Owen and Patrick O Perry. Bi-cross-validation of the svd and the nonnegative matrix factorization. *The annals of applied statistics*, pages 564–594, 2009.

[105] John Paisley, D Blei, and Michael I Jordan. Bayesian nonnegative matrix factorization with stochastic variational inference. *Handbook of Mixed Membership Models and Their Applications. Chapman and Hall/CRC*, 2014.

[106] Stephen E Palmer. Hierarchical structure in perceptual representation. *Cognitive psychology*, 9(4):441–474, 1977.

[107] Sinno Jialin Pan, Qiang Yang, et al. A survey on transfer learning. *IEEE Transactions on knowledge and data engineering*, 22(10):1345–1359, 2010.

[108] Solip Park, Jae-Seong Yang, Young-Eun Shin, Juyong Park, Sung Key Jang, and Sanguk Kim. Protein localization as a principal feature of the etiology and comorbidity of genetic diseases. *Molecular systems biology*, 7(1):494, 2011.

[109] Lorien Y Pratt. Discriminability-based transfer between neural networks. In *Advances in neural information processing systems*, pages 204–211, 1993.

[110] Alec Radford, Luke Metz, and Soumith Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. *arXiv preprint arXiv:1511.06434*, 2015.

[111] William M Rand. Objective criteria for the evaluation of clustering methods. *Journal of the American Statistical association*, 66(336):846–850, 1971.

[112] Masoumeh Rezaei and Reza Boostani. Using the genetic algorithm to enhance nonnegative matrix factorization initialization. *Expert Systems*, 31(3):213–219, 2014.

[113] Danilo Jimenez Rezende, Shakir Mohamed, and Daan Wierstra. Stochastic backpropagation and approximate inference in deep generative models. *arXiv preprint arXiv:1401.4082*, 2014.

[114] Jorma Rissanen. A universal prior for integers and estimation by minimum description length. *The Annals of statistics*, pages 416–431, 1983.

[115] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. Learning internal representations by error propagation. Technical report, California Univ San Diego La Jolla Inst for Cognitive Science, 1985.

[116] Tetsuya Sakurai, Akira Imakura, Yuto Inoue, and Yasunori Futamura. Alternating optimization method based on nonnegative matrix factorizations for deep neural networks. In *International Conference on Neural Information Processing*, pages 354–362. Springer, 2016.

[117] Anastasia A Samsonova, Mahesan Niranjan, Steven Russell, and Alvis Brazma. Prediction of gene expression in embryonic structures of drosophila melanogaster. *PLoS computational biology*, 3(7):e144, 2007.

[118] Andrew M Saxe, James L McClelland, and Surya Ganguli. Exact solutions to the nonlinear dynamics of learning in deep linear neural networks. *arXiv preprint arXiv:1312.6120*, 2013.

[119] Mikkel N Schmidt, Ole Winther, and Lars Kai Hansen. Bayesian non-negative matrix factorization. In *International Conference on Independent Component Analysis and Signal Separation*, pages 540–547. Springer, 2009.

[120] Farial Shahnaz, Michael W Berry, V Paul Pauca, and Robert J Plemmons. Document clustering using nonnegative matrix factorization. *Information Processing & Management*, 42(2):373–386, 2006.

[121] C. E. Shannon. A mathematical theory of communication. *The Bell System Technical Journal*, 27(3):379–423, July 1948. ISSN 0005-8580. doi: 10.1002/j.1538-7305.1948.tb01338.x.

[122] Madhusudana Shashanka, Bhiksha Raj, and Paris Smaragdis. Probabilistic latent variable models as nonnegative factorizations. *Computational intelligence and neuroscience*, 2008, 2008.

[123] Paris Smaragdis and Shrikant Venkataramani. A neural network alternative to non-negative audio models. In *Acoustics, Speech and Signal Processing (ICASSP), 2017 IEEE International Conference on*, pages 86–90. IEEE, 2017.

[124] Steven Squires, Rob Ewing, Adam Prügel-Bennett, and Mahesan Niranjan. A method of integrating spatial proteomics and protein-protein interaction network data. In *International Conference on Neural Information Processing*, pages 782–790. Springer, 2017.

[125] Steven Squires, Luis Montesdeoca, Adam Prügel-Bennett, and Mahesan Niranjan. Non-negative matrix factorization with exogenous inputs for modeling financial data. In *International Conference on Neural Information Processing*, pages 873–881. Springer, 2017.

[126] Steven Squires, Adam Prügel-Bennett, and Mahesan Niranjan. Rank selection in nonnegative matrix factorization using minimum description length. *Neural Computation*, 2017.

[127] Mehrdad Tamiz, R Hasham, DF Jones, B Hesni, and EK Fargher. A two staged goal programming model for portfolio selection. In *Multi-Objective Programming and Goal Programming*, pages 286–299. Springer, 1996.

[128] Denise JL Tan, Heidi Dvinge, Andrew Christoforou, Paul Bertone, Alfonso Martinez Arias, and Kathryn S Lilley. Mapping organelle proteins and protein complexes in drosophila melanogaster. *Journal of proteome research*, 8(6):2667–2678, 2009.

[129] Robert Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society. Series B (Methodological)*, pages 267–288, 1996.

[130] Andreĭ Nikolaevich Tikhonov, Vasiliĭ IAkovlevich Arsenin, and Fritz John. *Solutions of ill-posed problems*, volume 14. Winston Washington, DC, 1977.

[131] Naftali Tishby and Noga Zaslavsky. Deep learning and the information bottleneck principle. In *Information Theory Workshop (ITW), 2015 IEEE*, pages 1–5. IEEE, 2015.

[132] Jochen Triesch. A gradient rule for the plasticity of a neuron's intrinsic excitability. *Artificial Neural Networks: Biological Inspirations–ICANN 2005*, pages 65–70, 2005.

[133] Magnus O Ulfarsson and Victor Solo. Tuning parameter selection for nonnegative matrix factorization. In *ICASSP*, pages 6590–6594, 2013.

[134] Albert Vilamala, Paulo JG Lisboa, Sandra Ortega-Martorell, and Alfredo Vellido. Discriminant convex non-negative matrix factorization for the classification of human brain tumours. *Pattern Recognition Letters*, 34(14):1734–1747, 2013.

[135] E Wachsmuth, MW Oram, and DI Perrett. Recognition of objects and their component parts: responses of single units in the temporal cortex of the macaque. *Cerebral Cortex*, 4(5):509–522, 1994.

[136] Christopher S Wallace and David M Boulton. An information measure for classification. *The Computer Journal*, 11(2):185–194, 1968.

[137] Dong Wang and Huchuan Lu. On-line learning parts-based representation via incremental orthogonal projective non-negative matrix factorization. *Signal Processing*, 93(6):1608–1623, 2013.

[138] Hong-Qiang Wang, Chun-Hou Zheng, and Xing-Ming Zhao. jnmfma: a joint non-negative matrix factorization meta-analysis of transcriptomics data. *Bioinformatics*, page btu679, 2014.

[139] Hua Wang, Heng Huang, Chris Ding, and Feiping Nie. Predicting protein–protein interactions from multimodal biological data sources via nonnegative matrix trifactorization. *Journal of Computational Biology*, 20(4):344–358, 2013.

[140] Jie Wang. Stock trend extraction via matrix factorization. In *International Conference on Advanced Data Mining and Applications*, pages 516–526. Springer, 2012.

[141] Jim Jing-Yan Wang and Xin Gao. Max–min distance nonnegative matrix factorization. *Neural Networks*, 61:75–84, 2015.

[142] Naiyan Wang, Jingdong Wang, and Dit-Yan Yeung. Online robust non-negative dictionary learning for visual tracking. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 657–664, 2013.

[143] Yu-Xiong Wang and Yu-Jin Zhang. Nonnegative matrix factorization: A comprehensive review. *Knowledge and Data Engineering, IEEE Transactions on*, 25(6): 1336–1353, 2013.

[144] Jing Wei and Xin Tian. Network connectivity revealed through non-negative matrix factorization in rat medial prefrontal cortex during working memory task. In *Neural Engineering (NER), 2013 6th International IEEE/EMBS Conference on*, pages 202–205. IEEE, 2013.

[145] Andreas S Weigend, Bernardo A Huberman, and David E Rumelhart. Predicting the future: A connectionist approach. *International journal of neural systems*, 1 (03):193–209, 1990.

[146] Daniela Wieser and Mahesan Niranjan. Remote homology detection using a kernel method that combines sequence and secondary-structure similarity scores. *In silico biology*, 9(3):89–103, 2009.

[147] Svante Wold, Kim Esbensen, and Paul Geladi. Principal component analysis. *Chemometrics and intelligent laboratory systems*, 2(1-3):37–52, 1987.

[148] Qingyao Wu, Zhenyu Wang, Chunshan Li, Yunming Ye, Yueping Li, and Ning Sun. Protein functional properties prediction in sparsely-label ppi networks through regularized non-negative matrix factorization. *BMC systems biology*, 9(Suppl 1): S9, 2015.

[149] Siqi Wu, Antony Joseph, Ann S Hammonds, Susan E Celniker, Bin Yu, and Erwin Frise. Stability-driven nonnegative matrix factorization to interpret spatial gene expression and build local gene networks. *Proceedings of the National Academy of Sciences*, 113(16):4290–4295, 2016.

[150] Eric P Xing, Michael I Jordan, Stuart J Russell, and Andrew Y Ng. Distance metric learning with application to clustering with side-information. In *Advances in neural information processing systems*, pages 521–528, 2003.

[151] Jason Yosinski, Jeff Clune, Yoshua Bengio, and Hod Lipson. How transferable are features in deep neural networks? In *Advances in neural information processing systems*, pages 3320–3328, 2014.

[152] Zhijian Yuan and Erkki Oja. Projective nonnegative matrix factorization for image compression and feature extraction. *Image analysis*, pages 333–342, 2005.

[153] N Yuvaraj and P Vivekanandan. An efficient svm based tumor classification with symmetry non-negative matrix factorization using gene expression data. In *Information Communication and Embedded Systems (ICICES), 2013 International Conference on*, pages 761–768. IEEE, 2013.

[154] Rafal Zdunek. Initialization of nonnegative matrix factorization with vertices of convex polytope. In *Artificial Intelligence and Soft Computing*, pages 448–455. Springer, 2012.

[155] Matthew D Zeiler and Rob Fergus. Visualizing and understanding convolutional networks. In *European conference on computer vision*, pages 818–833. Springer, 2014.

[156] Daoqiang Zhang, Zhi-Hua Zhou, and Songcan Chen. Non-negative matrix factorization on kernels. In *PRICAI 2006: Trends in Artificial Intelligence*, pages 404–412. Springer, 2006.

[157] Shihua Zhang, Qingjiao Li, Juan Liu, and Xianghong Jasmine Zhou. A novel computational framework for simultaneous integration of multiple types of genomic data to identify microrna-gene regulatory modules. *Bioinformatics*, 27(13):i401–i409, 2011.

[158] Xiang Zhang, Naiyang Guan, Zhilong Jia, Xiaogang Qiu, and Zhigang Luo. Semi-supervised projective non-negative matrix factorization for cancer classification. *PloS one*, 10(9):e0138814, 2015.

[159] Yuan Zhang, Nan Du, Liang Ge, Kebin Jia, and Aidong Zhang. A collective nmf method for detecting protein functional module from multiple data sources. In *Proceedings of the ACM Conference on Bioinformatics, Computational Biology and Biomedicine*, pages 655–660. ACM, 2012.

[160] Hui Zou, Trevor Hastie, and Robert Tibshirani. Sparse principal component analysis. *Journal of computational and graphical statistics*, 15(2):265–286, 2006.