Seeding Non-Negative Matrix Factorizations with the Spherical K-Means Clustering

by

Stefan Wild

B.S., University of Colorado, 2002

A thesis submitted to the

Faculty of the Graduate School of the

University of Colorado in partial fulfillment

of the requirements for the degree of

Master of Science

Department of Applied Mathematics

2003

This thesis entitled:
Seeding Non-Negative Matrix Factorizations with the Spherical K-Means Clustering
written by Stefan Wild
has been approved for the Department of Applied Mathematics

James Curry

Anne Dougherty

Meredith Betterton

Date ______________

Wild, Stefan (M.S., Applied Mathematics)

Seeding Non-Negative Matrix Factorizations with the Spherical K-Means Clustering

Thesis directed by Professor James Curry

The size of data sets found in practice often prevents standard information retrieval and machine learning techniques, such as classification, feature extraction, and querying, from being easily applied on very large, and often times, uncompressed data sets. In general, because of our need to process and extract information, we are naturally led to some form of "data reduction" or compression. Such a reduction may be desirable to reduce our cost of storage or as a more efficient use of memory.

Unfortunately, the usual compaction or rank reduction techniques often fail to preserve the given array properties, such as non-negativity. For example, failure to preserve column additivity or basis and coefficient weights in the initial data set could either hinder correct visualization completely or lead to undesired artifacts.

In this thesis we explore a recent iterative technique called Non-negative Matrix Factorization (NMF). This technique preserves much of the structure of the original data and guarantees that both basis and weights are non-negative. Furthermore, several special properties are obtained as a result of the constrained optimization problem of NMF. Specifically, in the case of facial image data, the additive nature of NMF has been shown to result in a basis of facial features, such eyes, noses, and lips. We explore various methods for efficiently computing NMF, placing particular emphasis on the initialization of current NMF algorithms. We propose using the Spherical K-Means clustering method to produce a structured initialization for NMF. We demonstrate some of the properties and initial speed-up that result from this structured initialization. In proposing this initialization strategy, we also arrive at a new efficient way of choosing the rank of the low dimensional NMF representation. We apply NMF to a variety of different types of data. We analyze data sets comprised of: facial images, text documents, and remote sensing data. The application of NMF to remote sensing data is new. Several interesting results, analogous to those found for facial images, are obtained using the Spherical K-Means initialization and NMF and its variants.

Acknowledgements

First and foremost, I extend my thanks to my two advisors, James Curry and Anne Dougherty, with whom I have had the pleasure of working as both an undergraduate and a Master's candidate. We have had countless invaluable and inspiring discussions along the way, and the dozens of paths we have pursued have all contributed to this document. I'd also like to thank Meredith Betterton, the third member of my Master's committee.

I further wish to acknowledge my office mates, Dave Beltran-del-Rio, Barbara Jennings, Deborah Batista and Uli Schneider, who (unbeknownst to them) served as my graduate mentors. Additionally, there have been many others in the Department of Applied Math who have been a continual source of inspiration, and whose efforts are whole-heartedly appreciated.

I would also like to thank Dr. Curry's undergraduate research group Moorea Brega, Alejandro Cantarero, Corry Lee and their mentor Kristian Sandberg for their compilation of the Faces data set used here and for igniting the spark behind my initial exploration of Non-negative Matrix Factorization. Alexander Goetz from the Center for the Study of Earth from Space (CSES) gave the colloquium talk that inspired my work with remote sensing data. His post-doctoral fellows Bruce Kindel and Eric Johnson provided the AVIRIS data and gave me the necessary crash course in AVIRIS imaging and are due many thanks.

Computing has been an integral part of this work and I wish to thank Bruce Fast, for his IT wizardry, and the Manteuffel/McCormick research group, especially Luke Olson, for helping provide additional computing resources.

Lastly, I would like to express my deepest thanks to my family, Martin, Regula, Andy and Heidi Wild, and Hilary Petrock, for their everlasting support.

This work was initiated under NSF VIGRE Grant number DMS-9810751.

Contents

Tables

Figures

Chapter 1

Introduction

This thesis will address some of the aims of information retrieval and machine learning techniques, including classification, feature extraction, and querying. In particular, we focus on two specific aspects of these problems: speed and structure.

Even with the continuing decrease in cost of disk space and memory, it seems that there will always exist data sets that are just beyond the reach of modern computational techniques. In general, for these techniques to be applied to a large data set, we often desire some form of compression (compaction) of the original data. This compression reduces both the amount of physical drive space required to store data and the memory required in performing the computations that allow for exploration of the data. In performing this compression, we wish to maximize the speed of the actual compression as well as the speed of subsequent computations. Current techniques follow the strategies of rank reduction used in linear algebra and are founded on widespread research ([1], [10], [18], [22], and [26]).

However, in doing compression, these rank reduction techniques usually destroy any special structure of the original data array. In particular, current techniques fail to preserve the non-negativity of the data found in practice. Both the resulting basis and its accompanying coefficients (weights) lie outside of the original non-negative space of the data. This often prevents us from viewing the basis and/or sustaining the additivity of the basis components.

To address the issue of structure, we explore a recent iterative technique called Non-negative Matrix Factorization (NMF) ([33], [34], [35], and [36]) . This technique preserves much of the structure of the original data and guarantees that both basis and weights are non-negative. Further, several special properties are obtained as a result of the constrained optimization problem of NMF. Specifically, in the case of facial image data, the additive nature of the NMF has been shown to result in a basis of facial features: eyes, noses, and lips.

In this thesis we explore various methods for efficiently computing NMF. We place particular emphasis on the initialization of current NMF algorithms. We suggest using a clustering technique to obtain a structured initialization for NMF and illustrate some of the properties that result. We also further motivate the application of NMF to a variety of different types of data. We specifically include the results of NMF on a remote sensing data set and show how results analogous to those of the traditional facial image collections are obtained.

1.1 Outline

The layout of the thesis is as follows. In Chapter 2 we provide the necessary background and theory. We begin by introducing the general vector space model which allows us to represent a collection of data in a convenient numerical matrix form. In more specific terms, the vector space models for text and image collections are presented. We then lay the foundation for Non-negative Matrix Factorization (NMF) that will be relied on for the remainder of this document. A simple motivating example is introduced for illustration purposes. We review the three NMF algorithms used in practice and present their individual update strategies as iterative quality calculations. We also provide a sufficient background of a recent form of the K-Means Method called Spherical K-Means, whose efficient clustering approach will be used in subsequent chapters.

In Chapter 3 we revisit the three fundamental ideas (the vector space model, NMF and Spherical K-Means) of Chapter 2 from a practical implementation standpoint. We commence by introducing two data sets that are used to illustrate the techniques used throughout this document. We begin examining the numerical implementation of NMF by detailing the computational requirements of each of the three traditional NMF algorithms. We explicitly provide a numerical correction to the costly divide by zero errors historically encountered by NMF algorithms. A theorem is provided to allow for the normalization of the NMF basis vectors for any desired norm. We also briefly introduce the previously unexplored problem of initializing NMF algorithms.

We conclude Chapter 3 by considering the implementation of the Spherical K-Means clustering method. We examine the initialization of the algorithm as well as its empirical convergence. In doing so, we obtain some counterintuitive results on the relationship between the number of clusters and number of iterations required. We also make some remarks on a recent refinement that significantly reduces the potential number of dot product computations required at each iteration. With this refinement in place, we provide the upper bound on the computational complexity of Spherical K-Means iterations. We briefly mention how Spherical K-Means may be applied to any (unnormalized) non-negative data set and provide references to other recent refinements. We conclude Chapter 3 by relating the output of Spherical K-Means to our matrix factorization problem through an introduction of Concept Decompositions. Concept Decompositions provide a simple starting point for viewing NMF in a new light.

In Chapter 4 we implement NMF and Spherical K-Means on a collection of facial images. In doing so, we first verify the special structure of the NMF factors which we sought to obtain. Repeated implementation of each of the three traditional NMF algorithms allows us to illustrate the differences between the three outputs obtained. For purposes of comparison, we also provide the Spherical K-Means centroids for this data set and comment on the differences between the outputs of NMF and Spherical K-Means.

In Chapter 5 we turn to the problem of initializing (or, "seeding") NMF. We motivate structured initialization and make the case for specifically using the Spherical K-Means output. We then prove a theorem relating the objective functions of Spherical K-Means and the Euclidean Distance NMF algorithm. We also use the monotonicity guaranteed by NMF to prove a loose NMF error bound and we prove that under certain conditions the Spherical K-Means output results in a fixed point in the NMF updates. Next we show that improving the quality of the Spherical K-Means clusters improves the NMF initialization. We provide graphical results that illustrate the progression of the NMF basis for both random (historical) and Spherical K-Means (proposed) initializations. We conclude the chapter by suggesting an application of our proposed technique. We show how one could efficiently determine the number of basis vectors (rank) of our NMF approximation according to demands on the resulting error.

In Chapter 6, we introduce a third data set in an effort to further motivate the application of NMF algorithms to a variety of data sets. We first provide a thorough introduction of the type of spectral data used and the interpretation of NMF output we desire. We then show the results of NMF for both the historical and proposed initializations and comment on the error and structure of the two results. We further use the NMF output to solve an interesting problem from the remote sensing literature and compare our strategy with the technique currently used.

We conclude by listing a variety of future problems and work for this promising area in Chapter 7. Further concluding remarks may also be found in Chapter 7. Specific details on the machines used to obtain all of the results in this thesis may be found in Appendix A.

Chapter 2

Background and Theory

In this chapter, we will introduce the theory behind three fundamental topics. We begin by describing the Vector Space Model, whereby different types of data may be encoded and given a mathematical representation. Next, we introduce Non-negative Matrix Factorization and the three different objective functions used here. We will motivate Non-negative Matrix Factorization by a simple illustrative example before providing the traditional Non-negative Matrix Factorization update strategies. Finally, we use the theory of the traditional K-Means clustering method to motivate the Spherical K-Means clustering method used throughout the remainder of the paper.

2.1 The Vector Space Model

A vector space model can be used to encode a set of objects and features. In this model, each object is represented by a column vector of length m, where m is the total number of distinct object features in the entire set of n objects. We then build an $m \times n$ matrix, $\mathbf{A}$, where each element a_{ij} is the weight of feature i in object j as shown in Figure 2.1. Because of the numerous ways in which this general model may be applied, we now detail the vector space model for two specific types of data: text and images.

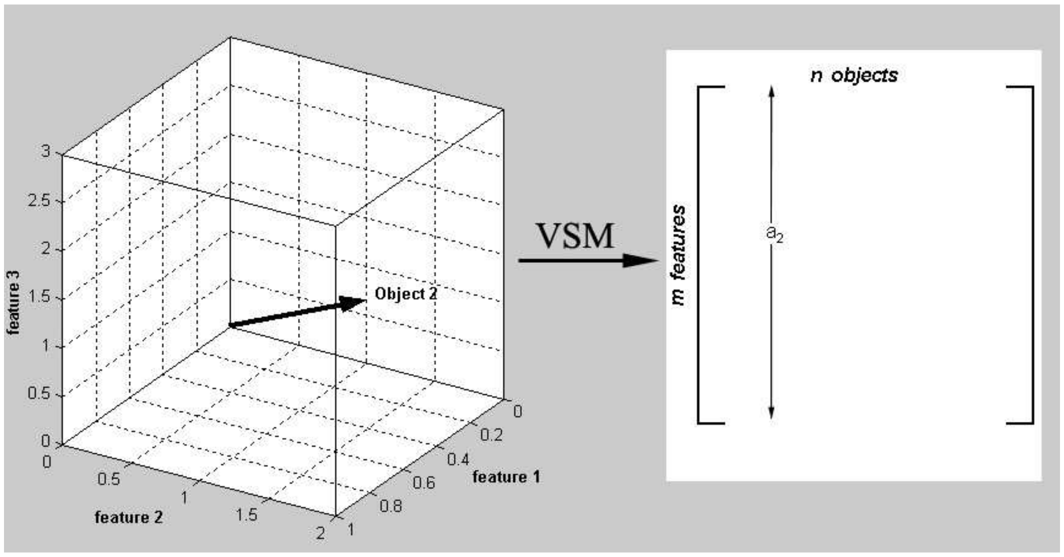


Figure 2.1: The Vector Space Model.

2.1.1 Vector Space Model for Text

The vector space model has been widely used in a variety of text applications, most notably in Latent Semantic Indexing (LSI) [10] and related search engine methods [1]. It is the goal of information retrieval techniques such as LSI to extract the hidden (or "latent") content from text documents and then represent the documents using only the distinguishing latent features. By latent features, we mean the features that primarily describe the conceptual content of the documents while never actually appearing as a specific document in the collection.

In general, we seek to represent a set of n documents as a linear combination of m terms by creating n document vectors $x_1, x_2, \dots, x_n$ in $\Re^m$. Indexed terms (terms used as features in the model) are determined through a series of preprocessing steps where all unique words are extracted from the document collection. Various information retrieval techniques (see [18] for example) such as removing stop words (frequently occurring words such as "the") and stemming (removing plurals and suffixes such as "-es" and "-ing") may also be implemented to reduce the number of unique terms and provide a more conceptual, lower dimensional model.

After preprocessing, the frequency f_{ij} for term i in document j can be used to represent the i,j-th element of the $m \times n$ term-by-document matrix $\mathbf{X}$. Additional local and global weighting schemes could be applied to take into account the effect of a particular term or document on the individual frequencies. A study of several suggested weighting schemes may be found in [29]. Many of these schemes have been implemented in Michael Berry's General Text Parser (GTP) [2] software which systematically creates term-by-document matrices given a collection of documents.

Because elements in the term-by-document matrix correspond to frequencies, the matrix $\mathbf{X}$ is non-negative for most weighting schemes. For non-negative matrices, we also choose to normalize each column, x_i, so that it is of unit length in the L^2 norm. Normalization has the effect of placing equal weight on each document in the collection and restricts the Frobenius norm of the matrix $\mathbf{X}$ so that $\|\mathbf{X}\|_F = \sum_{i,j} x_{i,j}^2 = m$. Further, because each document usually contains only a very small subset of terms, the vast majority of the entries in $\mathbf{X}$ are zero. In practice, term-by-document matrices are more than 95% sparse, and, in many cases, more than 99.5% sparse. For example, the Classic3 data set's [9] (introduced in Chapter 3) term-by-document matrix is shown in Figure 2.2. In this image, the white dots represent non-zero elements in the data matrix, while the black areas represent zeroes.

2.1.2 Vector Space Model for Image Collections

A vector space model may also be used to represent image collections. Traditionally, rectangular images are stored in matrices where each entry location corresponds to a pixel's location, and each entry weight represents the pixel's color. When merging several images together to form a collection, each image is represented as a column vector, $x_i \in \Re^m$, where m is the total number of pixels in the image. For a collection of n images, each row of the $m \times n$ pixel-by-picture matrix $\mathbf{X}$ now corresponds to the

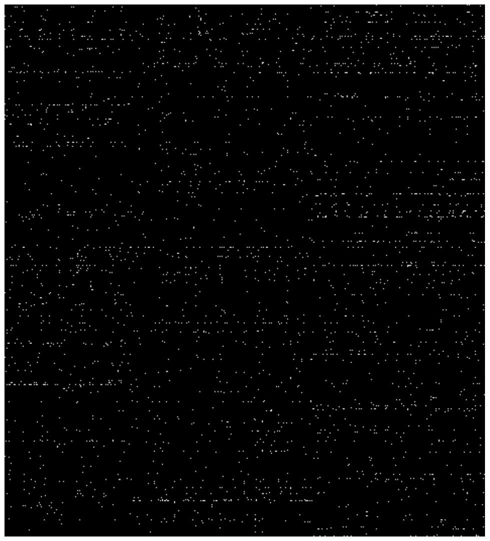


Figure 2.2: The Classic3 data set (4099 documents by 3893 terms): Black pixels correspond to zeroes and white pixels correspond to non-zero elements.

location of a particular pixel across all images in the collection. This restricts the images in the collection to be of the same width, height and resolution.

In this paper, we consider only black and white (grayscale) images, noting that color image collections could only be analyzed given an appropriate color weighting scheme (see color image experiments in [23]). For our images, term weights vary on a grey scale from 0 to 255, 0 corresponding to “solid” black and 255 corresponding to “solid” white. This weighting scheme is necessary for notation only in that it restricts our palette to consist of 256 distinct colors. Given this restriction, there is still a huge number (256^{pq}) of ways in which an image with $p \times q$ pixels can be represented (where $pq = m$ is the total number of rows in $\mathbf{X}$). However, for a specific pattern, such as a human face or a handwritten character, the number of appropriate configurations is a small subset of these possibilities [36]. It is this property that the subsequent analysis of the image vector space model wishes to exploit. Once again, to equally weight the contributions of each image in the collection, the column (image) vectors are normalized to have unit length in the (Euclidean) two-norm.

2.1.3 Vector Space Models for Other Data Types

So far we have introduced vector space models for the two types of data traditionally encountered in standard information retrieval situations. Extension to other types of data loosely involving n objects characterized by m features is fairly straightforward. As an example, in Chapter 6 we introduce a remote sensing data set and detail how this data set is encoded in a vector space model so that the suggested techniques may be applied.

2.2 Non-Negative Matrix Factorization

For the class of large problems where the benefits/efficiency of a vector space model becomes critical, we often desire some form of compression (compaction) of the original data. Compression is necessary for two reasons. First, we wish to reduce the amount of physical disk space required to store data. Second, and often both more important and more difficult to implement, the memory required in performing computations on the original (uncompressed) data set prevent us from being able to explore the data. Whether the intent is to extract features, classify occurrences of specific words or compare a query point with those in the collection, standard computing techniques remain largely unscalable. Further, even with the continuing decrease in cost and increase in availability of disk space and memory, it is likely that there will always exist data sets that are just beyond the reach of modern computational techniques.

A standard starting point for compression is rank reduction – the process of using a low dimensional subspace to approximate a much larger one. While traditional rank reducing techniques are often very good at addressing the need for a compressed data set, for many applications they often fall short in producing subspaces that lend themselves well to subsequent computations. Also, these techniques do not necessarily preserve a desired structure, resulting in lower dimensional bases outside of the non-negative space of the original data matrix. The QR, Singular Value (SVD), and Semi-Discrete (SDD, [30]) decompositions, and the Principal Component Analysis (PCA) classification technique that are traditionally used in information retrieval applications [1] where rank reduction is desired cannot enforce non-negativity of their bases. Non-negative Matrix Factorization (NMF) was proposed in 1997 by Lee and Seung (see [33], [34], and [35]). As we will see, NMF preserves much of the structure of the original data and guarantees that both basis and weights are non-negative. The drawback of NMF is relatively slow convergence. Recently, there has been considerable activity in this area as the suggested techniques are implemented and refined. For example, several recent contributions (see [24], [36], or [43]) compare the effectiveness of NMF to PCA for image classification purposes. We now introduce the general framework for NMF.

2.2.1 Non-Negative Matrix Factorization Theory

Given a non-negative matrix $\mathbf{X}$ of size $m \times n$, NMF algorithms seek to find non-negative factors $\mathbf{W}$ and $\mathbf{H}$ such that

$$\mathbf{X} \approx \tilde{\mathbf{X}} \equiv \mathbf{WH}, \quad \text{where} \quad \mathbf{W} \in \Re^{m\times r} \quad \text{and} \quad \mathbf{H} \in \Re^{r\times n}. \tag{2.1}$$

Intuitively, we think of $\mathbf{W}$ as the matrix containing the NMF basis and $\mathbf{H}$ as the matrix containing the accompanying coefficients (weights).

We also desire that the resulting factorization require less storage than the original data set. If an $m \times n$ matrix can be thought of as requiring mn units of storage, NMF produces factors $\mathbf{W}$ and $\mathbf{H}$ requiring $r(m+n)$ units of storage. The reasonable assumption of keeping the precision of NMF factors equal to the precision of the original matrix ensures that storage will be reduced whenever the number of basis vectors, r, is chosen such that:

$$r < \frac{nm}{m+n}. \tag{2.2}$$

In practice, r is usually chosen such that $r \ll \min(m,n)$ and its selection is analogous to choosing the desired rank in rank reduction problems.[1] To see this we write the factorizations in terms of the columns of $\mathbf{X}$ and $\mathbf{H}$:

$$x_j \approx \tilde{x_j} = \mathbf{W}h_j, \qquad \text{where} \quad x_j \in \Re^m \quad \text{and} \quad h_j \in \Re^r \qquad \text{for } j = 1, \ldots, n. \tag{2.3}$$

Using this representation, we see that the left factor $\mathbf{W}$ contains a basis used for the linear approximation of $\mathbf{X}$. The right factor $\mathbf{H}$ is a coefficient matrix used to add up combinations of the basis vectors in $\mathbf{W}$. The non-negative constraint on $\mathbf{W}$ allows us to visualize the basis columns in the same manner as the columns in the original data matrix. This is the first benefit of NMF versus alternative factorizations like the SVD (whose low rank approximation results in the smallest error) where the basis vectors contain negative components that prevent similar visualization. For text applications, this basis will be r conceptual (or representative) documents stored in the columns of $\mathbf{W}$ that can sum up to (approximately) reconstruct the original document collection. Similarly, for image collections, this basis consists of r representative images stored in the columns of $\mathbf{W}$.

The elements in $\mathbf{H}$ may be thought of as (non-negative) coefficients used to weight the linear combination of basis vectors used to approximate each column in $\mathbf{X}$. The non-negative restriction on these coefficients result in the additive nature of NMF. For many types of data (see for example: house and facial images in [35], handwriting samples in [33] and [43], and music tones in [27]), the additive property of NMF has been shown to result in bases that represent components of the original data (i.e.- doors and eyes, curves of letters and notes in a chord).

The non-negativity constraints on both $\mathbf{W}$ and $\mathbf{H}$ do not come without a cost. From a rank reduction standpoint, NMF is more computationally demanding (this will

[1] Actually, the problem of choosing r for NMF is considerably more cloudy than choosing a desired rank in traditional rank reduction techniques. Factorizations without constraints on the resulting factors (like the SVD) benefit from the geometry of the fundamental subspaces and accordingly, a rank is usually chosen based on the decay of the magnitudes of the eigenvalues of the data. The non-negativity constraints prevent a similar argument here. This topic will be explored further in Chapter 5.

be expanded upon later) and produces lower quality approximations in the sense of Frobenius error than traditional alternatives. Constrained basis vectors prevent NMF from matching the optimal lower rank approximation obtained by the SVD. We now illustrate this behavior by using three different techniques to approximate a rank 3889 matrix (the Classic3 matrix $\mathbf{T}$) for a number of small ranks (note that $r \ll \min(m, n) = 3893$).[2] Here, the quality of the approximations is measured by the Frobenius norm:

$$\text{Quality} \equiv \sqrt{\sum_{i=1}^{m}\sum_{j=1}^{n} e_{ij}^2} = \|\mathbf{E}\|_F = \|\mathbf{T} - \tilde{\mathbf{T}}_r\|_F \tag{2.4}$$

where $\tilde{\mathbf{T}}_r$ represents the rank r approximation of $\mathbf{T}$.

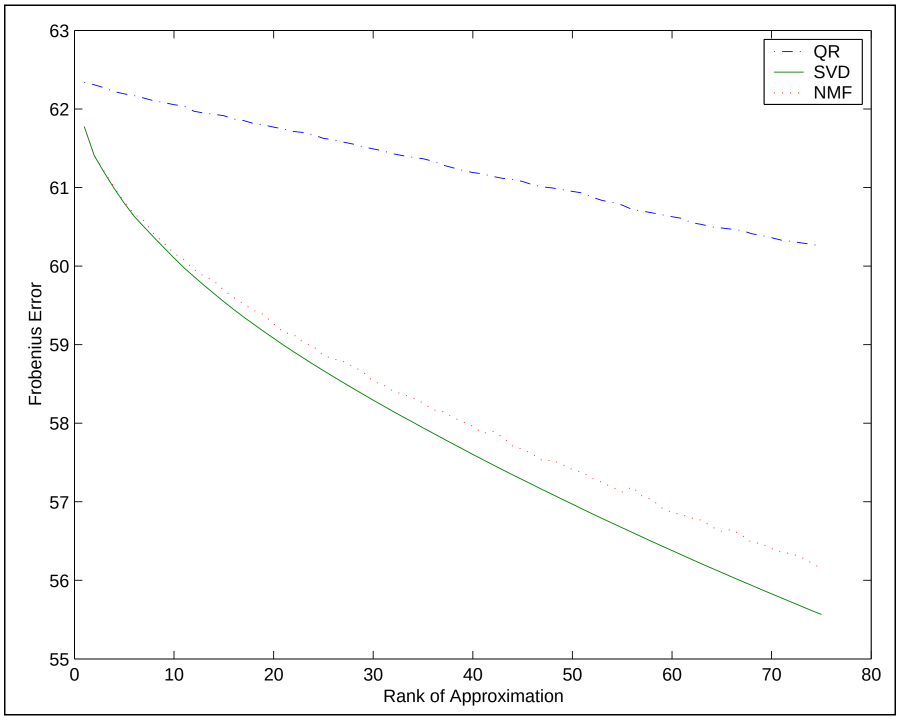


Figure 2.3: Quality of low rank approximations to Classic3 matrix $\mathbf{T}$ (as measured by the Frobenius norm).

Figure 2.3 shows the quality of the approximations obtained by the QR, SVD and NMF rank reducing techniques. First, the ordered QR approximation is shown to offer an example of a factorization of $\mathbf{T}$ into a product of two matrices which are

[2] We remark here that the Classic3 matrix $\mathbf{T}$ is not full rank.

not necessarily non-negative. Further, in this case the a priori orthogonality restriction placed on the factor $\mathbf{Q}$ yielded a relatively poor approximation. The SVD generates the lower rank approximation whose error is smallest in the Frobenius norm [22]. Lastly, some results[3] generated by one of the NMF algorithms detailed later are provided.

From this figure one should understand the relative effectiveness (when compared to the truncated QR) of NMF in producing lower rank approximations of a specific form. This form is achieved by a relatively small gain in error but does not require any more storage than the optimal SVD approximation. In the next section we further motivate the form of this factorization with a small example.

2.2.2 A Simple Motivating Example

To illustrate the additive property of the factors obtained by an NMF, consider the set of 5×5 tri-tone puzzles in Figure 2.4. Suppose we wish to reconstruct all six puzzles by adding together different combinations of three basic puzzles. The non-negativity of these three basic puzzles, which will make up the left factor $\mathbf{W}$, must be enforced to preserve the appearance of these puzzles. The non-negativity of the cofficients in the right factor $\mathbf{H}$ will allow only non-negative combinations of the basic puzzles. These two non-negativity constraints will prevent the six original puzzles in Figure 2.4 from being identically constructed.

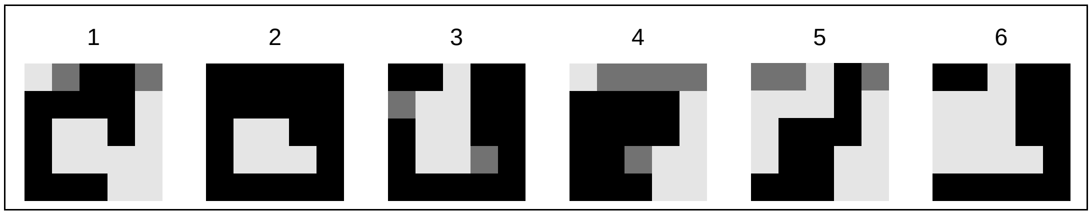


Figure 2.4: Motivating example: Six 25-pixel puzzles.

[3] Because of the iterative nature of NMF, these results represent one sample (non-unique) factorization. In practice, the error shown for each rank may be slightly reduced (but will remain above that of the SVD) by allowing more iterations and varied initializations.

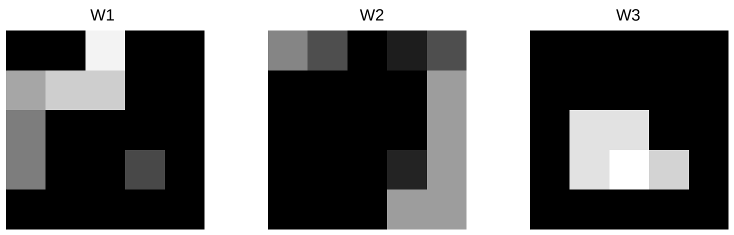


Figure 2.5: Motivating example: Final basic puzzles ($r = 3$).

In this example, the data matrix would be of size $m = 25 \times n = 6$ and the number of basic puzzles ($r = 3$) was chosen accordingly to reduce the storage requirements of the factors $\mathbf{W}$ and $\mathbf{H}$. As noted earlier, storage could also have been lowered by choosing $r = 4$ or $r = 2$ since $\frac{nm}{n+m} \approx 4.84$.

The final three basic puzzles, extracted from the puzzle basis $\mathbf{W}$, as produced by the LNMF algorithm[4] , introduced in the following section, are shown in Figure 2.5. These three puzzles would be difficult to visually construct from the original puzzle set. However, inspection confirms that all six original puzzles can be constructed (less some small error) by adding (strictly non-negtaive) combinations of the basis in Figure 2.5. One can think of these three basic puzzles as being filters applied at varying weights to reconstruct each original puzzle.[5]

Lastly, we again acknowledge that, in addition to the lowered rank, the constraints placed on our approximation usually prevent the original data from being identically reconstructed. We shall define $\mathbf{E}$ to be the error matrix associated with the original matrix $\mathbf{X}$ such that $\mathbf{E} \equiv |\mathbf{X} - \tilde{\mathbf{X}}|$. This error matrix is illustrated in Figure 2.6 which shows the error (where a black pixel corresponds to an exact replication of that pixel,

[4] LNMF algorithm with $r = 3$ and random initial seeding of $\mathbf{W}$ and $\mathbf{H}$. We do note that the three basic puzzles shown here are no longer tri-tone.

[5] For example, consider a stage manager who wants her lighting technician to produce a variety of surfaces using a very limited number of expensive gels (partially transparent colored light filters). NMF determines the two necessary parameters to solve this problem. First NMF produces a gel basis (in $\mathbf{W}$) that instructs the lighting technician what gels are needed to best recreate the desired textures. Next, NMF provides the weight (in $\mathbf{H}$) at which each gel must be applied to produce each individual texture. Then it is only a matter of adding the correct basic gels (at the correct weight) to produce any of the desired textures.

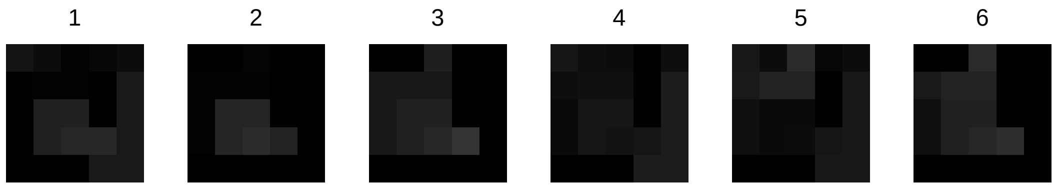


Figure 2.6: Motivating example: Error of approximating puzzles.

and the lighter the grey, the greater the approximation error for that pixel) of the rank three approximation for each pixel of the six original puzzles.

2.2.3 NMF Algorithms

As illustrated in the motivating example above, the benefits (parts-based representations of the original data) that the non-negative constraints on **W** and **H** result in do not come without a cost. Non-negative matrix factorizations can be very difficult to compute. Lee and Seung have suggested an approach similar to that used in Expectation-Maximization (EM) algorithms (see [11]). This approach seeks to iteratively update the factorization based on a given objective function. We now introduce the two traditional objective functions that produce the NMF algorithms used in the literature. We also introduce a recently proposed refinement called Local Non-Negative Matrix Factorization (LMNF) that leads to a third NMF algorithm.

The three algorithms introduced below each seek to minimize a different objective function (distance measure). Each of these objective functions could be minimized with several different iterative procedures. The particular update strategies given here are shown because of their implementation ease and because they have been proven to monotonically decrease their respective objective function ([34] and [36]). We acknowledge that other update strategies that monotonically decrease one of the objective functions here are conceivable.

2.2.3.1 Euclidean Distance Algorithm

First, we consider the Euclidean distance between each column of $\mathbf{X}$ and its approximation $\tilde{\mathbf{X}} = \mathbf{WH}$. For computational purposes, we will use the sum of the squared distances between each column vector x_j in the original data matrix and its approximation $\tilde{v}_j \approx \mathbf{W}h_j$. Using this distance measure, we arrive at the following objective function, which uses the Frobenius norm for matrices introduced earlier:

$$\Theta_{NMF_E}(\mathbf{W},\mathbf{H}) \equiv \sum_{j=1}^{n} \|x_j - \mathbf{W}h_j\|_2^2 = \|\mathbf{X} - \mathbf{WH}\|_F^2 \equiv \sum_{i=1}^{m}\sum_{j=1}^{n}\left(X_{ij} - \sum_{l=1}^{r} W_{il}H_{lj}\right)^2 \tag{2.5}$$

In doing NMF using the Euclidean Distance Algorithm, we wish to find the factors $\mathbf{W}$ and $\mathbf{H}$ that minimize the objective function $\Theta_{NMF_E}(\mathbf{W},\mathbf{H})$. The lower bound of this objective function is zero and will only be attained when a strict equality $\mathbf{X} = \mathbf{WH}$ is obtained. There are many ways to minimize this objective function, however, because of the lack of convexity in both variables $\mathbf{W}$ and $\mathbf{H}$, we can, at best, expect to achieve only local minima [34].

Thus far, researchers in this area have chosen to balance algorithm complexity and convergence speed by using the following update procedure:

$$H_{aj} \leftarrow H_{aj}\frac{[\mathbf{W}^T\mathbf{X}]_{aj}}{[\mathbf{W}^T\mathbf{WH}]_{aj}} \tag{2.6}$$

$$W_{ia} \leftarrow W_{ia}\frac{[\mathbf{XH}^T]_{ia}}{[\mathbf{WHH}^T]_{ia}} \tag{2.7}$$

In (2.6) and (2.7), $[\cdot]_{ij}$ indicates that the noted divisions and multiplications are computed element by element. When written this way, it is evident that the update consists of multiplying the current factors by a measure of the quality of the current approximation. Figure 2.7 illustrates the progression of the three basic puzzles of our motivating example. Each horizontal row represents what the basis collection in $\mathbf{W}$ looks like every fifth iteration. The last row closely resembles the final basis (obtained after 50 iterations) shown in Figure 2.5.

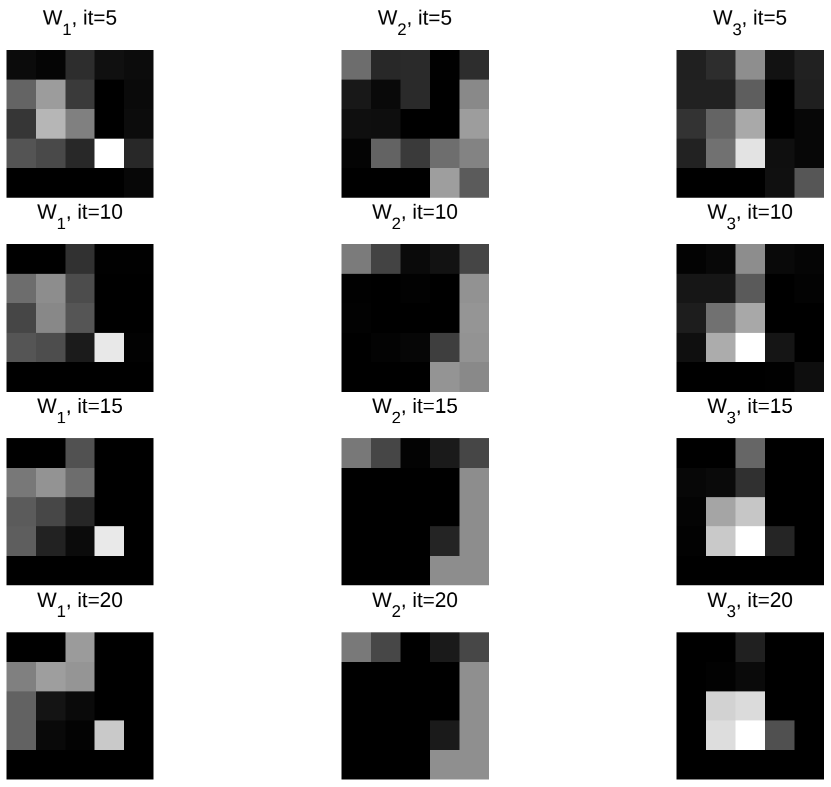


Figure 2.7: Motivating example: Progression of basic puzzles. Compare with Figure 2.5.

To aid convergence, we use the newest factors of $\mathbf{W}$ and $\mathbf{H}$ available. From an iterative standpoint, it may be helpful to write the update above as:

$$H_{aj}^{(t+1)} = H_{aj}^{(t)} Q_E \left(\mathbf{W}^{(t)}, \mathbf{X}^T, \mathbf{H}^{(t)} \right)_{aj} \tag{2.8}$$

$$W_{ia}^{(t+1)} = W_{ia}^{(t)} Q_E \left(\mathbf{X}^T, \mathbf{H}^{(t+1)}, \mathbf{W}^{(t)} \right)_{ia} \tag{2.9}$$

This representation explicitly reveals the multiplication of the previous factor by a measure of quality

$$Q_E(\mathbf{A}, \mathbf{B}, \mathbf{C}) \equiv \frac{[\mathbf{A}^T \mathbf{B}^T]_{ia}}{[\mathbf{A}\mathbf{C} \diamond \mathbf{C}^T]_{ia}}, \tag{2.10}$$

where $\mathbf{AC} \diamond \mathbf{C}^T$ signifies the "correct" right or left multiplication of $\mathbf{AC}$ by $\mathbf{C}^T$. Under these updates, the Euclidean distance objective function Θ_{NMF_E} has been proven [34]

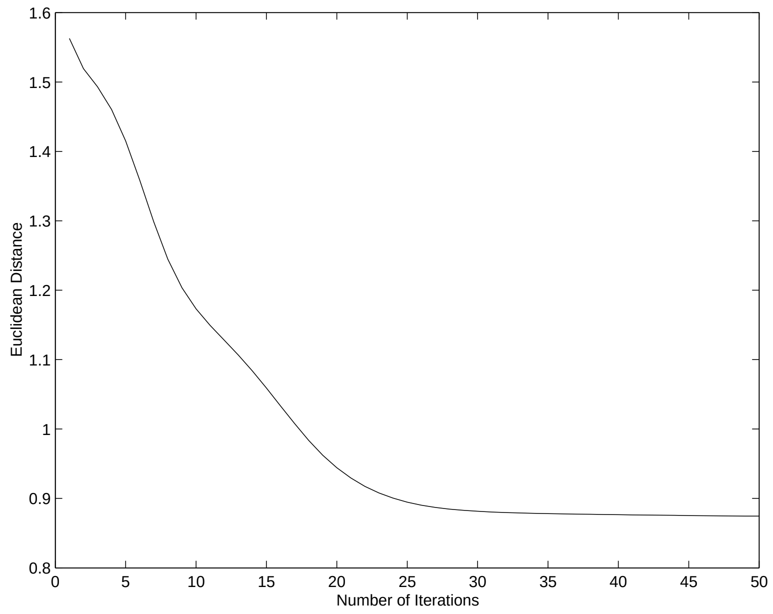


Figure 2.8: Motivating example: Θ_{NMF_E} objective function behavior.

to be monotonically decreasing:

$$\Theta_{NMF_E}\left(\mathbf{W}^{(t+1)}, \mathbf{H}^{(t+1)}\right) \leq \Theta_{NMF_E}\left(\mathbf{W}^{(t)}, \mathbf{H}^{(t)}\right) \qquad \text{for}\, t = 0, 1, \ldots \tag{2.11}$$

Figure 2.8 illustrates the monotone behavior of the objective function Θ_{NMF_E} for the puzzles in the motivating example. Here we see that after approximately 25 iterations, the distance has essentially reached its asymptotic behavior and the factors **W** and **H** become stable. A similar nonincreasing objective function plot may also be obtained using the Divergence or Local NMF algorithms introduced next.

2.2.3.2 Divergence Algorithm

The second objective function that is commonly used in practice is called the divergence, or entropy, measure:

$$\Theta_{NMF_D}(\mathbf{W}, \mathbf{H}) \equiv \text{Div}(\mathbf{X}\|\mathbf{WH}) \equiv \sum_{i=1}^{m}\sum_{j=1}^{n}\left(X_{ij}\log\frac{X_{ij}}{\sum_{l=1}^{r} W_{il}H_{lj}} - X_{ij} + [\mathbf{WH}]_{ij}\right) \tag{2.12}$$

The objective function $\Theta_{NMF_D}(\mathbf{W},\mathbf{H})$ is not a distance measure because, strictly speaking, it is not symmetric in $\mathbf{X}$ and $\mathbf{WH}$. Further motivation behind this objective function can be seen when the columns of $\mathbf{X}$ and the columns of the approximation $\mathbf{WH}$ sum to 1. In this case, $\Theta_{NMF_D}(\mathbf{W},\mathbf{H})$ reduces to the Kullback-Leibler information measure used in probability theory [34]. This objective function is related to the likelihood of generating the columns in $\mathbf{X}$ from the basis $\mathbf{W}$ and encoding coefficients $\mathbf{H}$.

Again, this objective function equals its lower bound of zero only when we have strict equality, $\mathbf{X} = \mathbf{WH}$. To balance complexity and speed, the following update rules are commonly used:

$$H_{aj} \leftarrow H_{aj} \sum_i [\mathbf{W}^T]_{ai} \frac{X_{ij}}{[\mathbf{WH}]_{ij}} \tag{2.13}$$

$$W_{ia} \leftarrow W_{ia} \sum_j \frac{X_{ij}}{[\mathbf{WH}]_{ij}} [\mathbf{H}^T]_{ja} \tag{2.14}$$

$$W_{ia} \leftarrow \frac{W_{ia}}{\sum_j W_{ja}} \tag{2.15}$$

where the subscripts again indicate element by element division or multiplication. Writing the above update (minus the final basis normalization) in iterative form, we have:

$$H_{aj}^{(t+1)} = H_{aj}^{(t)} Q_D \left(\left[\mathbf{W}^{T\,(t)}\right]_{ai}, \frac{X_{ij}}{[\mathbf{W}^{(t)}\mathbf{H}^{(t)}]_{ij}} \right)_{aj} \tag{2.16}$$

$$W_{ia}^{(t+1)} = W_{ia}^{(t)} Q_D \left(\frac{X_{ij}}{[\mathbf{W}^{(t)}\mathbf{H}^{(t+1)}]_{ij}}, \left[\mathbf{H}^{T\,(t+1)}\right]_{ja} \right)_{ia} \tag{2.17}$$

where

$$Q_D(\mathbf{A},\mathbf{B})_{ij} \equiv \sum_k A_{ik} B_{kj} = (\mathbf{AB})_{ij}, \tag{2.18}$$

Following [34], we will call the procedure that uses the above update rules to minimize $\Theta_{NMF_D}(\mathbf{W},\mathbf{H})$ the Divergence Algorithm. Lee and Seung have proven in [35] that this algorithm monotonically decreases the objective function Θ_{NMF_D}:

$$\Theta_{NMF_D}\left(\mathbf{W}^{(t+1)},\mathbf{H}^{(t+1)}\right) \leq \Theta_{NMF_D}\left(\mathbf{W}^{(t)},\mathbf{H}^{(t)}\right) \qquad \text{for } t = 0, 1, \ldots \tag{2.19}$$

2.2.3.3 Local NMF Algorithm

A recently proposed refinement [36] of NMF is a slight variation of the Divergence Algorithm detailed above. Local Non-negative Matrix Factorization (LNMF) has an objective function which seeks to impose constraints on the spatial locality of the features of a data set:

$$\Theta_{NMF_L}(\mathbf{W},\mathbf{H}) \equiv \sum_{i=1}^{m}\sum_{j=1}^{n}\left(X_{ij}\log\frac{X_{ij}}{[\mathbf{WH}]_{ij}} - X_{ij} + [\mathbf{WH}]_{ij} + \alpha U_{ij}\right) - \beta\sum_{i} V_{ii}, \tag{2.20}$$

where α, $\beta > 0$ are some constants, $\mathbf{U} = \mathbf{W}^T\mathbf{W}$ and $\mathbf{V} = \mathbf{H}\mathbf{H}^T$. This objective function is the Divergence objective function Θ_{NMF_D} with three additional terms:

(1) Minimize the number of basis components (columns of $\mathbf{W}$) required to represent $\mathbf{X}$. Given the normalization $\sum_i W_{ij} = 1 \quad \text{for } j = 1, \dots, r$, we wish to minimize $\sum_i W_{ij}^2$ so that the basis contains as many non-zero elements as possible. This is accomplished by minimizing $\sum_i U_{ii}$.

(2) Make the basis vectors as orthogonal as possible in order to minimize redundancies. This is equivalent to minimizing $\sum_{i \neq j} U_{ij}$.

(3) Retain the components that give the most important information. Maximize the total "activity" of each component that is kept by maximizing $\sum_i V_{ii}$.

A set of update rules that minimize this objective function are:

$$H_{aj} \leftarrow \sqrt{H_{aj}\sum_i [\mathbf{W}^T]_{ai}\frac{X_{ij}}{[\mathbf{WH}]_{ij}}} \tag{2.21}$$

$$W_{ia} \leftarrow W_{ia}\sum_j \frac{X_{ij}}{[\mathbf{WH}]_{ij}}[\mathbf{H}^T]_{ja} \tag{2.22}$$

$$W_{ia} \leftarrow \frac{W_{ia}}{\sum_j W_{ja}} \tag{2.23}$$

The structure of this LNMF update for $\mathbf{W}$ is identical to that of the Divergence Algorithm update, differing only in the coefficient matrix $\mathbf{H}$ used. The update for $\mathbf{H}$ now uses an element by element square root to satisfy the three additional constraints detailed above.

Writing the above update (minus the final basis normalization) in iterative form, we have:

$$H_{aj}^{(t+1)} = \sqrt{H_{aj}^{(t)} Q_D \left(\left[\mathbf{W}^{T\,(t)}\right]_{ai}, \frac{X_{ij}}{[\mathbf{W}^{(t)}\mathbf{H}^{(t)}]_{ij}} \right)_{aj}} \tag{2.24}$$

$$W_{ia}^{(t+1)} = W_{ia}^{(t)} Q_D \left(\frac{X_{ij}}{[\mathbf{W}^{(t)}\mathbf{H}^{(t+1)}]_{ij}}, \left[\mathbf{H}^{T\,(t+1)}\right]_{ja} \right)_{ia} \tag{2.25}$$

where $Q_D(\mathbf{A}, \mathbf{B})_{ij} \equiv [\mathbf{AB}]_{ij}$ is the same quality function defined for the Divergence Algorithm. It has recently been proven [36] that the above update strategy results in a monotonically decreasing objective function:

$$\Theta_{NMF_L}\left(\mathbf{W}^{(t+1)}, \mathbf{H}^{(t+1)}\right) \leq \Theta_{NMF_L}\left(\mathbf{W}^{(t)}, \mathbf{H}^{(t)}\right) \qquad \text{for } t = 0, 1, \ldots . \tag{2.26}$$

Having introduced the framework for NMF, we now turn to the seemingly disjoint topic of clustering non-negative data sets.

2.3 Clustering with K-Means and Spherical K-Means

Recently, it has become increasingly desirable to restructure large data sets into smaller similar groups. The high dimensionality involved in, for example, text data sets has fueled the demand for efficient clustering techniques. These clustering techniques can be organized into two main classes: those based on iterative centroid methods and those relying on the Singular Value Decomposition. Of the latter, the Principal Direction Divisive Partitioning (PDDP) [5] algorithm has recently been the most explored. In this thesis we will not further discuss PDDP but will instead focus on centroid-based methods which allow us to enforce certain constraints. We now introduce the primary centroid-based clustering method, K-Means.

2.3.1 K-Means

K-Means[6] (first introduced for clustering in the 1960's [17]) is the most widely used clustering technique [25]. The output of K-Means is a set $\{c_j\}_{j=1}^k$ of centroids in $\Re^m$ representing k disjoint subsets $\{\pi_j\}_{j=1}^k$ of the original data matrix $\mathbf{X}$. By disjoint we mean that:

$$\bigcup_{j=1}^{k} \pi_j = \{x_1, x_2, \dots, x_n\} = X \qquad \text{and} \qquad \pi_j \cap \pi_l = \emptyset \quad \text{for } j \neq l. \tag{2.27}$$

where x_i denotes the i-th column vector of the $m \times n$ data matrix $\mathbf{X}$. We will let Π_k denote the clustering defined by each of the k subsets π_j, $\Pi_k \equiv \{\pi_j\}_{j=1}^k$, where the subscript k explicitly defines the number of (non-empty) clusters obtained. The Stirling Numbers of the Second Kind [31] count the number of possible ways to partition an n-set into k cells. Accordingly, distributing the n vectors in $\mathbf{X}$ into k non-empty indistinguishable[7] subsets results in $S(n,k)$ possible outcomes, where

$$S(n,k) \equiv \frac{1}{k!}\sum_{j=0}^{k}(-1)^j \binom{k}{j}(k-j)^n \tag{2.28}$$

or alternatively,

$$S(n,k) = \frac{1}{k!}\sum_{\substack{x_1+\ldots+x_k=n \\ x_i \geq 1}} \binom{n}{x_1, x_2, \dots, x_k}. \tag{2.29}$$

The Stirling numbers quickly become very large as n increases. This dramatic increase is illustrated in Table 2.1. The problem of finding an optimal partition from this large set of possible partitions is NP-complete (See [19] and [28]) and greatly depends on the objective of the clustering.

[6] Sometimes alternatively referred to as the Lloyd-Max Algorithm.

[7] Neither the order of the clusters $\{\pi_j\}_{j=1}^k$ nor the order of the vectors within each cluster π_j matters.

n	**k**	$\approx S(n,k)$
8	4	1701
16	4	$1.71 \cdot 10^{8}$
32	4	$7.68 \cdot 10^{17}$
64	4	$1.42 \cdot 10^{37}$
128	4	$4.82 \cdot 10^{75}$
128	8	$9.77 \cdot 10^{110}$
128	16	$6.38 \cdot 10^{140}$

Table 2.1: Sample approximate Stirling numbers of the second kind.

2.3.1.1 Indicator Vector

We will now introduce some notation that will be used for the remainder of this section. We let $\iota(\Pi_k)$ denote an n-long indicator vector associated with clustering Π_k:

$$\iota_i \equiv j \quad \text{if} \quad x_i \in \pi_j. \tag{2.30}$$

Using this notation we can then explicitly define $\eta(\pi_j)$, the number of vectors contained in the j-th cluster, and c_j the centroid (or average) of the $\eta(\pi_j)$ vectors contained in the j-th cluster:

$$\eta(\pi_j) \equiv \sum_{i=1}^{n} \delta_{j,\iota_i} \quad \text{where} \quad \delta_{j,\iota_i} \equiv \begin{cases} 1 & \text{if} \quad j = \iota_i \\ 0 & \text{if} \quad j \neq \iota_i \end{cases} \tag{2.31}$$

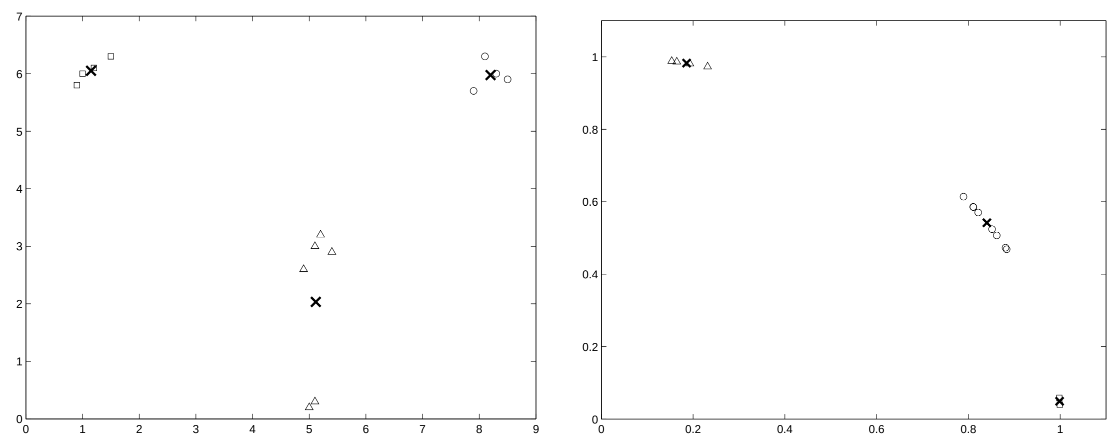
and

$$c_j \equiv \frac{1}{\eta_j} \sum_{i=1}^{n} \delta_{j,\iota_i} x_i. \tag{2.32}$$

When written this way, we can see that for any clustering Π_k of $\mathbf{X}$, the K-Means method requires only the storage of the n-long vector $\iota(\Pi_k)$. $\iota(\Pi_k)$ contains all of the information needed to uniquely define a clustering Π_k.

2.3.1.2 K-Means Objective Function

The K-Means method seeks to partition the data set $\mathbf{X}$ into k disjoint clusters so that each point in the cluster is “closer” to the centroid associated with that cluster



(a) (b)

Figure 2.9: Comparing clusterings: (a) K-Means, (b) Spherical K-Means.

than it is to the other $k-1$ centroids. A variety of metrics can be used to define the closeness of a point to a centroid but the (square of the) Euclidean Distance in $\Re^m$, $\|x_i - c_{\iota_i}\|_2^2$, is the most widely used and studied metric for K-Means ([6], [18]) . The natural objective function which K-Means seeks to minimize is then:

$$\Theta_{KM}(\Pi_k) \equiv \sum_{j=1}^{k} \sum_{x_i \in \pi_j} (x_i - c_j)^2 = \sum_{i=1}^{n} (x_i - c_{\iota_i})^2. \tag{2.33}$$

A K-Means algorithm should then monotonically decreases Θ_{KM} in search of the partitioning Π_k that minimizes this quantity. Algorithms that implement K-Means are usually iterative gradient descent methods which converge to local minima of Θ_{KM}. For more on the convergence of K-Means, we refer the reader to the study by Bottou and Bengio [6].

However, the traditional K-Means method and its Euclidean distance-based objective function Θ_{KM} have proven to be inappropriate for a variety of classification applications [15]. For example, consider the case of 14 web pages containing content on three different subjects. Projecting these 14 pages onto the plane yields the points shown in Figure 2.9 (a). The K-Means method (with $k = 3$) produced the three centroids (denoted by the $\times$'s) shown. However, in this case the three subjects were not correctly identified. The cluster in the upper right of the figure consists of essentially the same text as the four points in the center of the figure but with each word being twice as frequent. This simple example is indicative of many of the techniques that past web designers exploited to trick early search engines on the World Wide Web. The clusters in Figure 2.9 (b) resulted from the Spherical K-Means clustering introduced next and correctly correspond to the three subjects of the web documents.

2.3.2 Spherical K-Means

In 2001, Dhillon and Modha introduced the Spherical K-Means method [15] for clustering large sets of sparse text data. This method was established by taking advantage of some of the pre-processing often performed when working with text in a vector space model. In order to assign equal weight to each of the n points in a data set, the column vectors $x_1, x_2, \ldots, x_n$ of the original $m \times n$ data matrix $\mathbf{X}$ are normalized to be of unit length in the Euclidean norm.[8] The effect of this normalization is to only account for the direction of each vector only and not the length. We further restrict the data set $\mathbf{X}$ to consist entirely of non-negative elements, a natural property of most text and image vector space models.

Consequently, each point defined by the n column vectors in $\mathbf{X}$ will lie on the part of the unit sphere in the positive orthant of $\Re^m$ (Recall the two-dimensional case illustrated in Figure 2.9 (a)). It is this restriction, coupled with the non-negative data, which allows us to use the Cosine Similarity measure, $\cos(\theta_{x,y})$, to quantify the similarity of two vectors:

$$\cos(\theta_{x,y}) = \|x\|_2 \|y\|_2 \cos(\theta_{x,y}) =: x^T y \tag{2.34}$$

[8] The assumption of normalized columns of $\mathbf{X}$ will later be relaxed so that the methods described below will work on any non-negative matrix. We make this simplifying assumption for illustrative and historical purposes only.

where the first equality is a result of the normalization of x and y and the second follows from the definition of the standard inner product.

The Cosine Similarity provides a bounded $(0 \leq \cos(\theta_{x,y}) \leq 1)$[9] similarity measure where values closer to one indicate more similar vectors (and the smaller the angle $\theta_{x,y}$ between the two vectors x and y). Excluding normalization costs, this distance measure is less computationally expensive (fewer additions/subtractions for the same number of multiplications) than its Euclidean distance counterpart. Further, the calculation of an inner product is significantly aided by the sparsity that occurs naturally in many data sets.

For purposes of notation, we will henceforth assume that $\|\cdot\|$ denotes the Euclidean 2-norm except where indicated otherwise. In order to use the inner product to define the Cosine Similarity of a unit vector x_i and a centroid c_j, the Spherical K-Means method requires that each centroid is also of unit length:

$$c_j = \frac{\hat{c}_j}{\|\hat{c}_j\|} = \frac{\sum_{x_i \in \pi_j} x_i}{\|\sum_{x_i \in \pi_j} x_i\|}. \tag{2.35}$$

With this adjustment, we may now define the quality (or coherence) of cluster π_j with centroid c_j:

$$Q(\pi_j) \equiv \sum_{x_i \in \pi_j} x_i^T c_j = \sum_{i=1}^{n} \delta_{j,\iota_i} x_i^T c_j. \tag{2.36}$$

The Cauchy-Schwarz inequality then allows us to characterize the contribution of all vectors in a given cluster π_j:

$$\sum_{x_i \in \pi_j} x_i^T z \leq \sum_{x_i \in \pi_j} x_i^T c_j, \qquad \forall z \in \Re^m. \tag{2.37}$$

This confirms that, when combined, all of the vectors contained in a cluster are closer to the cluster's centroid than any other vector.

We extend the quality measure above to include all k clusters to arrive at the

[9] The values of $\cos(\theta_{x,y}) \in [-1, 0)$ are eliminated by the non-negativity of the data vectors x and y.

objective function which Spherical K-Means seeks to maximize:

$$\Theta_{SKM}(\Pi_k) \equiv \sum_{j=1}^{k} \sum_{x_i \in \pi_j} x_i^T c_j = \sum_{i=1}^{n} x_i^T c_{\iota_i} \tag{2.38}$$

Θ_{SKM} seeks to maximize the intracluster coherence while minimizing the intercluster coherence.

Geometrically, we note that the boundary between any two clusters is defined by the hypercircle (in the positive orthant) which is defined by the intersection of the unit hypersphere and a hyperplane through the origin. This plane runs through the bisector of the line segment between the two centroids. An illustration of this geometric interpretation is shown in Figure 2.10.

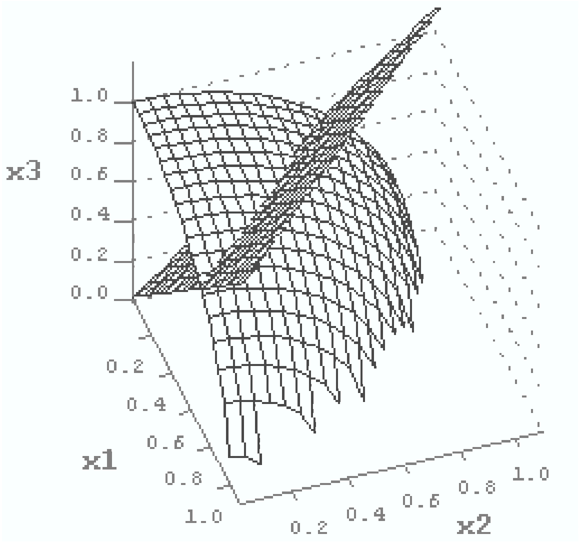


Figure 2.10: Geometric interpretation of cluster boundaries in three dimensions.

2.3.2.1 The Algorithm

We now summarize the Spherical K-Means algorithm which has been proven [15] to be nondecreasing in the objective function Θ_{SKM}:

(1) Initialize k centroids $\{c_j^{(0)}\}_{j=1}^{k}$, set $t = 0$.

(2) Compute $d_{ij}^{(t)} = x_i^T c_j^{(t)}$ for $i = 1, 2, \ldots, n$ and $j = 1, 2, \ldots, k$

(3) Define the new partitioning $\Pi_k^{(t+1)}$ by updating each cluster:

$$\pi_j^{(t+1)} = \{x_i \,|\, j = \arg\max_l d_{il}^{(t)}\}$$

(4) Recompute each centroid:

$$c_j^{(t+1)} = \frac{\sum_{x_i \in \pi_j^{(t+1)}} x_i}{\| \sum_{x_i \in \pi_j^{(t+1)}} x_i \|}$$

The initialization, convergence, and other computational considerations for the above algorithm are explored in the following chapter.

Chapter 3

Computational Considerations

In this chapter we introduce two data sets that will be used to illustrate computational results for the remainder of this thesis. With these test data sets introduced and the theoretical framework of Chapter 2 for Non-negative Matrix Factorization and Spherical K-Means in place, we then address further computational considerations for both techniques. We will explore the computational requirements, a necessary numerical correction, a factor normalization theorem and factor initialization for NMF algorithms. Next, we will detail the initialization, convergence, computational requirements and recent refinements of Spherical K-Means. Lastly, we introduce an important decomposition obtained by the Spherical K-Means clustering process called a concept decomposition.

3.1 Test Data Sets

Throughout this document we will refer to a set of test data sets/matrices on which all subsequent numerical computations will be performed. We now introduce both a facial image data set and a text data set. A third (physical) data set will be introduced and explored in Chapter 6.

3.1.1 Faces Data Set

The "Faces" data set represents a well known collection of faces [40] with several additions for a total of 382 facial images. Each square picture is 64 by 64 pixels, leading to a 4096×382 pixel by picture matrix $\mathbf{F}$. The face pictures have been cropped and centered so that each face's features are located in roughly the same area. Elements of noise such as baseball caps, glasses or a background to the side of a subject's chin are also present throughout the collection. Figure 3.1 shows 16 sample faces from the data set. Despite the relatively low number (when compared to image collections encountered in practice) of images in $\mathbf{F}$, the matrix is less than 0.1% sparse, and serves well as a computationally demanding test set.

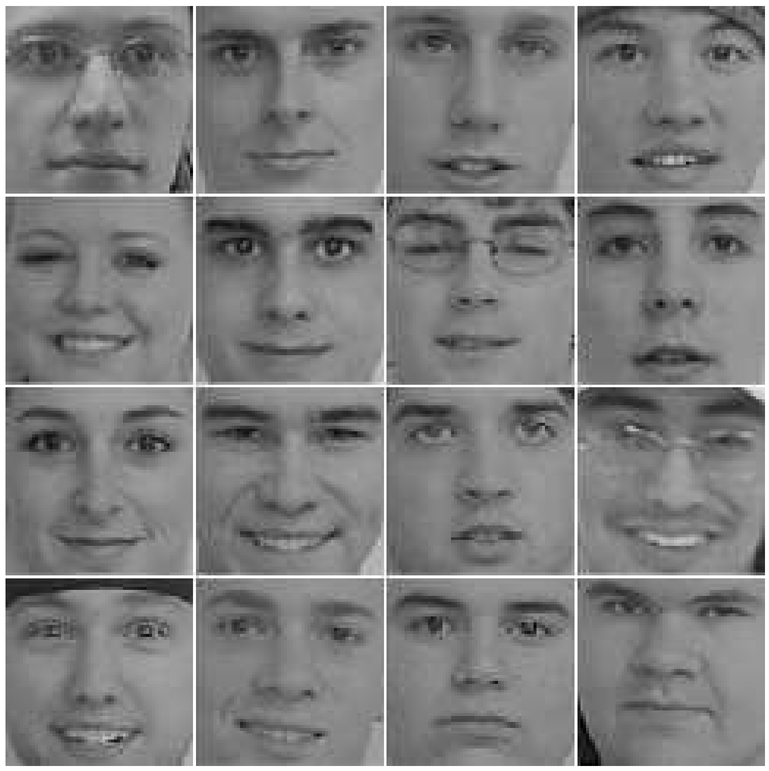


Figure 3.1: Sixteen sample columns (images of size 64×64) from the Faces data set.

acid	activity	air
analysis	automatic	bibliographic
blood	bodies	body
bone	book	books
boundary	buckling	cancer
case	cases	catalog
cell	cells	chemical
child	children	circular
classification	clinical	communication
computer	cost	current
cylinder	days	development
diagnosis	dimensional	disease
distribution	dna	document
documents	drag	edge
effect	effects	equation
equations	excretion	experimental

Table 3.1: 48 frequently occurring terms from the Classic3 data set.

3.1.2 Classic3 Data Set

The "Classic3" text data set [9] is often used as a standard test for information retrieval (IR) techniques. The set is a merger of three (Cisi, Cranfield and Medline) popular sets of abstracts from 1460 informational retrieval papers, 1400 aeronautical systems papers and 1033 medical journals, respectively. Table 3.1 shows an example of some of the most frequently occurring (non-stop list) words in the Classic3 collection. After preprocessing, 4099 unique terms remain. As is often the case in practice, the resulting 4099×3893 term-by-document matrix $\mathbf{T}$ is over 98.89% sparse (Recall Figure 2.2). While this data set is representative of high dimensional text sets, in general one expects larger data sets (like the World Wide Web) to have many more documents than terms ($m \gg n$).

Data Set	**Matrix**	**Rank**	**Dimensions (feature by point)**
FACES	$\mathbf{F}$	382	4096×382 (pixel by picture)
CLASSIC3	$\mathbf{T}$	3889	4099×3893 (term by document)
AVIRIS	$\mathbf{A}$	196	198×40000 (band by pixel)

Table 3.2: Summary of the three test data sets.

Table 3.2 summarizes the two data sets introduced here and the AVIRIS data set of Chapter 6 for comparison.

3.2 NMF Computational Considerations

A number of additional factors need to be considered before implementing any one of the three NMF algorithms introduced in Chapter 2. In this section we will detail the computational complexity of the three NMF algorithms and correct a shortcoming of the Divergence and Local NMF algorithms. We then establish a theorem which allows arbitrary normalization of the left factor **W**. Finally, we introduce the problem of NMF initialization which the latter parts of this document will focus on.

3.2.1 Computational Requirements

In order to quantify the computational requirements of the three algorithms introduced in Chapter 2, we now detail the number of floating point operations and storage required for each algorithm in Table 3.3.

Algorithm	**Multiplications**[1]	**Additions**[2]	**Storage**[3]
Euclidean Distance	$6mnr + 2r(m+n)$	$6mnr$	$r(m+n)$
Divergence	$4mnr + 2mn + r(2m+n)$	$4mnr - 2mn - rn$	$r(m+n)$
Local NMF[4]	$4mnr + 2mn + 2r(m+n)$	$4mnr - 2mn - rn$	$r(m+n)$

Table 3.3: Computational requirements of NMF algorithms.

3.2.2 Dividing by Zero

Both the Divergence and Local NMF algorithms require an element by element division by $[\mathbf{WH}]_{ij}$ in the updates for both factors **W** and **H**. This division may result in dividing $\frac{0}{0}$.[5] In order to have the optimal solution $\mathbf{X} = \mathbf{WH}$ be a fixed point in

[1] Here we neglect any terms that are linear or less in m, n and k ($\mathcal{O}(m,n,k)$).
[2] Here we neglect any terms that are linear or less in m, n and k ($\mathcal{O}(m,n,k)$).
[3] In addition to the mn required to store the original data matrix.
[4] Here we count a square root as one multiplication.
[5] To date, we have never seen a division of $\frac{x}{0}$ for any x other than $x = 0$.

these two algorithms, we define this quality coefficient to be 1 when it is indeterminant. This is a necessary remark because $\frac{0}{0}$ division becomes increasingly frequent for sparse $\mathbf{W}$ or $\mathbf{H}$ factors. These sparse factors are either obtained through initialization of the factors using sparse matrices or, more generally, as a result of a sparse initial matrix $\mathbf{X}$.

Seung has recently suggested (through personal correspondence) that these indeterminant entries may be corrected by replacing the element by element divisions of $\frac{\mathbf{X}}{\mathbf{WH}}$ in the Divergence algorithm (and hence the Local NMF algorithm) with the following numerical perturbation:

$$\frac{X_{ij}}{[\mathbf{WH}]_{ij}} \rightarrow \frac{[\mathbf{X}+\varepsilon]_{ij}}{[\mathbf{WH}+\varepsilon]_{ij}} \tag{3.1}$$

where ε is machine-epsilon. The comment here is that the addition of such a small number (≈ 0) to the both the original matrix $\mathbf{X}$ and the approximation $\mathbf{WH}$ numerically "affects" only the elements which are zero and allows the iteration to continue without encountering $\frac{0}{0}$. In practice, we have found that the modification in (3.1) significantly aids computations (avoids costly errors) while maintaining any sparsity inherent in the data.

3.2.3 Normalizing the Basis W

So far, the only constraint that we have placed on the factors $\mathbf{W}$ and $\mathbf{H}$ is nonnegativity. However, the development of the Divergence algorithm (and subsequently, the LNMF algorithm) introduces a normalization where all of the basis vectors in $\mathbf{W}$ are of unit length with respect to the one-norm. This normalization is required to maintain the proper form of the Kullback-Leibler information measure [18] and is easily enforced because of the linearity of the measure. However, the one-norm does not help us gain additional insight into the structure of the basis vectors and we wonder if other normalizations could be applied without losing the monotonicity of the two updates. We now introduce a theorem that allows one to enforce any desired normalization on the basis vectors in $\mathbf{W}$. Later, we will use the (Euclidean) two-norm but the following theorem is norm independent.

Theorem 1 Given a Non-negative Matrix Factorization $\tilde{\mathbf{X}} = \mathbf{WH}$ induced by the objective function Θ and any vector norm $\mathcal{N} = \|\cdot\|$, the basis factor $\mathbf{W}$ may be normalized to have columns of unit length in $\mathcal{N}$ without losing the monotone convergence of the generating objective function Θ.

Proof: First define a diagonal matrix $\mathbf{N}$ whose elements are the norms of each of the r columns of the basis factor $\mathbf{W}$:

$$\mathbf{N} \equiv \begin{bmatrix} \|w_1\| & 0 & \dots\dots\dots\dots\dots & 0 \\ 0 & \|w_2\| & 0 \;\; \dots\dots\dots\dots & 0 \\ \dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots \\ 0 & \dots & 0 \;\; \|w_i\| \;\; 0 \;\; \dots & 0 \\ \dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots \\ 0 & \dots\dots\dots\dots\dots\dots & 0 & \|w_r\| \end{bmatrix}. \tag{3.2}$$

Our strategy is to multiply $\mathbf{N}$ and its inverse $\mathbf{N}^{-1} = \operatorname{diag}\left(\frac{1}{\|w_1\|}, \frac{1}{\|w_2\|}, \dots, \frac{1}{\|w_r\|},\right)$ between the factors in each iteration of the NMF algorithm induced by Θ. Then:

$$\tilde{\mathbf{X}} = \mathbf{WH} = \mathbf{WN}^{-1}\mathbf{NH} = \hat{\mathbf{W}}\hat{\mathbf{H}} \tag{3.3}$$

where

$$\hat{\mathbf{W}} = \begin{bmatrix} | & | & & | \\ \frac{w_1}{\|w_1\|} & \frac{w_2}{\|w_2\|} & \dots & \frac{w_r}{\|w_r\|} \\ | & | & & | \end{bmatrix} \tag{3.4}$$

so that $\hat{\mathbf{W}}$ now contains columns of unit length in $\mathcal{N}$ and $\hat{\mathbf{H}}$ is scaled appropriately (by $\mathbf{N}$). In applying this normalization, we do not lose the equality in (3.3). Therefore, we conclude that the normalized factorization $\hat{\mathbf{W}}\hat{\mathbf{H}}$ obtained from any factorization $\mathbf{WH}$ that results in monotonicity of Θ maintains this monotonicity. $\square$

This theorem shows explicitly that the theoretical error of $\tilde{\mathbf{X}}$ is not affected by the normalization process. For the implementation of the algorithm, the above proof allows us to enforce a specified norm without requiring more operations for each iteration. We therefore benefit from the computational ease of enforcing the "natural" one-norm (L_1) in the $\mathbf{W}$ update before applying $\mathbf{N}^{-1}\mathbf{N}$ to the final factorization. At a cost of r norm calculations and two matrix-vector multiplications, we now have a basis with a nice geometric interpretation and have minimized the round-off error that repeated normalizations often create.

3.2.4 Factor Initialization

The initialization of Non-negative Matrix Factorizations remains an open problem which the second half of this thesis addresses in detail. A given NMF algorithm for an $m \times n$ data set $\mathbf{X}$ necessarily requires three inputs:

(1) A desired "rank" r

(2) An $m \times r$ basis initialization $\mathbf{W}^{(0)}$

(3) An $r \times n$ coefficient initialization $\mathbf{H}^{(0)}$

As a starting point, the first input, r, specifying the number of basis vectors, is usually chosen such that $r < \frac{mn}{m+n}$ so that the lower dimensional representation requires less storage than the initial data set. Beyond this storage constraint, there is currently no way of determining an "optimal" r for a specific type of data. For truly low dimensional vector space model representations, r is usually chosen to be between 5 and 50 [1]. In practice, when $\max(m,n) \gtrapprox 10^4$, r is usually chosen to be between 50 and 100. However, for very large problems, an r value in the hundreds may provide a significant reduction in storage while maintaining the "quality" of the approximation. Due to the computational expense of a single iteration of an NMF algorithm, it is usually extremely expensive (and often inappropriate, as a result of the general convergence rates of the algorithms) to try a large[6] number of different r values in an attempt to balance the trade-off between storage, computation time and approximation/basis quality.

The updates introduced earlier are guaranteed (independent of the selection of $\mathbf{W}^{(0)}$ and $\mathbf{H}^{(0)}$) to monotonically decrease their respective objective functions. Recall the way in which we previously represented the factor updates as multiplying the previous factor by a measure of quality. For example, the update for $\mathbf{W}$ in the Divergence Algorithm may be written as

$$W_{ia}^{(t+1)} = W_{ia}^{(t)} Q_D \left(\frac{X_{ij}}{[\mathbf{W}^{(t)}\mathbf{H}^{(t+1)}]_{ij}}, \left[\mathbf{H}^{T\,(t+1)}\right]_{ja} \right)_{ia}.$$

We repeat this to emphasize one important consideration for factor initializations. The above representation makes it apparent that once a zero has been obtained in the i, j position of a factor, it will remain a zero. In fact, very sparse initializations may lead to “worse” local minima of the objective functions. Currently, there has been very little work done on determining a good initialization for $\mathbf{W}$ or $\mathbf{H}$ and it is standard to initialize Non-negative Matrix Factorizations with full (non-sparse) random matrices $\mathbf{W}^{(0)}$ and $\mathbf{H}^{(0)}$. It is the intent of this document to propose alternative initializations.

3.3 Spherical K-Means Computational Considerations

A number of factors need to be considered before numerically computing the Spherical K-Means clustering Π_k. In this section we will detail the necessary initial parameters required by the Spherical K-Means algorithm, illustrate the algorithm's convergence and then refine the algorithm's implementation. We end by defining an important matrix decomposition, called a “concept decomposition,” based on the centroids output by Spherical K-Means.

[6] For many large problems, even performing 500 iterations for a handful of different r values can take several days. In theory (as a result of the CPU architecture), a 266 Mhz Pentium machine would require 2 full days to complete 500 iterations of the Divergence NMF algorithm on the Classic3 data set for five different r values where $r \approx 50$. (This computation requires more than 10 trillion floating point operations.)

3.3.1 Initialization of Spherical K-Means

A significant and non-trivial part of the Spherical K-Means algorithm (and its general counterpart, K-Means [7]) is choosing k, the number of disjoint clusters to partition the data into. Similar to the choice of r in a Non-negative Matrix Factorization, the selection of k for Spherical K-Means remains an open and largely unexplored topic. This is largely because of the varying goals that one may have when clustering and the strong dependence of a "latent rank"[7] on the type and structure of the given data. In very high dimensions, it can be impossible to visualize the data set in a way that would reveal an optimal value for k. In practice, the very unrestrictive bound of $k \ll \min(m, n)$ is usually chosen as a starting point. Practical approaches to select k are analogous to those used in Principal Component Analysis where a k value is chosen based on the changing slope of the objective function [26]. For purposes of discussion we will therefore choose a few different values of k for each test set.

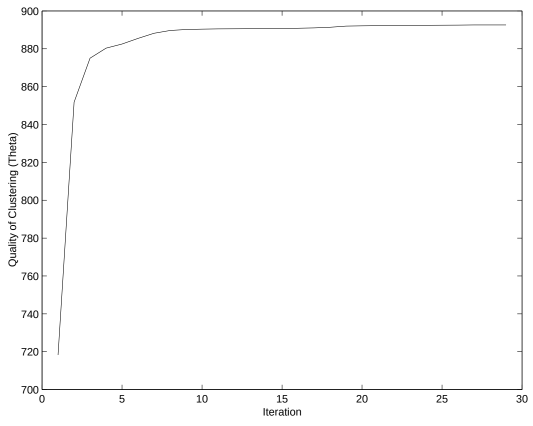
Once the number of clusters has been determined, the Spherical K-Means algorithm needs a centroid initialization for each cluster, $\{c_j^{(0)}\}_{j=1}^{k}$. One method of initializing these centroids is to (randomly) choose each centroid to be a different column vector from the data set. This initialization attempts to avoid a common difficulty of many K-Means techniques that occurs when an initialized cluster ends up containing no points after some iteration.[8] It should be noted that the Spherical K-Means initialization has a nontrivial impact on the resulting clustering scheme because Spherical K-Means (like other gradient ascent methods) is prone to getting trapped in local maxima when searching for the true global maximum of the given objective function [15]. Other suggested initializations include random initialization and perturbing the centroid of the

[7] By latent rank we mean the "rank" that is (usually) much smaller than the true rank of the data matrix but still retains the "majority" of the significant information. The problem of discovering a latent rank is analogous to tracking the decay of the magnitude of eigenvalues [22] in other rank reduction techniques.

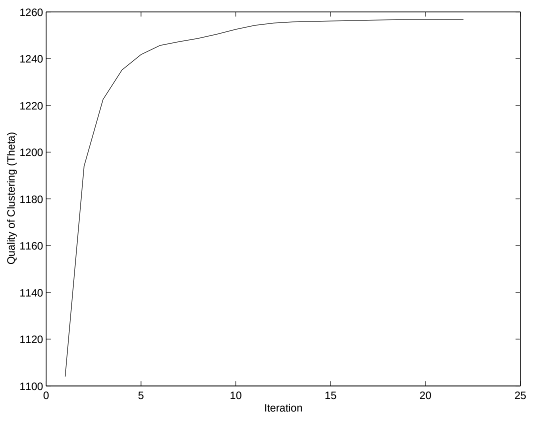
[8] Once a cluster becomes empty, it no longer has a set of points from which to obtain a cluster centroid and therefore will remain empty for the remainder of the clustering process (unless re-initialized).

whole data set. However, both of these methods cannot assure that the resulting clusters will be nonempty. Therefore, we will initialize each centroid by a randomly chosen column vector of $\mathbf{X}$.

3.3.2 Convergence



(a) $k = 10$



(b) $k = 50$

Figure 3.2: Sample convergence of Spherical K-Means using the Classic3 data set.

The stopping criteria for the (iterative) Spherical K-Means algorithm may be based on the change of the objective function, the number of points changing clusters, or a specified maximum number of iterations or objective function value. The monotonicity of Θ_{SKM} obtained with this algorithm is illustrated in Figure 3.2. This figure shows the quality of the Spherical K-Means clustering at each iteration for the Classic3 data set $\mathbf{T}$ for $k = 10$ and $k = 50$.

We now wish to explore[9] the number of iterations required for the Spherical K-Means algorithm to maximize the objective function Θ_{SKM}. Figure 3.3 (a) illustrates the number of iterations required for the above algorithm to converge to a local maximum for the Classic3 data set $\mathbf{T}$ with k values 1 through 75. By converge, here we mean

[9] Such results are previously unpublished.

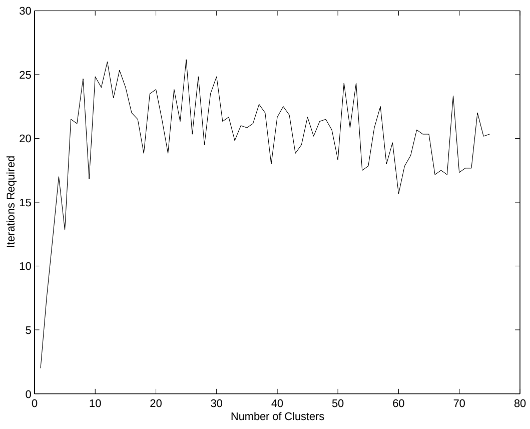
Number of Clusters (k)	**Number of Iterations**	**Average**
4	12, 12, 13, 17, 23, 25	17.00
5	9, 11, 12, 14, 14, 17	12.83
6	14, 16, 19, 20, 28, 32	21.50
36	17, 17, 19, 21, 22, 31	21.17
37	13, 16, 21, 21, 28, 37	22.67
38	16, 20, 21, 23, 25, 27	22.00
73	16, 16, 21, 25, 27, 27	22.00
74	17, 17, 18, 18, 22, 29	20.17
75	15, 17, 19, 19, 22, 30	20.33

Table 3.4: Sample relationship between number of clusters and iterations required for the Spherical K-Means algorithm to converge.

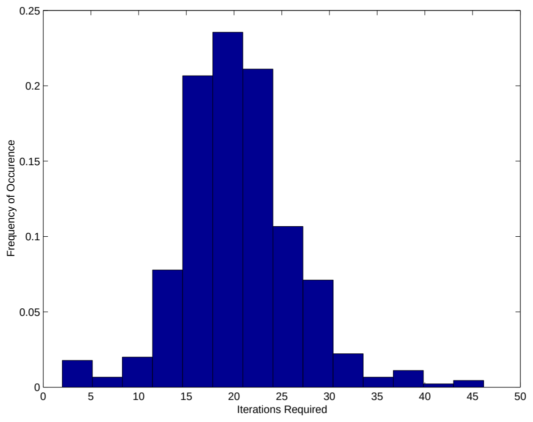
that all n documents in $\mathbf{T}$ have settled into their final cluster and will no longer change clusters in future iterations. Consequently, the minimum number of iterations to obtain convergence will be 2. One iteration is required to partition the data set and a second is required to verify that the partition remains unchanged. This is the case for $k = 1$, where the one centroid is just the normalized mean of the n points. Because of the significant variation in the number of iterations required, Figure 3.3 (a) represents the average number of iterations required for six different clusterings Π_k (resulting from six different random initializations). Table 3.4 shows some sample sets of the number of iterations required for the six clusterings.

In general, we have found no correspondence between the number of iterations required to converge and the number of clusters k. In the specific case of the Classic3 data set $\mathbf{T}$, approximately the same number (20) of iterations of the Spherical K-Means algorithm were required for any number of clusters greater than three. The frequency histogram of the number of iterations required for $6 \cdot 75 = 450$ clusterings obtained in Figure 3.3 (b) is intended to show the order of iterations that a large problem (4099×3893) requires to converge.[10]

[10] Compare the order of iterations needed here with the order of iterations needed for NMF convergence (see for example Figure 4.6 (b)).



(a)



(b)

Figure 3.3: Number of iterations required for convergence of Spherical K-Means on Classic3 data set: (a) $k = 1$ through 75, (b) Frequency histogram.

3.3.3 Comparison Refinement

The vast majority of the computations required by the Spherical K-Means algorithm occurs when the $n \cdot k$ comparisons (scalar products in $\Re^m$) are made between the n column vectors $x_1, \dots, x_n$ and the k centroids $c_1, \dots, c_k$. However, we notice (recall Figure 3.2) that after several iterations, the objective function increases, and therefore the number of points changing clusters becomes smaller and smaller. Motivated by the behavior illustrated in Figure 3.2, the "Comparison Refinement" [12] was proposed to empirically reduce the number of operations required for later iterations.

For normalized column vectors ($\|x_i\| = 1$), we notice that there is an upper bound on the change of the Cosine Similarity measure at the t-th iteration:

$$\left| x_i^T c_j^{(t-1)} - x_i^T c_j^{(t)} \right| \leq \|x_i\| \left\| c_j^{(t-1)} - c_j^{(t)} \right\| = \left\| c_j^{(t-1)} - c_j^{(t)} \right\|. \tag{3.5}$$

This inequality leads to an upper bound for the Cosine Similarity measure between x_i and the new set of centroids:

$$x_i^T c_j^{(t)} \leq x_i^T c_j^{(t-1)} + \left\| c_j^{(t-1)} - c_j^{(t)} \right\|. \tag{3.6}$$

We use this information to form an $n \times k$ upper bound matrix $\mathbf{U}_0$ whose i,j-th entry is the bound in (3.6). At each iteration, we are then no longer required to compute the $n \cdot k$ inner products $x_i^T c_j^{(t)}$ for $j = 1, \dots, k$ and $i = 1, \dots, n$. At each iteration we only compute the similarity between x_i and its last associated centroid $c_{\iota_i}^{(t)}$ and check whether this similarity is less than the upper bound in $\mathbf{U}$. If

$$x_i^T c_{\iota_i}^{(t)} < U_{i,j} = x_i^T c_j^{(t-1)} + \left\| c_j^{(t-1)} - c_j^{(t)} \right\| \tag{3.7}$$

it is necessary to compute the inner product $x_i^T c_j^{(t)}$ and we update the upper bound matrix $\mathbf{U}$ based on this new information: $U_{i,j} = x_i^T c_j^{(t)}$. This clever method [12] allows us to find $j = \arg\max_l x_i^T c_l^{(t)}$ by only checking the new inner products computed.

From an implementation standpoint, the Comparison Refinement is usually only begun after the centroids have begun to stabilize (when the number of points changing clusters slows down) as in Figure 3.2. The upper limit of the number of operations required (shown in the next section) at each iteration of the Spherical K-Means algorithm with the Comparison Refinement is larger than the upper limit for the original Spherical K-Means algorithm.[11] However, in practice, the number of operations required by the Comparison Refinement dramatically decreases as more iterations are computed (and the clusters stabilize). For extensive computational results of this refinement we refer the interested reader to [12].

3.3.4 Computational Requirements

In order to quantify the computational requirements of the Spherical K-Means Algorithm introduced above, we now detail the number of floating point operations and storage required for each iteration. Table 3.5 shows the computational requirements of the original Spherical K-Means algorithm as well as the Spherical K-Means Algorithm using the Comparison Refinement.

[11] This is due to the extra expense of computing $\left\| c_j^{(t-1)} - c_j^{(t)} \right\|$ for $j = 1, \dots, k$ at each iteration.

Algorithm[12]	**Multiplications**[13]	**Additions**	**Storage**[14]
Spherical K-Means	$mnk + mn + 2nk$	$mnk + 2mn$	$mk + n$
Com. Refinement[15]	$< mnk + mn + 2nk + mk$	$< mnk + 2mn + 2mk + nk$	$2mk + n$

Table 3.5: Computational requirements of the Spherical K-Means algorithm.

3.3.5 Unnormalized Data

We now loosen the restriction on the column vectors $x_1, \dots, x_n$ so that the Spherical K-Means algorithm may be applied to any non-negative matrix $\mathbf{X}$. Recall that we previously required that each column vector was of unit Euclidean length, $\|x_i\|_2 = 1$ for $i = 1, \dots, n$. Suppose now that $\mathbf{X}$ is a matrix consisting entirely of non-negative elements and with columns of varying length and that normalization is unfavorable (either because of the associated cost due to a large number of features, m, or because of round-off/precision concerns).

Recall that the original restriction was enforced for two primary reasons. First, normalization was required so that the Cosine Similarity measure (2.34) could be used to partition the data. From an implementation standpoint, this measure is used only to make a comparison between a column vector x_i and the k centroids (always defined to have unit length) $c_1, \dots, c_k$. Consider the scalar product of the unnormalized vector x_i and centroid c_j:

$$x_i^T c_j = \|x_i\|_2 \cos(\theta_{x_i,c_j}) \tag{3.8}$$

Since each centroid c_j is fixed until the end of an iteration, the new cluster that x_i belongs to will be based on which previous centroid represents the smallest departure (angle) from x_i. If $x_i \in \pi_l^{(t)}$ (with associated centroid $c_l^{(t)}$) at iteration t, then in the

[12] Here we neglect any terms that are linear or less in m, n and k ($\mathcal{O}(m,n,k)$).

[13] Here we count each comparison as one multiplication. Again, we neglect any terms that are linear or less in m, n and k ($\mathcal{O}(m,n,k)$).

[14] In addition to the mn required to store the original matrix. Storing an indicator of length n is optional but included here.

[15] Again, the "Comparison Refinement" initially requires more operations (multiplications and additions) than the original Spherical K-Means Method. However, after several iterations, it will require far fewer operations per iteration (see [12]).

next iteration x_i will only move to $\pi_j^{(t)}, j \neq l$ if:

$$\cos\left(\theta_{x_i, c_j^{(t)}}\right) - \cos\left(\theta_{x_i, c_l^{(t)}}\right) > 0 \tag{3.9}$$

or equivalently,

$$x_i^T \left[c_j^{(t)} - c_l^{(t)}\right] = \|x_i\|_2 \left[\cos\left(\theta_{x_i, c_j^{(t)}}\right) - \cos\left(\theta_{x_i, c_l^{(t)}}\right)\right] > 0. \tag{3.10}$$

In the Spherical K-Means algorithm this defines the new partitioning $\Pi_k^{(t+1)}$ by updating the set of clusters $\{\pi_j^{(t+1)}\}_{j=1}^k$ as follows:

$$\pi_j^{(t+1)} = \left\{x_i \,|\, j = \arg\max_l x_i^T c_l^{(t)}\right\} = \left\{x_i \,|\, j = \arg\max_l \bar{x}_i^T c_l^{(t)}\right\}. \tag{3.11}$$

where $\bar{x}_i$ represents the normalization (which need not be computed as the second equality shows) of x_i: $\bar{x} \equiv \frac{x_i}{\|x_i\|}$.

The second reason for our original restriction was because of the objective function $\Theta_{SKM}(\Pi_k)$ in (2.38). With the relaxation of the length of x_i, we have lost the monotonicity of the original objective function in the Spherical K-Means algorithm. This is not to say that we have lost the original objective function entirely. The Spherical K-Means algorithm with unnormalized columns in $\mathbf{X}$ will monotonically increase the objective function:

$$\Theta_{SKM'}(\Pi_k) \equiv \sum_{j=1}^{k} \sum_{x_i \in \pi_j} \frac{1}{\|x_i\|_2} x_i^T c_j. \tag{3.12}$$

Here, we do note that the length $\|x_i\|_2$ was explicitly needed for the calculation of $\Theta_{SKM'}(\Pi_k)$. However, for most applications, the objective function is only calculated for use as one possible stopping criteria. Using an alternative stopping criteria, such as the number of points that changed clusters in the previous iteration, eliminates the need for ever explicitly computing $\Theta'_{SKM}(\Pi_k)$ or $\|x_i\|$.

In summary, the original objective function is no longer appropriate because it would place more weight on the centroids closest to the column vectors with the longest lengths. When clustering unnormalized column vectors x_i, the Spherical K-Means algorithm places equal weight on each vector to arrive at an optimal partition. From the above statements we see that even though each vector is treated like it is on the unit hypersphere (because the cosine measure only requires the direction, and not the length, of x_i), the actual point may remain separated from this hypersphere.

3.3.6 Other Recent Refinements

Modha and Spangler have recently investigated the relationship between feature weighting in the vector space model and Spherical K-Means results [38]. By introducing a distortion measure, they find a method for determining the optimal (within computational and heuristic constraints) feature weighting. Modha and Spangler also attempt to provide a general framework for the feature spaces in the K-Means and Spherical K-Means clustering problems.

Dhillon, Guan and Kogan have also recently proposed the "Ping Pong Method" to push the Spherical K-Means algorithm out of particularly "bad" local maxima (see [14] and [13]). This method seeks to correct the problem of the Spherical K-Means algorithm converging to a clustering Π_k which locally maximizes $\Theta_{SKM}(\Pi_k)$ but fails to see neighboring objective function peaks which may be higher.[16] This kind of behavior (which, in general, occurs as a result of gradient ascent techniques) is especially nonnegligible for clusterings where the number of points in a cluster π_j is low.

The "Ping Pong Method" takes advantage of the computations already needed by the Spherical K-Means algorithm and alternates between Spherical K-Means iterations and first variation refinements. In a first variation refinement, a data point is removed from one cluster and added to another in an attempt to create a better (as measured by the objective function $\Theta(\Pi_k)$) partitioning that could not be reached through stan-

[16] The size of the theoretical gain in quality has not yet been quantified. Refer to Figure 3.5 for a specific example of the range of qualities obtained using the Spherical K-Means algorithm.

dard Spherical K-Means iteration. Some work [14] has also been done in making a chain of data point movements by following the Kernighan-Lin heuristic used in graph partitioning.

While not significantly increasing the computational requirements of the Spherical K-Means algorithm, the "Ping Pong Method" can reduce the number of clusters. For our purposes, it is critical to avoid this degeneracy so that the final number of clusters is equal to the initial cluster number parameter k. Acknowledging that the proposed refinement would be guaranteed to do no worse [13] than the standard Spherical K-Means, in this document we do not implement the "Ping Pong Method."

With the Spherical K-Means algorithm now properly explored, we turn to an important matrix approximation based on the Spherical K-Means centroids.

3.3.7 Concept Decompositions

Suppose that we want to compute a matrix approximation based on the clustering Π_k obtained with Spherical K-Means. The simplest way to obtain such an approximation is to use the final Spherical K-Means centroids $\{c_j\}_{j=1}^k$ and the indicator vector $\iota(\Pi_k)$ to define a $m \times k$ "concept matrix" $\mathbf{C}$ and a sparse $k \times n$ matrix Δ:

$$\mathbf{C} \equiv \begin{bmatrix} | & | & & | \\ c_1 & c_2 & \dots & c_k \\ | & | & & | \end{bmatrix} \tag{3.13}$$

and

$$\Delta_{i,j} \equiv \begin{cases} 1 & \text{if} \quad \iota_j = i \\ 0 & \text{else} \end{cases} \qquad \text{for} \quad j = 1, \dots, n \tag{3.14}$$

We then define the elementary approximation $\hat{X}$ as the product of these two matrices:

$$\mathbf{X} \approx \hat{\mathbf{X}} \equiv \mathbf{C}\Delta. \tag{3.15}$$

Following Dhillon and Modha [15], we assume that the k centroids $\{c_j\}_{j=1}^k$ that make up the columns of $\mathbf{C}$ are linearly independent[17] so that $\hat{X}$ represents a rank k approximation of $\mathbf{X}$. Intuitively, this approximation replaces every column vector x_i with its associated cluster centroid c_j, where $x_i \in \pi_j$. However, as some of the n vectors in $\mathbf{X}$ are located near the intersection of two or more clusters, we are motivated to use linear (instead of binomial) combinations of the centroids in $\mathbf{C}$. The problem of finding the best linear combination is then a least squares projection problem resulting in the following approximation:

$$\mathbf{X} \approx \tilde{\mathbf{X}} \equiv \mathbf{CZ} \qquad \text{where} \quad \mathbf{Z} \equiv \left(\mathbf{C}^T\mathbf{C}\right)^{-1}\mathbf{C}^T\mathbf{X}, \tag{3.16}$$

where the existence of the inverse of $\mathbf{C}^T\mathbf{C}$ depends on the linear independence of the columns of $\mathbf{C}$. The reader should also note that the elements in the least squares solution $\mathbf{Z}$ are unconstrained in sign.

The approximation $\tilde{\mathbf{X}}$ is guaranteed to be better than the previous approximation $\hat{\mathbf{X}}$ in the Frobenius norm because the new right factor, $\mathbf{Z}$, is the closed form minimal solution to $\|\mathbf{X} - \mathbf{CZ}\|_F$. Dhillon and Modha have provided a firm empirical base for the "Concept Decompositions" defined above and we refer the reader to [15].

For illustrative purposes, we have compared the error of the low rank Concept Decompositions[18] with that of the best low rank approximation obtained using the truncated Singular Value Decomposition in Figure 3.4. Shown are the errors of the rank 1 through 75 approximations of the Classic3 matrix $\mathbf{T}$ (originally rank 3889). For comparison, we have also included the corresponding Non-negative Matrix Factorization results introduced in Figure 2.3.

Before we proceed, we briefly note the relationship between the quality of the

[17] In theory, the centroids may be linearly dependent, however, we have found that (especially for the $k \ll \min(m, n)$ used in practical applications) the assumption of linear independence of the centroids holds empirically.

[18] Because Spherical K-Means at best obtains local maxima of $\Theta_{SKM}(\Pi_k)$, one can expect that the actual global maximum will be slightly better. Shown (Figure 3.4) are the better of two clusterings for each rank with a maximum number of iterations of 100.

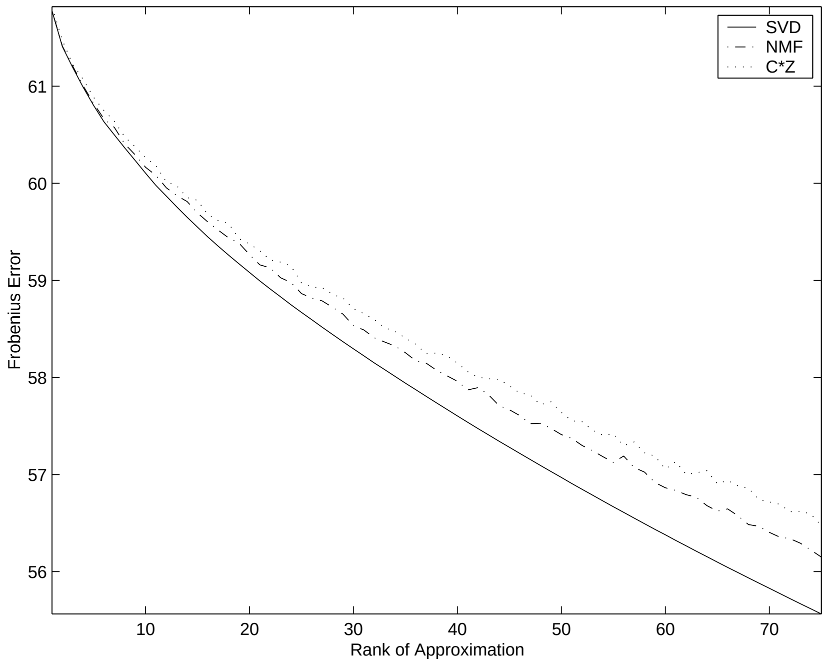


Figure 3.4: Comparing concept decompositions with the SVD and NMF.

clustering obtained by the Spherical K-Means algorithm (as measured by the objective function $\Theta_{SKM}(\Pi_k)$) and the error made by the resulting concept decomposition (as measured by the Frobenius error, $\mathbf{E}_k \equiv \|\mathbf{X} - \mathbf{CZ}\|_F$). In illustrating this relationship, we acknowledge that the "best" clustering does not necessarily lead to the "best" concept decomposition even though the clustering directly provides the left factor $\mathbf{C}$. This discrepancy arises because the basis provided, in this case, $\mathbf{C}$, is not the optimal rank k basis for the subsequent least squares problem. The basis provided by the Spherical K-Means clustering is based on centroids of strictly disjoint clusters and does not take into account the linear combinations of centroids which will result in a better approximation.

Despite this pitfall, we have found that the quality of a clustering is a good indicator of the error of its associated concept decomposition. For example, consider the clustering of the Faces data set $\mathbf{F}$ into $k = 10$ disjoint clusters. Using 1150 random initializations (to be further detailed in the next section) we computed both the quality of the clustering obtained (using Θ_{SKM}) and the Frobenius error, $\mathbf{E}_k$, of the resulting concept decomposition. The results are shown in Figure 3.5 where we see that "good" clusterings do tend to produce "good" concept decompositions.

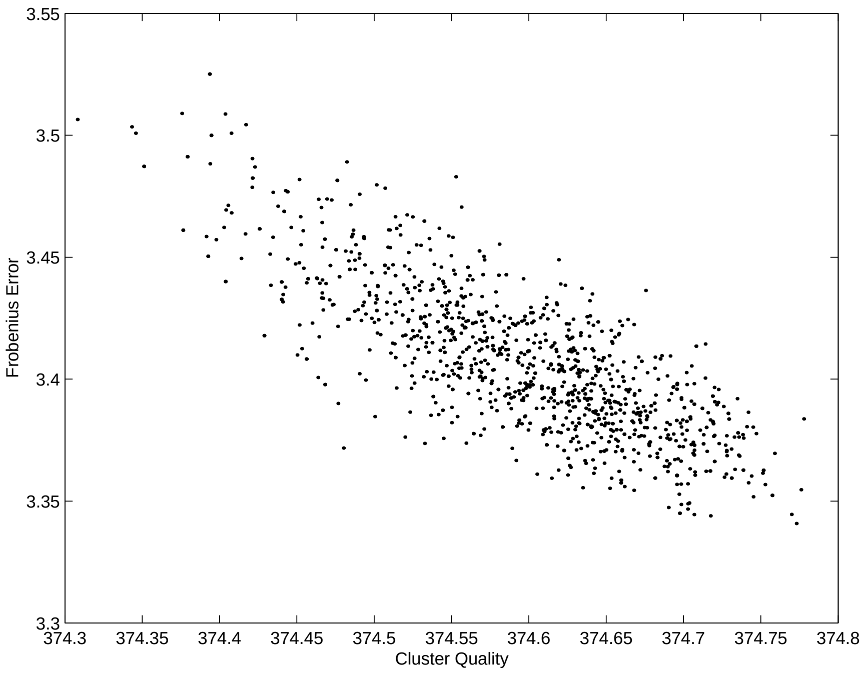


Figure 3.5: Comparing quality of Spherical K-Means clustering with error of concept decompositions.

Chapter 4

NMF and Spherical K-Means Examples Using Faces Data Set

In this chapter we will illustrate the output obtained from Non-negative Matrix Factorization and Spherical K-Means clustering for the Faces data set introduced in Chapter 3. Recall that the Faces data set consisted of 382 photographs of faces (64×64 pixels) stored in the 4096×382 vector space model $\mathbf{F}$. Using NMF, we first wish to represent these 382 faces as an additive combination of 24 different "basis faces", effectively reducing the dimension of $\mathbf{F}$ from 382 to 24. We will show the 24 basis faces that result from each of the three NMF algorithms introduced earlier. We will also illustrate the convergence of each algorithm by plotting the progress of the Frobenius error[1] of the approximation. We will run a maximum of 4000 iterations for each technique to be confident that we are displaying the true long-term behavior.[2] Next, we display the clustering results for the Spherical K-Means algorithm using $k = 24$ clusters. We will show the 24 resulting "centroid faces" and compare their structure and number of iterations needed to the NMF results.

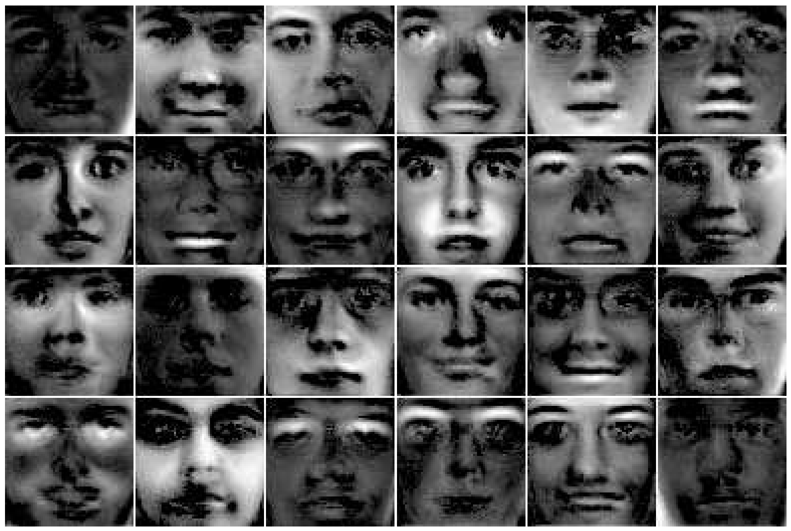


Figure 4.1: Euclidean distance algorithm: basis faces ($r = 24$).

4.1 NMF Examples Using Faces Data Set

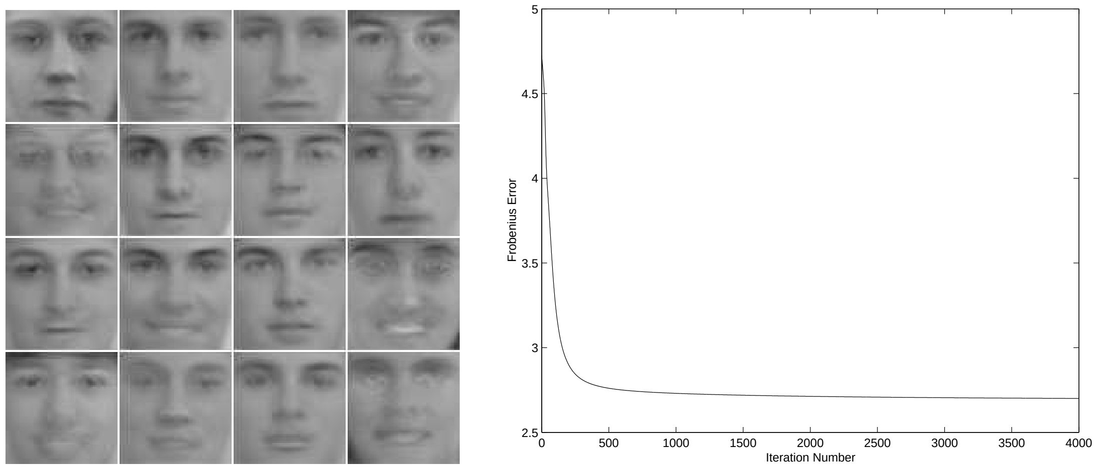
4.1.1 Euclidean Distance Algorithm

Figure 4.1 shows the 24 faces contained in the basis $\mathbf{W}$ that results from an implementation[3] of the Euclidean distance algorithm. These faces appear to look like what one expects to see from taking averages of several faces with certain features (i.e. eyebrows) highlighted. We do note that without a Non-negative Matrix Factorization, the lower dimensional basis would be impossible to visualize. Conceptually, the 24 faces shown are what the Euclidean distance NMF algorithm found to be the 24 faces (average faces) that the entire collection of 382 faces could best (in the Frobenius norm) be written as using only positive combinations. The resulting low rank (rank 24, compared to the original $rank(\mathbf{F}) = 382$) approximations for the first 16 faces in $\mathbf{F}$ are shown in

[1] The Euclidean distance algorithm is the only one of the three algorithms which is guaranteed to monotonically decrease the Frobenius error. However, for comparison, we have found that this error is also systematically decreased by the other two algorithms.

[2] Using the same (random) initializations of $\mathbf{W}$ and $\mathbf{H}$ for each of the three algorithms. 4000 iterations is significantly more iterations than is used in practice, but we use this number here to truly capture the long-term (stabilized) behavior of the factorizations.

[3] We stress that this implementation is only unique once initializations of $\mathbf{W}$ and $\mathbf{H}$ are specified. Again, this is true for all three algorithms tested.



(a) Approximation (b) Frobenius Error

Figure 4.2: NMF with the Euclidean distance algorithm.

Figure 4.2 (a) (Compare with the 16 of the 382 original faces in Figure 3.1). Due to the objective function used, this algorithm provides the rank 24 approximation with the smallest Frobenius error. This error is shown in Figure 4.2 (b).

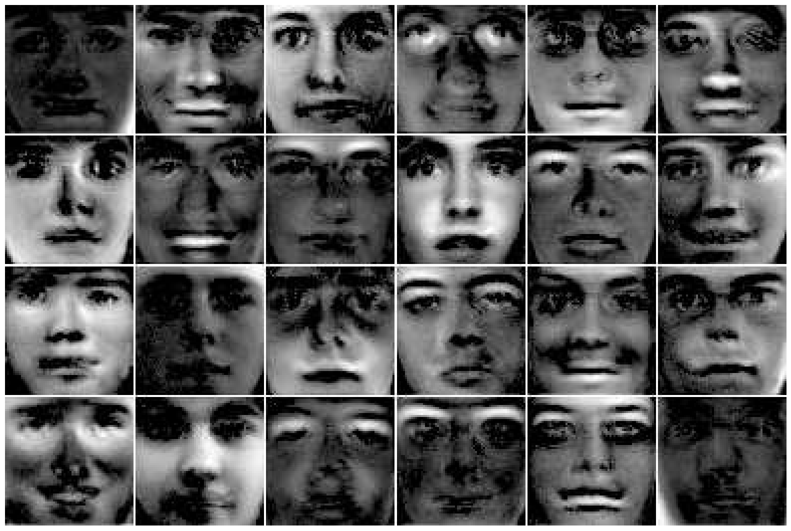
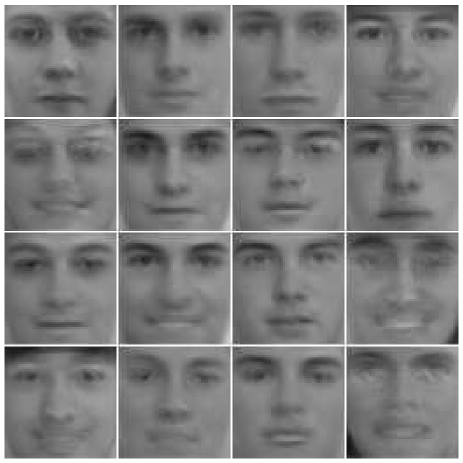
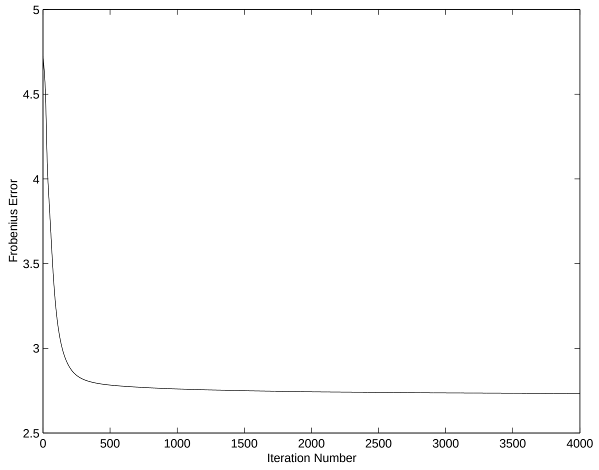


Figure 4.3: Divergence algorithm: basis faces ($r = 24$).

4.1.2 Divergence Algorithm



(a) Approximation



(b) Frobenius Error

Figure 4.4: NMF with the Divergence algorithm.

Figure 4.3 shows the 24 faces contained in the basis **W** that results from an implementation of the Divergence algorithm. These faces are similar in structure to those in Figure 4.1 and generally look like “average” faces with a slight emphasis on certain components (such as lips or eye sockets). It is also interesting to note which basis faces appear in the same position as the similar one in Figure 4.1. For these faces, we see that the random initialization used for both algorithms produced fairly similar basis columns (in the same location in **W**).[4]

The resulting low rank approximations for the first 16 faces in **F** are shown in Figure 4.4 (a) (Again, compare with Figure 3.1). In terms of the error that results from this approximation, the Divergence Algorithm leads to error that is very close to that of the “best” factorization in the Frobenius norm. Compare the scale of Figure 4.4 (b) with that of Figure 4.2 (b) and notice that the approximation error that results from

[4] One may see several similar faces in the two bases shown in different locations. However, the location of a particular face (column) in the basis carries no significance (unlike the basis for the ordered SVD or QR factorizations) and could be changed by permuting the r columns of **W** and r rows of **H** appropriately.

the Divergence algorithm is comparable with that of the Euclidean distance algorithm. Empirically, we have found that the convergence, measured by when the asymptotic error has been attained, of the Divergence algorithm is the best of the three algorithms presented here.

4.1.3 Local NMF Algorithm

Figure 4.5 shows the 24 faces contained in the basis $\mathbf{W}$ that result from an implementation of the Local NMF (LNMF) algorithm. These faces differ significantly from those in the two previous bases.[5] The images shown here are much more representative of parts of faces. Some images represent eyes, others noses or eyebrows and still others the left and right backgrounds of the images. This basis is characteristic of the kind of "component-based" basis that NMF theoretically creates, allowing us to visualize how a particular face could be constructed from different weights of noses, eyes, chins, et cetera. Before further examining some of the properties of this rather unusual basis, we explore two of the weaknesses of the LNMF algorithm: its resulting error, and slow convergence.

Figure 4.6 (b) shows the Frobenius error, resulting from the LNMF algorithm, obtained at each iteration. First we note that this error (even after 4000 iterations) is more than 3 times that of the other two algorithms. Recall that, by design, we constructed the columns of $\mathbf{X}$ to be of unit length in the two norm and therefore $\|\mathbf{X}\|_F = \sqrt{n} \approx 19.55$. Therefore, after 4000 iterations we have obtained a rank 24 approximation whose relative error is nearly 51% compared to the very similar updates of the Divergence algorithm which acquired its asymptotic error much sooner and created a rank 24 approximation with a relative error of only 17%. We also remark that while a significant amount of the approximation error decays in the first few iterations, the subsequent behavior of the error is unlike the previous two algorithms in that the error

[5] Recall that the Local NMF objective function was crafted to obtain its special structure.

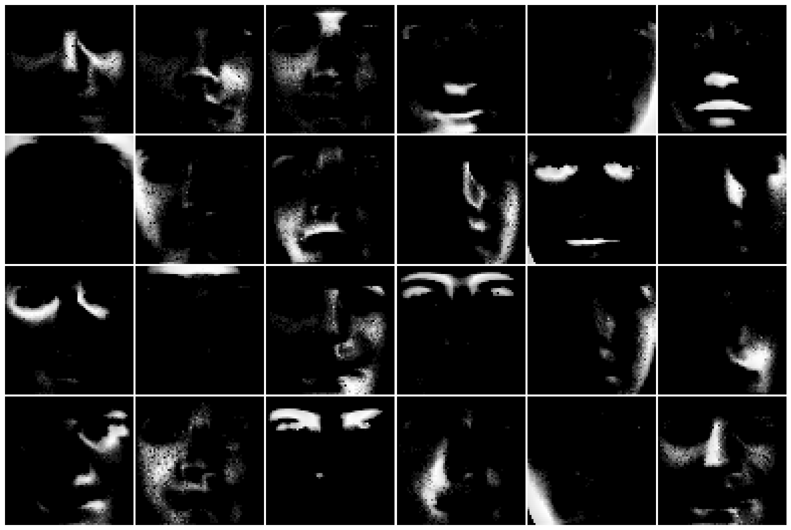
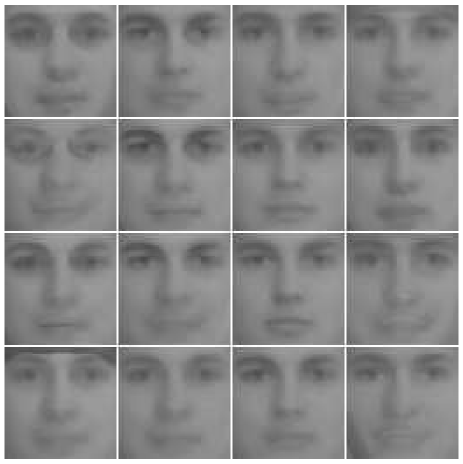
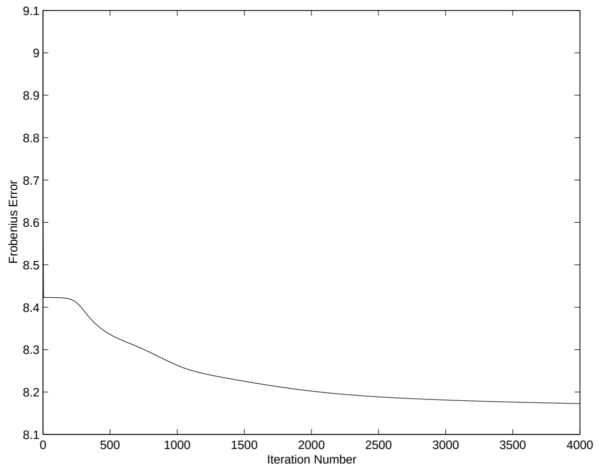


Figure 4.5: Local NMF algorithm: basis faces ($r = 24$).


(a) Approximation



(b) Frobenius Error

Figure 4.6: NMF with the Local NMF algorithm.

appears to stabilize in two different places. Further, the error appears to be continuing to decrease, more than its Euclidean distance and Divergence counterparts, even after 4000 iterations.

We now wish to more closely investigate the structure of the special basis in Figure 4.5. Recall that the difference of the $\mathbf{W}$ and $\mathbf{H}$ updates of the Divergence and LNMF algorithms differed only by a square root. However, this subtle difference was designed to enforce some special properties of the basis in $\mathbf{W}$.

Algorithm[6]	**Orthogonality**	**Sparsity**	**Relative Error**[7]
Euclidean Distance	113.3529	22.28%	13.82%
Divergence	115.8721	22.88%	13.98%
Local NMF	7.2137	85.48%	41.82%

Table 4.1: Comparing the basis $\mathbf{W}$ for three NMF algorithms.

First, LNMF attempts to make the basis vectors $\{w_1, w_2, \ldots, w_r\}$ as "orthogonal as possible." To test the orthogonality we begin by normalizing each basis vector so that $\|w_j\|^2 = w_j^T w_j = 1$ for $j = 1, \ldots, r$. We then define our orthogonality measure as the sum of the $\frac{r(r-1)}{2}$ inner products:

$$\phi(\mathbf{W}) \equiv \sum_{i \neq j} w_i^T w_j = \sum_{i \neq j} \cos(\Theta_{ij}). \tag{4.1}$$

Equation (4.1) includes the cosine interpretation of this measure where $\cos(\Theta_{ij})$ corresponds to the cosine of the angle (in m dimensional space) between w_i and w_j. $\phi(\mathbf{W})$ is a bounded measure: if $\mathbf{W}$ is an orthogonal basis, then $\phi(\mathbf{W}) = 0$, whereas if the columns of $\mathbf{W}$ are identical, then $\phi(\mathbf{W}) = \frac{r(r-1)}{2}$ (276 in the case of $r = 24$). The second column of Table 4.1 shows the value of this measure for all three of the bases plotted earlier. Clearly, the basis for the LNMF is significantly more orthogonal than its two counterparts.

[6] The bases here are those shown in Figures 4.1, 4.3, and 4.5.

[7] The relative error shown here is the ratio $\frac{\|\mathbf{X}-\mathbf{WH}\|_F}{\|\mathbf{X}\|_F}$ after 4000 iterations, where $\|\mathbf{X}\|_F = \sqrt{n} \approx 19.55$.

Next, we wish to obtain localized basis components by enforcing the sparsity of **W**. We need to be careful in defining sparsity here because we must consider how small an element of **W** must be to numerically be the equivalent of a zero. In the case of grayscale images, our palette is restricted to consist of 256 colors. For quantification purposes we therefore use the following definition for sparsity:

$$\text{If} \quad w_{ij} < \frac{\|w_j\|_\infty}{256}, \quad \text{then} \quad \hat{w}_{ij} \equiv 0. \tag{4.2}$$

Conceptually, this definition treats basis elements with the same thresholding procedure as a computer image viewer would [37]. We remind the reader that this thresholding does not actually take place because we are more interested in the additive combination of basis vectors, and, when summing over the entire basis, these small elements become nonnegligible. We only do the thresholding procedure here to aid in the visualization of **W**. The third column of Table 4.1 shows the percentage of the $m * r = 98304$ elements in **W** that are zero with respect to the above definition. The LNMF basis is several times more sparse than the two alternatives and this property can be verified by the abundance of black regions in Figure 4.5.

In summary, the three NMF algorithms that monotonically decrease the objective functions Θ_{NMF_E}, Θ_{NMF_D}, and Θ_{NMF_L} each have their own benefits:

- The Euclidean distance algorithm minimizes the Frobenius error of the resulting low rank approximation.
- The Divergence algorithm maintains low Frobenius error and empirically converges faster.
- The Local NMF algorithm converges slowly with a relatively poor error but produces a basis with a very special structure.

4.2 Spherical K-Means Examples Using Faces Data Set

Suppose now that instead of obtaining the best 24 NMF basis faces (faces that can be added up to recreate the Faces data set) we wish to obtain the best 24 centroid faces. These faces are what would result from using Spherical K-Means to partition the 382 original faces into $k = 24$ disjoint clusters, each with an associated centroid face. Due to the definition of a centroid face, each of these 24 faces would necessarily be combinations (averages) of one or more of the original faces. Figure 4.7 shows the 24 resulting centroid faces for one particular (random initialization) implementation of Spherical K-Means.

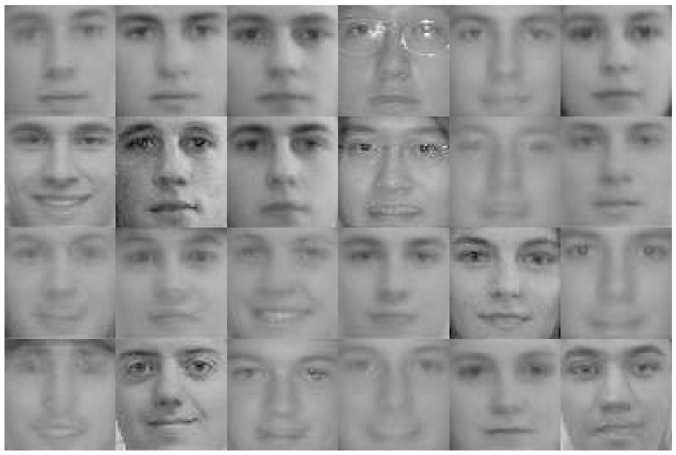
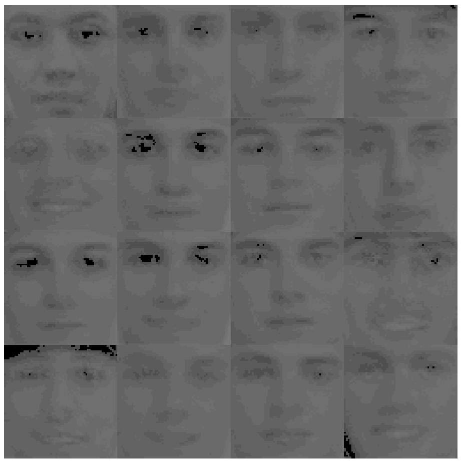
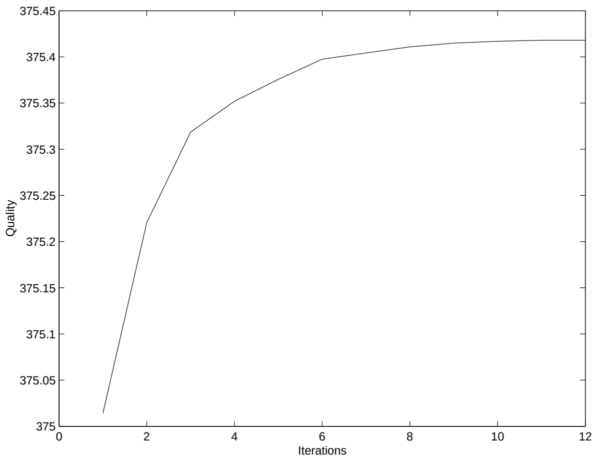


Figure 4.7: Spherical K-Means algorithm: centroid faces ($k = 24$).

In Figure 4.7, the faces which appear to be the most blurry are those whose associated centroids are the average of the greatest number of original faces. Those that are much more distinct and defined are typically those whose associated centroids are the average of the fewest number of original faces. For example, the well defined face near the center of Figure 4.7 was probably isolated (left in a cluster by itself) because the subject was wearing glasses and had their hair showing in the picture's frame. This should be contrasted with the face just to the right where one can just barely make out a glasses frame. For this frame, one or more subjects wearing glasses sufficiently resembled subjects without glasses so that they were combined in the same cluster. The partitioning of faces based on facial characteristics can also be seen in several of the other faces. For example, the face in the lower left seems to have been obtained by combining faces with eyes in several different locations. However, a characteristic of each of these faces was that the subject was wearing a hat or had the front part of their hair exposed. Other faces seem to have been based on other characteristics such as eyebrow location, mouth expression, or nose size/shape.



(a) Approximation



(b) Quality

Figure 4.8: Spherical K-Means on the Faces data set.

Figure 4.8 (a) shows the approximation of the 16 original faces used throughout this chapter using the centroid faces in Figure 4.7 and weights in the coefficient matrix **Z** as determined by the concept decompositions of Chapter 3. The reconstructed faces in this figure are visually worse than those obtained by the Euclidean distance and Divergence NMF algorithms. However, the number of $\mathcal{O}(mnk)$ iterations required to obtain these centroid faces is much less than the iterations used by the NMF algorithms. Figure 4.8 (b) shows the nondecreasing behavior of the quality (as measured by Θ_{SKM})

Algorithm[9]	**Orthogonality**	**Sparsity**	**Relative Error**[10]
Spherical K-Means	266.7736	.0092%	15.11%
Euclidean Distance	113.3529	22.28%	13.82%
Divergence	115.8721	22.88%	13.98%
Local NMF	7.2137	85.48%	41.82%

Table 4.2: Comparing the Spherical K-Means centroid basis **C** to the basis **W** from the NMF algorithms.

of the clustering obtained.[8]

Lastly, we examine the structural features of the basis of centroid faces using the same measures as were used for the NMF bases. Table 4.2 shows the orthogonality, sparsity and relative error for the Spherical K-Means basis **C** in relation to the NMF bases **W**. While the basis **C** is still full rank ($rank(\mathbf{C}) = 24$), its vectors are far from being orthogonal to each other. One might expect this because, despite the disjointness of the vectors, all of the centroids live in a very confined space of 64×64 pixel faces all cropped roughly the same. Clustering has also removed any sparsity inherent in the data – only 9 of the $m * k = 98304$ elements in **C** would be thresholded to zero. This is also to be expected because adding together different faces will cause a particular face to lose any sparsity it may originally have had (recall that **F** was already not sparse to begin with).

The benefit of the Spherical K-Means basis would then be that we obtain roughly the same quality approximation (as measured by the Frobenius norm) but do so in fewer operations than are required by NMF for the same data. The nature of the clustering ensures that at each iteration we already have a basis that resembles the original data (faces in this chapter). This is in contrast to NMF, where random initializations are traditionally used and the algorithms must work very hard (as measured by the number

[8] See Chapter 2 for a definition of the quality shown here.

[9] The bases here are those shown in Figures 4.1, 4.3, 4.5 and 4.7.

[10] The relative error shown here is the ratio $\frac{\|\mathbf{X}-\mathbf{WH}\|_F}{\|\mathbf{X}\|_F}$ after 4000 iterations, where $\|\mathbf{X}\|_F = \sqrt{n} \approx 19.55$. For Spherical K-Means, this error was obtained using the concept decomposition for the basis **C** defined in Chapter 3.

of iterations) on their basis to get it to travel back into the space of the original data. In the next chapter we intend to propose a technique to speed up this process.

Chapter 5

NMF Seeding Strategies Using Spherical K-Means

As we have seen in the previous chapter, the majority of the error reduction in NMF is done within approximately the first 100 iterations. Subsequent iterations produce smaller changes of elements in the basis **W** which still monotonically decrease the given objective function, but this decrease is virtually nothing compared to the earlier progress. Much of this document thus far has been an exploration of the NMF and Spherical K-Means techniques separately. In this chapter we propose a smarter initialization of NMF using the Spherical K-Means clustering. We will show that this initialization-factorization process produces an asymptotic error comparable to that of standard NMF with random initialization and does so in far fewer iterations.

We first prove a relationship between the Euclidean distance objective function and the Spherical K-Means objective function for a specific elementary NMF. Next, we explore what occurs geometrically when NMF is seeded with Spherical K-Means. We also provide numerous results showing the resulting speed-up. We conclude this chapter by emphasizing one of the primary applications of this seeding technique – determining the (approximation rank) parameter, r, for NMF and a given data set.

5.1 Why Spherical K-Means?

In this section we hope to establish that the centroids obtained by Spherical K-Means clustering provide a good initialization for the left factor **W**.

5.1.1 Initialization Considerations

Clearly, there are numerous alternative ways to initialize the left factor **W** or both factors **W** and **H** required by NMF. The only requirement on these factors (besides their size) is that they consist of strictly non-negative elements. An additional consideration may need to be taken into account based on the NMF objective function/ update scheme employed. For the multiplicative updates introduced in Chapter 2 where the previous iteration's factor is multiplied by a quality measure, the update cannot overcome the initialization of a zero. For example if we have an element in the left factor **W**, $w_{i,j}^{(t)} = 0$ at iteration t, then it will remain a zero for all subsequent iterations:

$$w_{i,j}^{(t+1)} \equiv w_{i,j}^{(t)} \, Q(\cdot)_{i,j} = 0. \tag{5.1}$$

While there have been no other NMF initialization studies to date, this result shows that the entirely nonzero random initializations are a starting point from which the largest number of possible Non-negative Matrix Factorizations may be obtained. Further, this result is somewhat counterintuitive in that one possible goal of NMF may be to obtain a sparse basis **W**. In fact, this is one of the first benefits of seeding with the Spherical K-Means clustering.

The centroids that result from Spherical K-Means reflect the sparsity of the original data. The centroids obtained are usually less sparse than a typical data vector because they are obtained by averaging several data vectors. However, because similar data vectors are clustered together, the centroid will exhibit any sparsity common to all of the vectors in a particular centroid. (The intersection of zero elements, if you will.) Therefore, the sparsity of the centroids reflect certain trends in the original data set (trends that NMF wishes to capture) and allows us to see a slight reduction in computational expenses. Again, once an element is zero, it remains zero and the computation in Equation (5.1) is unnecessary.

The location of zeros in the initialization described above could also be obtained using centroids obtained from other clustering methods. There are two primary reasons why we have chosen a Spherical K-Means clustering instead of its more general predecessor, K-Means. First, we wish to take advantage of the Spherical K-Means algorithm, with refinements, because of its efficiency and robustness for very large data sets. Second, because of the normalization inherent to NMF update schemes, we need the centroids to be as different as possible when normalized. Recall that one motivation for Spherical K-Means clustering was that it deals only with the direction of data vectors – each original data vector contributes equally to obtain the k characteristic centroids. As a result, the Spherical K-Means clustering yields centroids that will be more linearly independent than those from K-Means.[1] The normalization theorem shown in Chapter 3 demonstrates that NMF may be initialized with vectors of unit length in any norm. In the case of the Spherical K-Means centroids, NMF would begin with a factor $\mathbf{W}$ normalized in the Euclidean norm. Lastly, clustering techniques, such as PDDP [5], which are based on the SVD also provide inadequate initializations because they do not respect the non-negativity of both factors in NMF.

Another reason for using Spherical K-Means to seed NMF is that we have seen that the NMF algorithms included here initially do the work of obtaining basis vectors that resemble the original data before beginning the long process of finding more parts-based representations. In the specific example of the Faces data set, all three algorithms took the random initialization for the r columns of $\mathbf{W}$ and brought them back into the "face space" of the original matrix $\mathbf{F}$. Initializing with the Spherical K-Means k centroids ensures that we are already in this space. Further, as shown in Chapter 3, the computational requirements per iteration of Spherical K-Means is $\mathcal{O}(mnk)$ but with a smaller constant than the NMF algorithms. In searching for an initialization, our primary goal is to reduce the total number of operations actually performed and so it is our desire to stay in the space of techniques which do not increase the overall

[1] An example of this shortcoming of K-Means was shown in Figure 2.9.

computational complexity of the initialization-factorization process.

We now come to an important point, where we must combine the different notations used by NMF and Spherical K-Means. For the seeding process, the parameters k (number of clusters) and r (rank of approximation) are equivalent. It was for this reason that we have avoided the Spherical K-Means initializations that result in less than k centroids (which would mean that one cluster is left empty). From this point forward, when we say that we are initializing NMF using the Spherical K-Means centroids we mean that $\mathbf{W}^{(0)} \equiv \mathbf{C}$ where $\mathbf{C}$ is the centroid matrix

$$\mathbf{C} \equiv \begin{bmatrix} | & | & & | \\ c_1 & c_2 & & c_k \\ | & | & & | \end{bmatrix} \tag{5.2}$$

used by the concept decompositions of Chapter 3. The ordering of the centroids c_i is unimportant here, just as the column ordering of the final NMF factor $\mathbf{W}$ is unimportant.[2]

Before we move on, we acknowledge that there may be "better" ways to initialize NMF (either for error minimization or preservation of structure). However, this area has remained unexplored[3] and it is, at least partially, the intent of this document to show the benefits of seeding NMF. In doing so, we hope to motivate others to further develop alternative initialization strategies.

[2] Permuting the columns of the left factor $\mathbf{W}$ of a Non-negative Matrix Factorization is fine as long as the rows of the right factor $\mathbf{H}$ are similarly permuted.

[3] Very recent work has been done by Chu and Funderlic [8] on the relationships between clustering techniques and the SVD to motivate a centroid decomposition. However, their approach applies only to the unconstrained problem where the SVD reigns.

5.1.2 The Elementary NMF Using Centroids

We now define the "elementary" Non-negative Matrix Factorization, $\mathbf{E} \equiv \mathbf{W}\Delta$, where

$$\Delta_{i,j} \equiv \begin{cases} 1 & \text{if} \quad i = \arg\min_i \|x_j - w_i\|_2^2 \\ 0 & \text{else} \end{cases} \qquad \text{for} \quad j = 1, \ldots, n, \tag{5.3}$$

and, x_j and w_i are the column vectors of $\mathbf{X}$ and $\mathbf{W}$, respectively.

Clearly $\mathbf{E}$ is an NMF approximation that depends solely on the left NMF factor $\mathbf{W}$ and where the right factor $\mathbf{H}$ is replaced by a composition of elementary (indicator) vectors. Recalling that the NMF updates presented earlier first correct the right factor $\mathbf{H}$, we assert that it should be a goal of NMF initialization to begin with a basis in $\mathbf{W}$ that closely approximates the initial data matrix $\mathbf{X}$. To emphasize one way of doing this, we introduce the following theorem.

Theorem 2 For a data matrix $\mathbf{X}$ with n columns of unit length, maximizing the Spherical K-Means objective function Θ_{SKM} is equivalent to minimizing the Euclidean Distance NMF objective function Θ_{NMF_E} for the elementary Non-negative Matrix Factorization, $\mathbf{E} = \mathbf{C}\Delta$.

Proof: Recalling the notation used for the Spherical K-Means concept decompositions, we have:

$$\mathbf{C} \equiv \begin{bmatrix} | & | & & | \\ c_1 & c_2 & & c_k \\ | & | & & | \end{bmatrix}$$

where $\{c_i\}_{i=1}^{k}$ are the k centroids of unit length so that

$$c_{\iota_i} = \frac{\sum_{x_j \in \pi_{\iota_i}} x_j}{\|\sum_{x_j \in \pi_{\iota_i}} x_j\|} \qquad \text{with} \qquad x_i \in \pi_{\iota_i}.$$

We now use the Law of Cosines for vectors:

$$\|c_{\iota_i} - x_i\|^2 = \|c_{\iota_i}\|^2 + \|x_i\|^2 - 2\|c_{\iota_i}\|\|x_i\| \cos(\Theta_{x_i, c_{\iota_i}}).$$

Using the unit length of both the data vectors and centroids and solving for the cosine of the angle between x_i and c_{ι_i} we get

$$\cos(\Theta_{x_i,c_{\iota_i}}) = 1 - \frac{1}{2}\|c_{\iota_i} - x_i\|^2.$$

Finally we sum over all n data vectors to obtain

$$\sum_{i=1}^{n} \cos(\Theta_{x_i,c_{\iota_i}}) = n - \frac{1}{2}\sum_{i=1}^{n} \|c_{\iota_i} - x_i\|^2,$$

or,

$$\Theta_{SKM} = n - \frac{1}{2}\Theta_{NMF_E}{}^{E},$$

where $\Theta_{NMF_E}{}^{E}$ denotes the Euclidean Distance NMF objective function for the specific elementary factorization $\mathbf{E} = \mathbf{C}\Delta$. Thus we see that increasing Θ_{SKM} is equivalent to decreasing $\Theta_{NMF_E}{}^{E}$. $\square$

We remark that the above theorem can be extended to include a data matrix $\mathbf{X}$ whose columns are not necessarily of unit Euclidean length. The result would be a very similar maximization-minimization property but would need to address the varying weights $\|x_i\|$. Although this theorem follows directly from the Law of Cosines, we include it here as it offers the first bridge between the Spherical K-Means clustering problem and the NMF problem. We would now like to apply the results of this theorem to obtain a better factorization by relaxing the restriction on the left factor $\mathbf{H}$.

5.2 Initializing the Coefficient Matrix H

We already have some justification for using the Spherical K-Means centroids in $\mathbf{C}$ to seed the NMF basis $\mathbf{W}$. Since $\mathbf{C}$ is already computed from the Spherical K-Means process, we wish to take advantage of the computations already done to obtain a smart initialization for the accompanying left factor $\mathbf{H}$. One such strategy was introduced in the previous section, where the indicator vector ι was used to create the matrix Δ.

However, the initialization $\mathbf{W}^{(0)} = \mathbf{C}$, $\mathbf{H}^{(0)} = \Delta$ will result in a fixed point for the Euclidean Distance NMF algorithm as shown in the following Lemma.

Lemma 1 Given a dataset $\mathbf{X}$ with a Spherical K-Means clustering defined by the centroid matrix $\mathbf{C}$ and indicator matrix Δ, the initialization $\mathbf{W}^{(0)} = \mathbf{C}$, $\mathbf{H}^{(0)} = \Delta$ will result in a fixed point for the Euclidean Distance NMF algorithm.

Proof: Recall that the Euclidean distance NMF algorithm seeks to minimize the objective function:

$$\Theta_{NMF_E}(\mathbf{W}, \mathbf{H}) \equiv \sum_{j=1}^{n} \|x_j - \mathbf{W}h_j\|_2^2 = \|\mathbf{X} - \mathbf{W}\mathbf{H}\|_F^2. \tag{5.4}$$

The first step in the Euclidean distance NMF algorithm is to update the right factor $\mathbf{H}$ using the initializations $\mathbf{W}^{(0)} = C$, $\mathbf{H}^{(0)} = \Delta$. Notice that the multiplicative update of $\mathbf{H}$ will not change the location of the zeros in $\mathbf{H}^{(t)}$ for $t > 0$ given the zeros in the initialization $\mathbf{H}^{(0)}$:

$$H_{aj}^{(t+1)} = H_{aj}^{(t)} Q_E\left(\mathbf{W}^{(t)}, \mathbf{X}^T, \mathbf{H}^{(t)}\right)_{aj}. \tag{5.5}$$

After one update, each of the n columns of $\mathbf{H}$ will be multiplied by some constant, say q_i. We then turn to the update for the left factor $\mathbf{W}$. Recall that our initialization $\mathbf{W}^{(0)} = \mathbf{C}$ has the following the property by the Cauchy-Schwarz Inequality:

$$\sum_{x_i \in \pi_j} x_i^T w \leq \sum_{x_i \in \pi_j} x_i^T c_j, \qquad \forall w \in \Re^m, \quad \forall \pi_j$$

$$\Longleftrightarrow$$

$$\sum_{i=1}^{n} x_i^T w_{\iota_i} \leq \sum_{i=1}^{n} x_i^T c_{\iota_i}, \qquad \forall w_{\iota_i} \in \Re^m. \tag{5.6}$$

Here let $\{w_{\iota_i}\}_{\iota_i=1}^{k}$ denote the possible k-sets of basis vectors after future updates $\mathbf{W}^{(t)}$. Using the definition of the inner product and the Law of Cosines as in Theorem 2, we have:

$$\sum_{i=1}^{n} \frac{1}{2} \left(\|w_{\iota_i}\|^2 + \|x_i\|^2 - \|w_{\iota_i} - x_i\|^2 \right) \leq \sum_{i=1}^{n} \frac{1}{2} \left(\|c_{\iota_i}\|^2 + \|x_i\|^2 - \|c_{\iota_i} - x_i\|^2 \right), \tag{5.7}$$

or,

$$\sum_{i=1}^{n} \left(\|c_{\iota_i}\|^2 - \|w_{\iota_i}\|^2 \right) + \sum_{i=1}^{n} \left(\|w_{\iota_i} - x_i\|^2 - \|c_{\iota_i} - x_i\|^2 \right) \geq 0, \tag{5.8}$$

for any $w_{\iota_i} \in \mathbf{W}^{(t)}$.

Theorem 1 in Chapter 3 showed that the normalization of the basis vectors in $\mathbf{W}$ does not change the error of the approximation if the rows of $\mathbf{H}$ are also appropriately changed. Therefore we are justified in enforcing $\|w_j\| = \|c_j\| = 1$ without changing the above statements. This normalization gives

$$\sum_{i=1}^{n} \left(\|w_{\iota_i} - x_i\|^2 - \|c_{\iota_i} - x_i\|^2 \right) \geq 0. \tag{5.9}$$

Here we notice that $c_{\iota_i} = \mathbf{C}\Delta_i$ is the initial factorization and that $w_{\iota_i} = \frac{1}{q_i}\mathbf{W}^{(t)} H_i^{(t)}$ is the form of the subsequent updates of $\mathbf{W}$ given the restrictions on all subsequent $\mathbf{H}$ noted above. Equation (5.9) then means that:

$$\Theta_{NMF_E}\left(\mathbf{W}^{(t)}, \mathbf{H}^{(t)}\right) \geq \Theta_{NMF_E}(\mathbf{C}, \Delta). \tag{5.10}$$

However, from the monotonicity of the Euclidean distance NMF algorithm, we are guaranteed that:

$$\Theta_{NMF_E}\left(\mathbf{W}^{(t)}, \mathbf{H}^{(t)}\right) \leq \Theta_{NMF_E}\left(\mathbf{W}^{(0)}, \mathbf{H}^{(0)}\right) \qquad \text{for} \quad t \geq 0. \tag{5.11}$$

It then follows that the initialization $\mathbf{W}^{(0)} = \mathbf{C}$, $\mathbf{H}^{(0)} = \Delta$ will result in a fixed point:

$$\Theta_{NMF_E}\left(\mathbf{W}^{(t)}, \mathbf{H}^{(t)}\right) = \Theta_{NMF_E}(\mathbf{C}, \Delta) \qquad \text{for} \quad t \geq 0. \tag{5.12}$$

□

As a result of the above Lemma, we do not use the initialization $\mathbf{W}^{(0)} = \mathbf{C}$, $\mathbf{H}^{(0)} = \Delta$ because we would like NMF to improve an initial approximation. Another strategy is to use the coefficient matrix $\mathbf{Z}$ from the concept decompositions of Chapter 3. However, this coefficient matrix does not respect the non-negativity constraint placed on $\mathbf{H}$. We have explored rudimentary variations of $\mathbf{Z}$ such as taking its absolute value or replacing negative values with zero. These variations are somewhat justifiable given the properties of $\mathbf{Z}$. We have noticed that the number and magnitude of the negative values typically found in $\mathbf{Z}$ is always smaller than the positive values.[4] This is somewhat intuitive because we expect that in the i-th column of $\mathbf{Z}$, the largest value is usually found in row ι_i associated with the i-th point's centroid c_{ι_i}. For example, the concept decomposition of the Faces data set with 30 clusters resulted in a $\mathbf{Z}$ whose mean value over its $30 * 382 = 11460$ element values was $0.0328 > 0$. The histogram of these values is shown in Figure 5.1. Despite the majority of $\mathbf{Z}$ elements being non-negative, we have found superior ways to minimize the error of the initialization within NMF's non-negativity constraints.

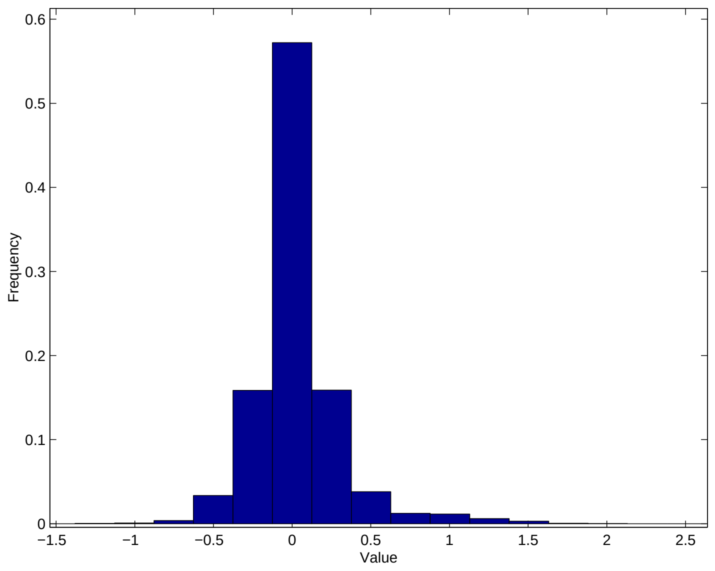


Figure 5.1: Histogram of the 11460 elements in $\mathbf{Z}$.

[4] This is particularly true when using a low number of clusters (<20) as will be illustrated in future sections.

Recall that the concept decomposition coefficient matrix $\mathbf{Z}$ is the least squares solution to $\|\mathbf{CZ} - \mathbf{X}\|_F^2$. If the columns of $\mathbf{Z}$ are not linearly independent, Rosen's method of finding minimum and basic solutions [42] may be applied. A variation of Rosen's method which deals with the specific non-negativity constraints which we desire is Non-Negative Least Squares (NNLS).

NNLS gives the least squares solution to $\|\mathbf{CY} - \mathbf{X}\|_F^2$ subject to the additional constraint $Y_{i,j} \geq 0$. Constrained least squares problems of this form must satisfy the Kuhn-Tucker Conditions which lead to the NNLS implementation suggested by Lawson and Hanson. It is not the purpose of this document to further develop NNLS but we refer the reader to Lawson and Hanson's **Solving Least Squares Problems** [32] for the NNLS algorithm used here. Here we are only concerned that a least squares solution Y that satisfies the NMF non-negativity constraints is obtainable.[5]

Before we proceed we prove the following rather intuitive lemma.

Lemma 2 Given a data matrix $\mathbf{X}$, and its associated Spherical K-Means centroid matrix $\mathbf{C}$ with elementary NMF right factor (indicator) Δ, we have:

$$\|\mathbf{X} - \mathbf{C}\Delta\|_F^2 \geq \|\mathbf{X} - \mathbf{C}\mathbf{H}_N\|_F^2 \geq \|\mathbf{X} - \mathbf{W}^{(1)}\mathbf{H}^{(1)}\|_F^2 \geq \|\mathbf{X} - \mathbf{W}^{(\infty)}\mathbf{H}^{(\infty)}\|_F^2,$$

where $\mathbf{H}_N$ is the Non-Negative Least Squares coefficient matrix and $\mathbf{W}^{(t)}$, $\mathbf{H}^{(t)}$ are the NMF factors that result from t iterations of the Euclidean distance NMF algorithm with initializations $\mathbf{W}^{(0)} = C$ and $\mathbf{H}^{(0)} = \mathbf{H}_N$, respectively.

Proof: The NNLS solution $\mathbf{H}_N$ satisfies $\|\mathbf{C}\mathbf{H}_N - \mathbf{X}\|_F^2 \leq \|\mathbf{CY} - \mathbf{X}\|_F^2$ for all non-negative $\mathbf{Y}$. In the particular case of $\mathbf{Y} = \Delta$ we have the first inequality. The second and third inequalities come from the monotonic decrease of the objective function Θ_{NMF_E} by the Euclidean distance NMF algorithm. These inequalities only hold because of the feasibility of the initializations $\mathbf{W}^{(0)} = \mathbf{C}$ and $\mathbf{H}^{(0)} = \mathbf{H}_N$. Here, $\mathbf{W}^{(\infty)}$, $\mathbf{H}^{(\infty)}$ denote

[5] Lawson and Hanson prove that their NNLS algorithm will converge in a finite number of iterations and state that convergence is usually obtained after only $\frac{1}{2}k$ steps, where k is the length of the solution vectors y_i.

the best (long-term) factors obtainable by the Euclidean distance objective function with the given initialization. □

Conceptually, the first inequality in Lemma 2 verifies that using the centroids as a left factor and other nonnegative right factors than the indicator matrix Δ will result in factorizations that are at least as good as the elementary NMF. This is illustrated in Figure 5.2 where the middle data point can be better approximated using a positive combination of c_1 and c_2 than it could be just using its associated centroid, c_1.

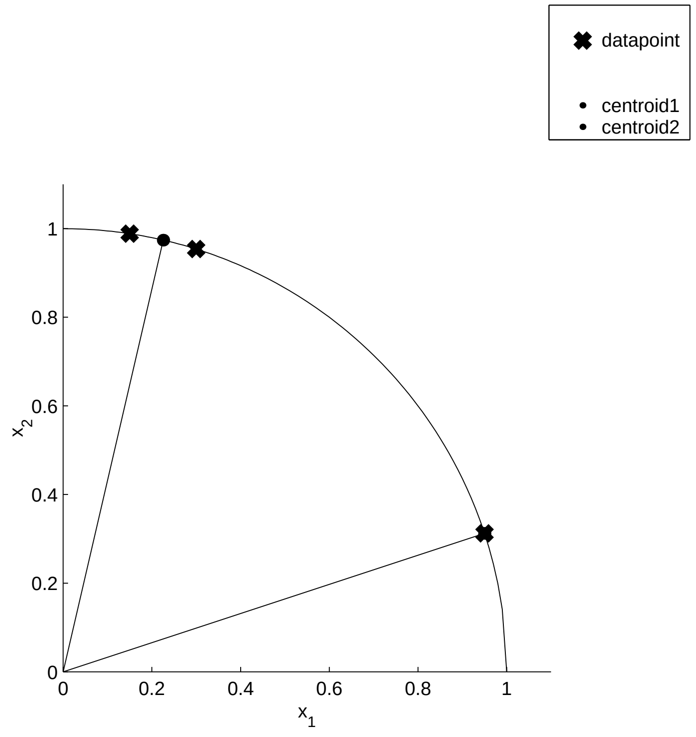


Figure 5.2: Graphical representation of Lemma 2.

The subsequent inequalities of Lemma 2 emphasize that the pairing of $\mathbf{C}$ and its associated NNLS solution factor is not necessarily the best NMF in the Euclidean distance sense. Iterations of the Euclidean distance NMF algorithm will change both factors (so that the left factor is no longer fixed to be $\mathbf{C}$ as in our initialization strategies) in search of a better factorization. In the next section we explore some computational results using the various initializations introduced in this section.

5.3 Initialization Results

5.3.1 Better Clustering, Better Initialization

The first step in empirically validating the initialization of $\mathbf{W}$ using $\mathbf{C}$ is showing that improving the clustering that defines the centroids in $\mathbf{C}$ will improve the quality of the initialization. We illustrate this property in Figure 5.3.

We begin by clustering the Classic3 data set $\mathbf{T}$ into 12 disjoint clusters using Spherical K-Means. The Spherical K-Means algorithm required approximately 20 iterations to converge to an optimal clustering. Every two iterations, we saved the current set of centroids and then used these centroids to seed the $\mathbf{W}^{(0)}$ with the same random initialization of $\mathbf{H}^{(0)}$.[6]

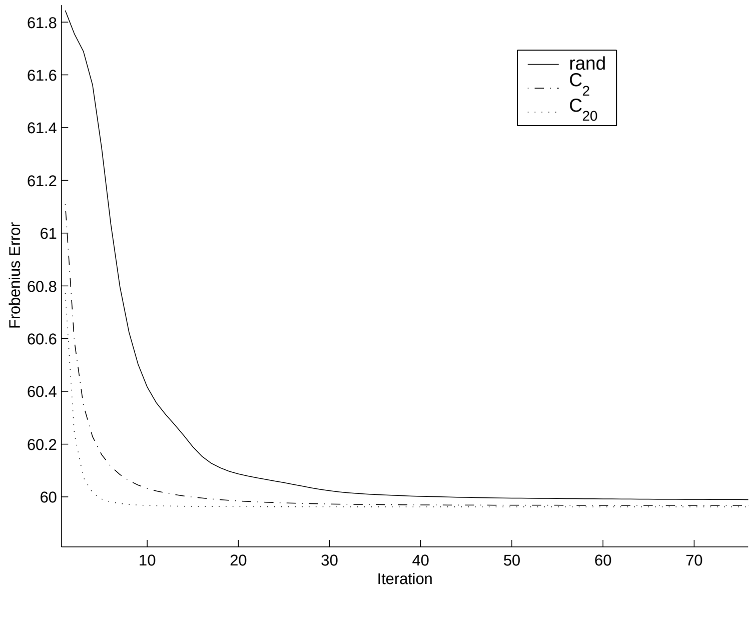


Figure 5.3: Improving the quality of the Spherical K-Means clustering to obtain a better initialization for $\mathbf{W}$.

Figure 5.3 shows the Frobenius error that resulted at each iteration of the Eu-

[6] Random initialization of $\mathbf{H}^{(0)}$ is used here as a starting point so that we only need consider the changes as a result of the left factor initialization.

clidean distance NMF algorithm. First, we show the error using the standard random initialization. Next, we show the error made using $\mathbf{W}^{(0)} = \mathbf{C}_2$, where $\mathbf{C}_2$ is the centroid matrix that resulted after two Spherical K-Means iterations. Lastly, we show the error made using $\mathbf{W}^{(0)} = \mathbf{C}_{20}$ for the centroid matrix, $\mathbf{C}_{20}$, of the final clustering. We have purposely neglected to show the intermediate centroid matrices ($\mathbf{C}_4$, $\mathbf{C}_6$, etc.) because these initializations result in errors between those of $\mathbf{C}_2$ and $\mathbf{C}_{20}$.

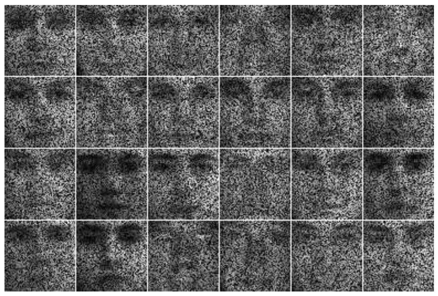
From this figure, we emphasize that the optimal clustering will give a better (at least initially) Non-negative Matrix Factorization than inferior clusterings. Further, the asymptotic value/error of the objective function Θ_{NMF_E} is obtained in approximately 5 iterations. The random initialization required more than 150 iterations to obtain this same error. The other statement that we make in light of these results is that if we only want to allocate a fixed number of iterations to the Spherical K-Means algorithm (i.e. exceeding this amount of operations might make a random initialization more efficient overall), the clustering can be stopped and the current centroid matrix used. For the specific example above, the initializations using $\mathbf{C}_{12}$, $\mathbf{C}_{14}$, $\mathbf{C}_{16}$, and $\mathbf{C}_{18}$ resulted in essentially the same Frobenius error as did the initialization using the optimal $\mathbf{C}_{20}$.

For the remainder of this thesis, we will assume that, except where noted, the centroid matrix $\mathbf{C}$ is obtained using the best (as measured by Θ_{SKM}) clustering.

5.3.2 Progression of the Centroid Initialization in NMF

We now explore the progression of the left factor $\mathbf{W}$ in NMF using different initializations. This progression is well illustrated by using the Faces data set $\mathbf{F}$ so that the reader can better visualize the progression of $\mathbf{W}$ by seeing the changes made in the basis faces obtained. We will use the same standard random initialization $H^{(0)}$ for the left factor in the cases of random initialization and Spherical K-Means initialization.[7]

[7] An optimal clustering was obtained with the Spherical K-Means algorithm.

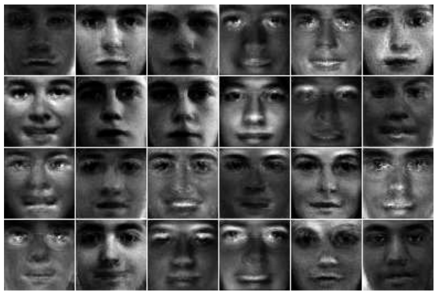

(a) random initialization, 20 it.

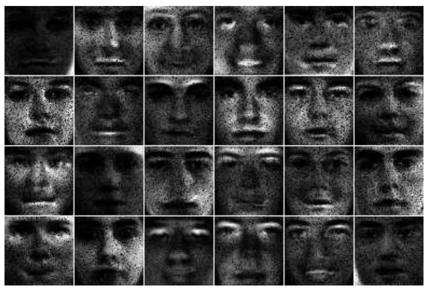
(b) centroid initialization, 20 it.

(c) random initialization, 40 it.

(d) centroid initialization, 40 it.

(e) random initialization, 60 it.


(f) centroid initialization, 60 it.


(g) random initialization, 100 it.

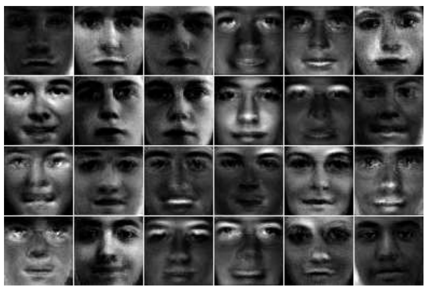

(h) centroid initialization, 100 it.

Figure 5.4: Progression of Faces NMF basis **W** for different initializations.

For consistency, we will let $\mathbf{W}$ consist of 24 basis faces as in the examples of Chapter 4.

Figure 5.4 then shows the progression of these 24 faces after 20, 40, 60, and 100 iterations of the Euclidean distance NMF algorithm. The faces on the left are those obtained using random initialization, while those on the right are from the Spherical K-Means initialization. For visualization, each basis face was normalized to have elements of maximum value 1 ($\|w_i\|_\infty = 1$) so that there exists one pixel on each face which is "pure white."

From Figure 5.4 (a) we see the NMF algorithm's work done to begin to sculpt faces out of the random initialization. Conversely, the Spherical K-Means initialization ensured that $W^{(0)}$ would already resemble faces and Figure 5.4 (b) still has the appearance of the centroids in Figure 4.7. As the Euclidean distance NMF algorithm progresses, the random initialization begins to lose its original noise and more closely resemble the faces (with emphasized components) that we expect. While the Spherical K-Means initialization already began in the "face space", we do not mean to say that the Euclidean distance NMF algorithm is not doing any work on the initialization. The basis faces obtained after 100 iterations are definitely different from those after 20 iterations. However, with the centroid initialization, the algorithm has a head start at beginning to emphasize facial features/components. The interested reader should compare the faces obtained after 100 iterations in each column with the final faces obtained after 4000 iterations using (random initialization of) the Euclidean distance NMF algorithm in Figure 4.1.

These basis faces provide an excellent way of tracking the progression of the left factor $\mathbf{W}$. However, we do not wish to lose sight of our other goal of efficiently minimizing the error of the resulting approximation. Figure 5.5 depicts the Frobenius error of the NMF approximations for the two initializations illustrated in Figure 5.4. This figure shows the head start in error minimization that the Spherical K-Means initialization benefits from and shows that the initialization maintains this advantage

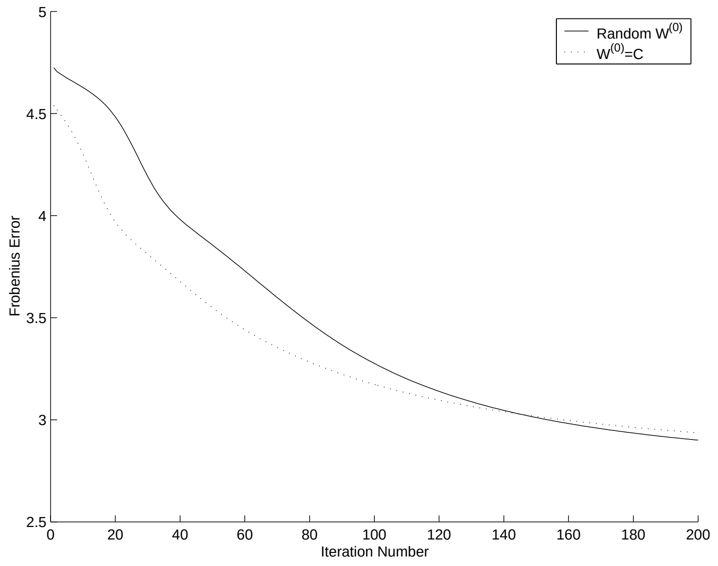


Figure 5.5: Frobenius error given two different initializations for $\mathbf{W}$.

after 100 iterations (Figure 5.4 (g) and (h)).

Interestingly, in the long term, the random initialization will actually achieve a lower error than the centroid initialization. This is because the centroid initialization immediately enforces certain restrictions on the left factor $\mathbf{W}$ and the Euclidean distance NMF algorithm cannot pull the factorization out of a local minimum. The asymptotic error for this problem is approximately 2.75 (as shown in Figure 4.2) and so one could argue that this long term intersection is insubstantial when compared to the reduction in error obtained after only 60 iterations.

Before continuing, we return to the situation described by the clusters shown in Figure 5.2. In that figure, one of the three data points was not able to be represented as a non-negative combination of the two centroids because it lay outside of the cone between the two centroids. However, in this specific example ($m = k = 2$) we know that there do exist pairs of basis vectors which can (through non-negative combinations) reconstruct all three data points.

NMF seeded with these two Spherical K-Means centroids would need to push the first centroid away from the second centroid in order to decrease the Frobenius error of the approximation. This corresponds to making the cone between the two centroids large enough to enclose all three data points. Generalizing to higher dimensions, the expansion of this cone corresponds to an increase of the angles between the extreme points defined by the basis vectors in $\mathbf{W}$.

Table 5.1 follows the orthogonality and sparsity measures of Chapter 4 to quantify this orthogonalizing behavior for the specific example of the faces in Figure 5.4. Shown are the structure of the original centroid basis and the basis that results from 20, 100 and 300 iterations of the Euclidean distance NMF algorithm using the centroid basis as the initialization of $\mathbf{W}$ and the usual random matrix as the initialization of $\mathbf{H}$.[8] Here we see that the angles between the basis vectors increase[9] dramatically as a result of NMF iterations even though we may not see these dramatic changes in Figure 5.4. As certain facial features are emphasized, we also note that the sparsity of the basis increases. Lastly, the relative error obtained is shown and displays the monotonicity guaranteed by the Euclidean distance algorithm.

We hope that the above material sufficiently motivates our application of structured initialization using the Spherical K-Means clustering in Section 5.4. We finish this section by exploring the initialization of the right factor $\mathbf{H}$.

5.4 Suggested Application: Choosing the r for NMF

In the previous sections we have shown how the Spherical K-Means initialization of NMF and specifically the Euclidean distance NMF algorithm provides a significant reduction in approximation error made initially. However, as iterations continue, we have

[8] The interested reader may compare this table to Table 4.1 which shows the same measures for the long-term behaviors of the three different NMF algorithms.

[9] In the table, we measure the cosine of the angles between the basis vectors and so a larger number means that the angles between the vectors are actually smaller.

[10] The relative error shown here is the ratio $\frac{\|\mathbf{X}-\mathbf{WH}\|_F}{\|\mathbf{X}\|_F}$ after, where $\|\mathbf{X}\|_F = \sqrt{n} \approx 19.55$.

[11] Long-term NMF using random initialization, recall Table 4.2.

Basis W	**Orthogonality**	**Sparsity**	**Relative Error**[10]
Before NMF	266.7736	.0092%	-
20 NMF iterations	248.6283	.0539%	20.31%
100 NMF iterations	195.6309	2.957%	16.24%
300 NMF iterations	174.1342	7.611%	14.97%
4000 NMF iterations[11]	113.3529	22.28%	13.82%

Table 5.1: Tracking the change in structure of the basis **W** using NMF with the Spherical K-Means initialization.

also seen that the restrictive nature of this initialization prevents the NMF algorithm from maintaining this advantage in error. In fact, we have seen that in the long-term, we may expect the NMF with random initialization to result in less error than the specially seeded NMF. We now suggest an application of the structured initialization process that seeks to benefit from the short-term behavior encountered here.

First we note that while the iterations of both Spherical K-Means and NMF algorithms are $\mathcal{O}(mnr)$, the Spherical K-Means algorithm requirements exhibit a lower constant of the leading term.[12] A further reduction in computational costs of the Spherical K-Means algorithm may be obtained using the Comparison Refinement provided in Chapter 3. Additionally, we have shown that the Spherical K-Means algorithm requires far fewer iterations to converge than the three NMF algorithms of Chapter 2. This property provides a fundamental insight into our suggested application.

Adaptive rank reduction methods, methods that may change the rank of the current approximation based on the progression/convergence of the resulting error, would be very beneficial to the common user [16]. The common user would have no justification for the a priori selection of a particular rank r of a low rank approximation. Adaptive methods would provide the user some flexibility in their selection, and would theoretically arrive at a low dimensional approximation that satisfies the user's error (Frobenius error) requirements. However, the area of adaptive NMF methods remains

[12] Combining additions and multiplications, the constant in front of the mnr term for the Spherical K-Means algorithm is 2, whereas the same constant is 12 in the case of the Euclidean distance NMF algorithm.

unexplored in the literature.[13] The difficulty of selecting r for NMF is made worse by the non-negative restrictions on $\mathbf{W}$ and $\mathbf{H}$. We remind the reader that these restrictions prevent us from using the behavior of the eigenvalues to offer insight into this problem.

The alternative to adaptive methods is to actually compute the NMF for several different values of r. While this could be very expensive (from a computational standpoint), a user focused on obtaining the special structure of the NMF output at a desired error level might repeatedly implement an NMF algorithm until the desired error is obtained. A particular disadvantage of this strategy is that, because of the significant error reduction that occurs in the early iterations of NMF (with random initializations), a user should not necessarily terminate the iterative process after a low number of iterations. However, in the case of Spherical K-Means, we have seen that a user can terminate the clustering algorithm before convergence is obtained, at a relatively small cost of error in the resulting NMF initialization.

Following this idea, we suggest that the Spherical K-Means initialization of $\mathbf{W}$, coupled with the corresponding best (NNLS) initialization of $\mathbf{H}$, be used to obtain a rough indication on the expected NMF error. Since we still expect the first NMF iteration to result in a significant decrease in error, we will compute a single NMF iteration using this initialization to further improve the quality of the approximation. This process is shown for ranks 1–75 of the Classic3 data set in Figure 5.6. This figure shows how the suggested process results in error behavior that tracks the long-term NMF error, especially for low ranks, reasonably well.

The initialization-factorization process used is:

(1) Compute the centroid matrix $\mathbf{C}$ using Spherical K-Means. The Spherical K-Means algorithm may be terminated after a fixed number of iterations (before it has converged) to keep the cost of the step to a minimum.

[13] For clustering, a very recently published method [16] suggests changing the r (k) value of clustering techniques to overcome the locality of minimums obtained, and suggests the development of other adaptive clustering methods.

(2) Using this $\mathbf{C}$, compute the non-negative coefficient matrix $\mathbf{N}$ which minimizes $\|\mathbf{X} - \mathbf{CN}\|_F$ using the NNLS algorithm.

(3) Compute one iteration of the Euclidean distance NMF algorithm using the initializations $\mathbf{W}^{(0)} \equiv \mathbf{C}$ and $\mathbf{H}^{(0)} \equiv \mathbf{N}$.

We remark that each additional step in the above process brings us one step closer to the long-term error of NMF. For example, Theorem 2 allows us to immediately obtain an upper bound on the NMF error using the elementary NMF. Using the elementary NMF error (obtained after only computing the Spherical K-Means objective function Θ_{SKM}) and Lemma 2, we immediately know that:

$$\Theta_{NMF_E} \leq {\Theta_{NMF_E}}^E = 2\left(n - \Theta_{SKM}\right). \tag{5.13}$$

Computing a better left factor $\mathbf{H}$ using NNLS will then result in a lower error and the stars in Figure 5.6. Even though a single NMF iteration is expensive, we have seen that the very first (post-initialization) NMF iteration results in the greatest reduction of error and so this first iteration has been computed to produce the dots in Figure 5.6.

We now illustrate the initialization-factorization technique suggested in the preceding paragraph. Figure 5.7 shows the initialization-factorization results for ranks 1–30 on the Faces data set. First we note that the average number of Spherical K-Means iterations required to obtain the optimal clustering for these 30 $r = k$ values was 11.74. Assuming that the NNLS calculation requires the same number of operations as a single NMF iteration, our initialization-factorization process for this data set is equivalent to computing approximately 4 NMF iterations. Aggregating over all values of r, we then need the same number of operations as approximately 120 NMF iterations to generate the initialization-factorization points in Figure 5.7. This computational expense is less than that required to obtain the long-term NMF error for a single r value. Using the dots in Figure 5.7, we then see that for $r = 24$ there is guaranteed to exist (we have just constructed one) a Non-negative Matrix Factorization whose relative error is less than the 17.1%.

The user may wish to further improve upon this error (and the corresponding structure of the NMF factors) by now computing the long-term NMF using random initializations. The user benefits from the efficient calculation of r by only having to compute this long-term NMF for one r value and, in the case of $r = 24$, will obtain a long-term relative error of 14.1%. This selection of r may be implemented for any particular desired error. If instead, a user wanted a factorization resulting in a relative error less than 20%, the initialization-factorization process would ensure that there existed such a factorization for $r = 4$. This process allows for the efficient calculation of the r value that satisfies the error demands while minimizing the storage requirements. At the very least, the initialization-factorization process provides a loose upper bound on the error that we expect to achieve by performing the costly NMF iterations.

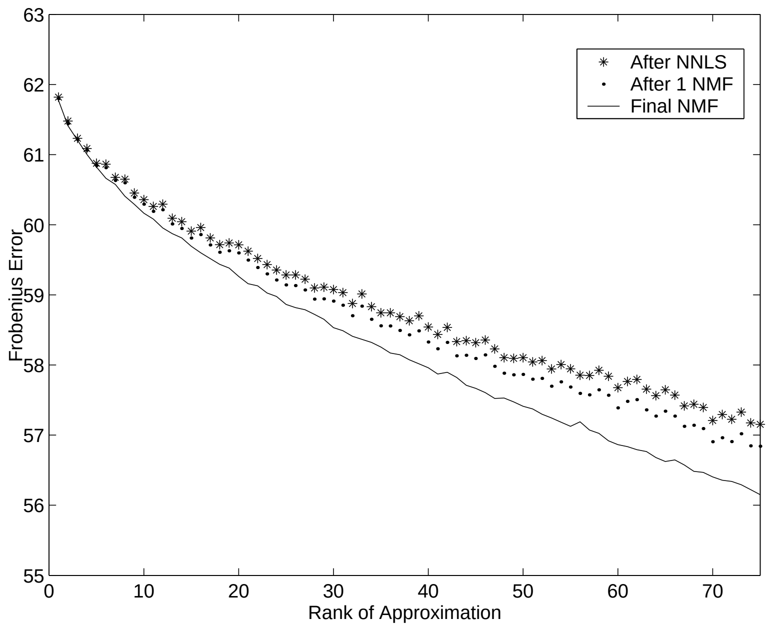


Figure 5.6: Classic3: Comparing the long-term NMF error with that obtained after (1) cluster-NNLS and (2) a single iteration of NMF using cluster-NNLS as initialization.

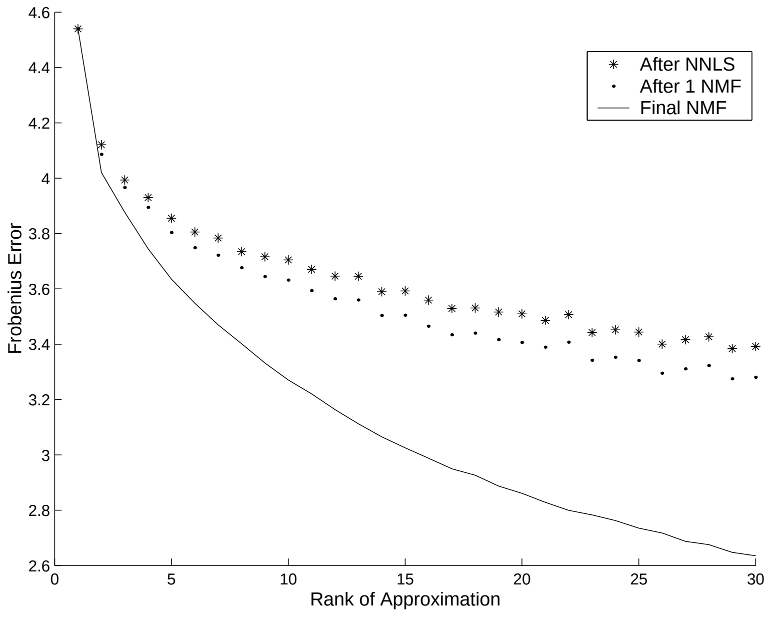


Figure 5.7: Faces: Comparing the long-term NMF error with that obtained after (1) cluster-NNLS and (2) a single iteration of NMF using cluster-NNLS as initialization.

Chapter 6

AVIRIS Results

In this chapter we motivate the application of Non-negative Matrix Factorization to other types of data. We first introduce a remote sensing device called AVIRIS and the data that it collects. In particular, we look at a $4km \times 4km$ region over Cuprite, Nevada. We then perform NMF with random initialization on this data set to obtain a basis of spectral profiles and point out some shortcomings of the basis obtained. We correct these shortcomings by seeding the NMF with the Spherical K-Means clustering and explore the new results. It is our goal to use these results to "unmix" the spectral AVIRIS data in an unsupervised manner that is new to the field of remote sensing.

6.1 AVIRIS Data Set

Text and images make up the standard data corpus used to test Non-negative Matrix Factorization and clustering techniques. We now introduce a third data set intended to illustrate the robustness of the techniques proposed. In this thesis, it is our goal that the reader be inspired by this application of NMF to a physical data set and to emphasize that the suggested methods not be limited to text and image data.

The AVIRIS (Airborne Visible/Infrared Imaging Spectrometer) first began operations aboard a NASA research craft in 1987. It has since become a standard instrument used for remote sensing of the Earth. The AVIRIS project collects spectral radiance data for characterization of the Earth's surface and atmosphere. AVIRIS data is often

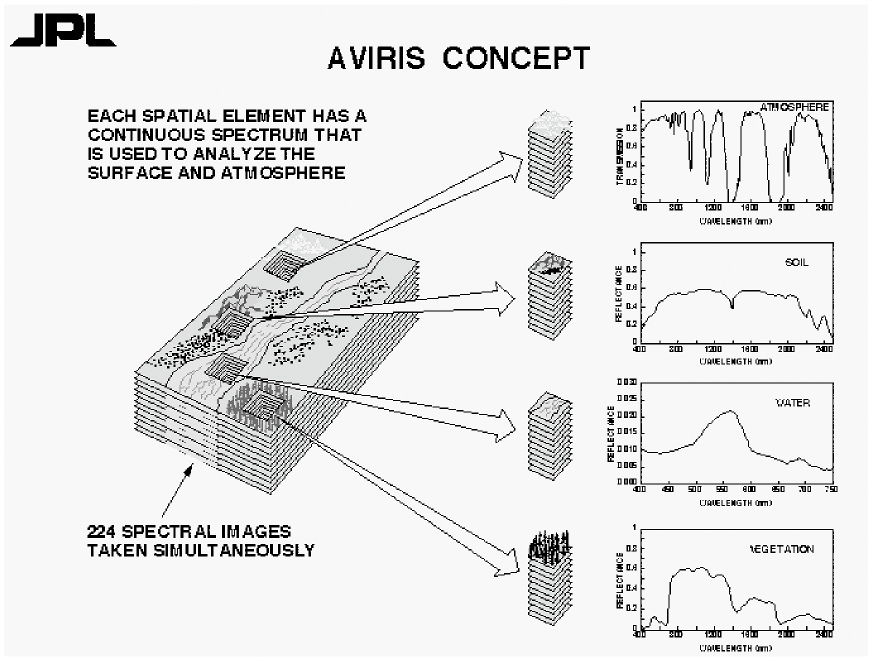

JPL
AVIRIS CONCEPT
EACH SPATIAL ELEMENT HAS A CONTINUOUS SPECTRUM THAT IS USED TO ANALYZE THE SURFACE AND ATMOSPHERE
224 SPECTRAL IMAGES TAKEN SIMULTANEOUSLY
ATMOSPHERE
TRANSMISSION
SOIL
REFLECTANCE
WATER
VEGETATION
WAVELENGTH (nm)

Figure 6.1: Three dimensional cube of spectral data from AVIRIS [39].

used in the fields of oceanography, environmental science, snow hydrology, geology, volcanology, soil and land management, atmospheric and aerosol studies, agriculture, and limnology. Applications under development include the assessment and monitoring of environmental hazards such as toxic waste, oil spills, and land/air/water pollution [39].

As a NASA ER-2 jet equipped with AVIRIS flies over an area, it takes spectral images of the Earth's surface so that each pixel corresponds to a 20 meter by 20 meter[1] land plot. For each location (pixel) reflectance, measured by the ratio of outgoing to incoming radiances, data for 224 contiguous spectral channels (bands), each about 10 nanometers wide, is recorded. These wavelengths (from 400 to 2500 nanometers) correspond to visible, near and short-wavelength radiation and are used for quantitative characterization of surface features. See Figure 6.1 for an illustration of the

[1] 20 meter by 20 meter pixels are obtained when flying over surfaces at sea level. Due to the fixed field of view of AVIRIS, as the altitude of the surfaces below AVIRIS increases, the surface area that each pixel represents becomes smaller. For example, the Cuprite, NV site introduced later with an approximate elevation of 1500m above sea level will actually produce 18 meter by 18 meter pixels when imaged [20]. For the purposes of this document we will assume that each pixel represents a 400 square-meter square plot.

3-dimensional cube of data generated.

One area that has received extensive study with AVIRIS flights is Cuprite, NV ([3], [4], [20], [21], [44]). Throughout this location are scattered several small orebodies and the various geological and mineral features of the area are well documented from groundmapping. The entire Cuprite, NV site that we have worked with is shown in Figure 6.2. This image consists of a total of 314,368 pixels (614 pixels wide by 512 tall), each with a spectral profile consisting of 224 data points as shown in Figure 6.3.

Some preprocessing of the data is needed for the methods described in this paper. First, for purposes of illustration, we have chosen to work with a small subset of the Cuprite, NV site. In this paper, we will only consider the 200×200 pixel square in the center of Figure 6.2. Further, we have removed bands numbered 107–114 and 152–169. These bands correspond to the 1400nm and 1900nm regions, respectively, where water vapor is so strongly absorbing that few photons reach the surface and even fewer return to the AVIRIS sensor [21]. The noise at these two wavelength regions (quantified by a very low signal-to-noise ratio) is shown in the example spectral profile obtained by AVIRIS in Figure 6.3.

Additionally, some noise may have led to slightly negative values of reflectance for a couple of the lower band numbers. For these very low bands (band numbers 1–2) it is very difficult to calibrate and atmospherically correct the AVIRIS output. For the purposes of this paper, when negative values are encountered, they are replaced with zeros. This leaves us with the $198 \times 40,000$ band by pixel[2] matrix $\mathbf{A}$. From a numerical precision standpoint, the reflectance data recorded by AVIRIS takes on integer values from 1 to 10,000. Before computing NMF, we normalize each pixel to be of unit length so that each pixel's spectral profile contributes equally to the collection

[2] The reader should be careful with the use of the word pixel here. Recall that each of the m pixels from a facial image corresponded to the m features in the $m \times n$ matrix $\mathbf{F}$. Here, each of the n pixels (locations) corresponds to a "document" (or face) with m spectral features in the $m \times n$ matrix $\mathbf{A}$. The reader may temporarily ignore the fact that each of these m-long arrays came from an actual pixel of the image in Figure 6.2.

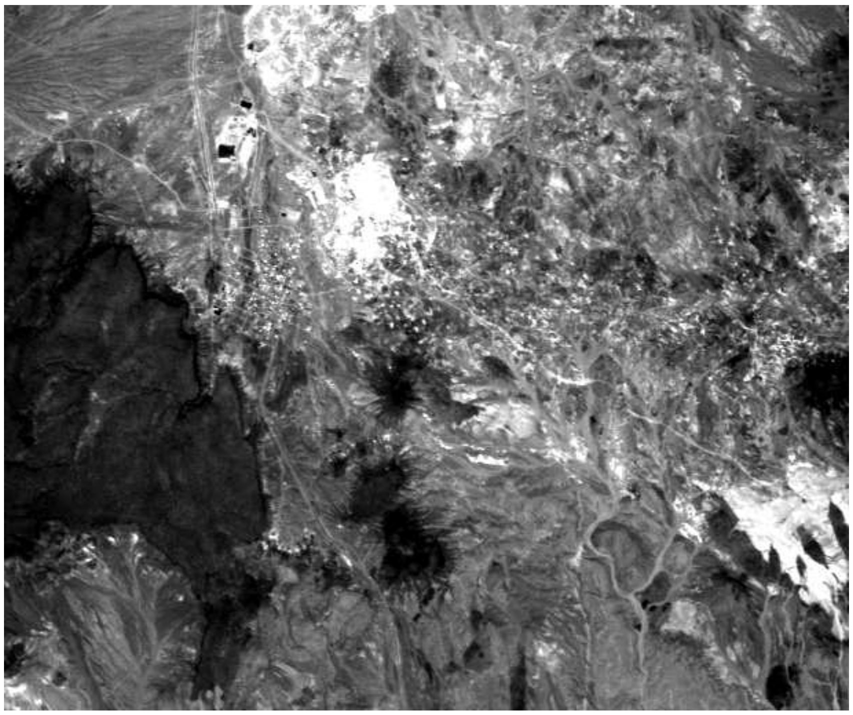


Figure 6.2: 614 × 512 spectral image of the Cuprite, NV region.

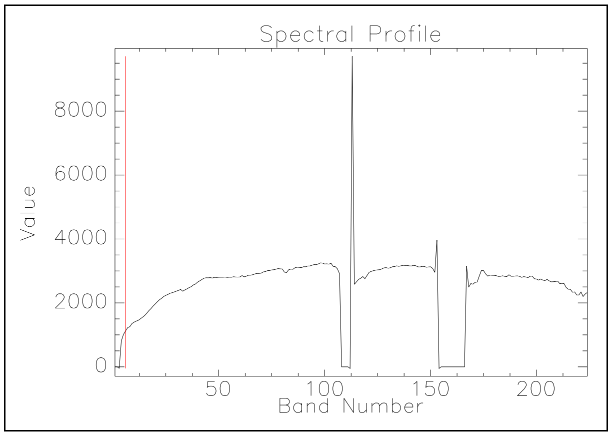


Figure 6.3: Sample spectral profile for "Pixel #1" of Cuprite, NV.

of 40,000 profiles. Consequently, only the relative differences between reflectance at different bands matters and not the overall magnitude.

6.2 Feature Extraction on the AVIRIS Data Set

We now wish to examine NMF applied to the AVIRIS data set. Recall that each point in this data set corresponds to a physical location on a 200×200 pixel image. Each point is characterized by 198 features, each corresponding to a different wavelength (in nanometers). The value that each feature takes on is the reflectance returned to an AVIRIS sensor for that particular wavelength emission.

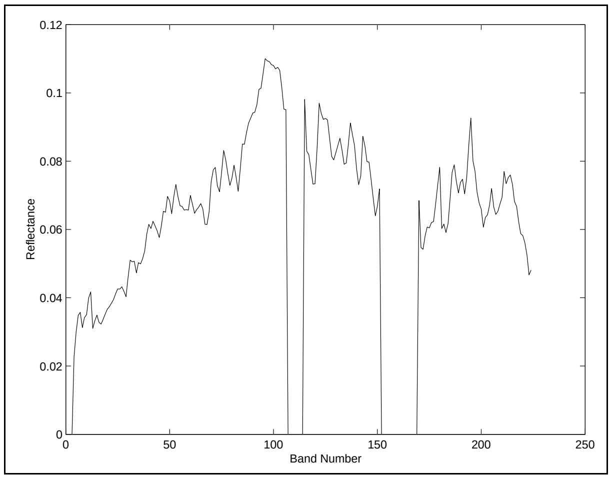


Figure 6.4: Sample (preprocessed) spectral profile for a given location.

We also emphasize that reflectance is a continuous[3] function of wavelength and that each feature (band number) corresponds to a sampling (taken every micrometer) of a particular location's spectral profile. An example of one such profile after our preprocessing has been applied is shown in Figure 6.4.

Due to the limitations of AVIRIS imagery, each location actually consists of a 20 meter by 20 meter square. It is our goal to improve this resolution by doing sub-pixel

[3] Subsequent plots may indicate that there are two places where discontinuities in reflectance occur. However, these small jumps are the result of the truncation that was done to eliminate the wavelengths whose signal-to-noise ratio is too low (see previous section).

extraction. Based on a location's spectral profile, we would like to determine what primary physical components exist within the 400 square meter area that the profile represents. To determine these components we will perform NMF on the 40000 locations in the AVIRIS data set. Using this many locations, we hope to obtain the $k = 12$ (chosen empirically) components that could best be added together to reconstruct each location's profile as closely as possible.

6.2.1 Random Initialization

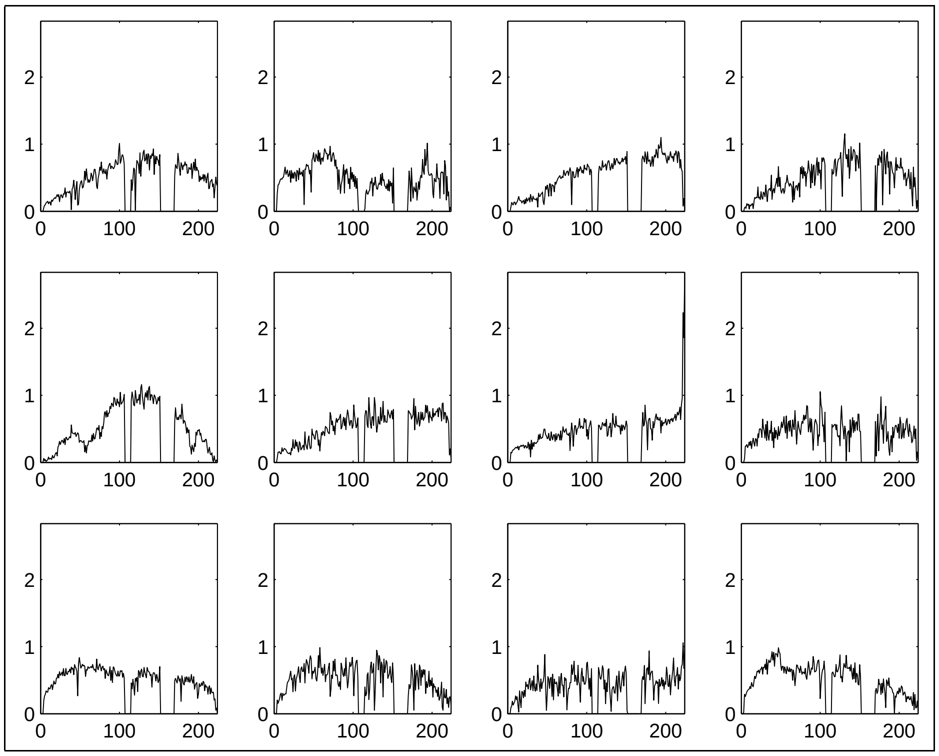


Figure 6.5: NMF basis profiles with random initialization.

Using random initializations for **W** and **H**, 300 iterations of the Euclidean distance NMF algorithm were performed to obtain the spectral basis in Figure 6.5. The Frobenius error of the approximation induced at each iteration is shown in Figure 6.6 to illustrate the convergence of NMF for this data set and initialization. The basis profiles

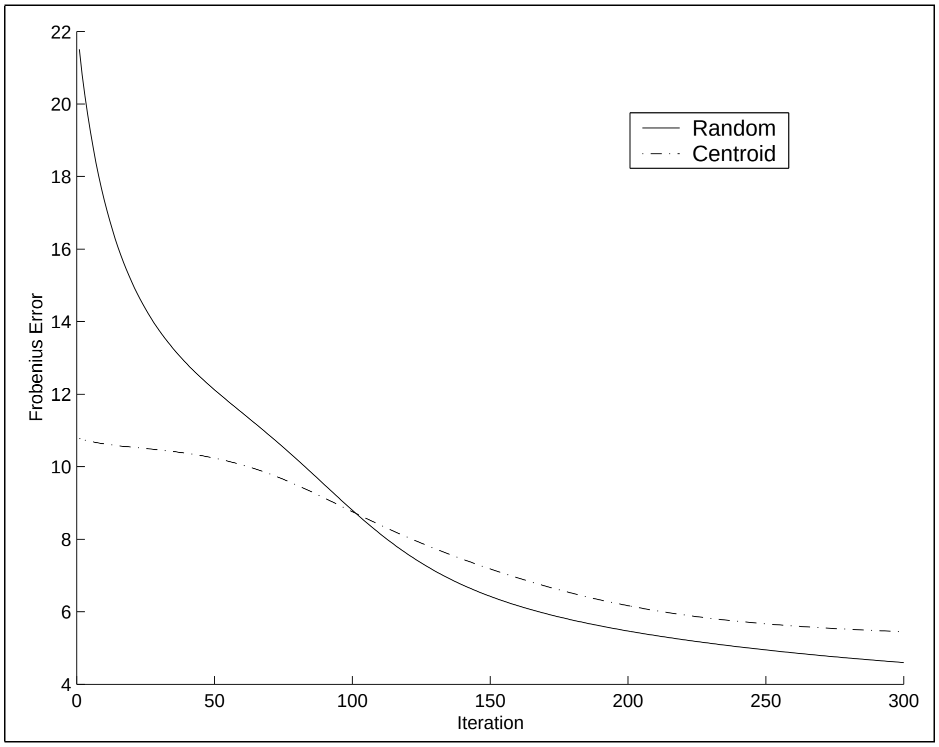


Figure 6.6: Error of NMF approximation for both random and centroid initializations.

shown here no longer maintain the continuity originally seen in Figure 6.4. This is not to say that the basis profiles found will not sum up to reconstruct the original profiles. The basis profiles found are just corrupted with noise and do not maintain the clear structure that we would prefer to obtain. A set of continuous basis profiles would allow us to determine if they correspond to other known surfaces (i.e. sand, clay, vegetation, etc.). In an effort to remove some of this noise, in the next section we will use the Spherical K-Means structured initialization whose error is shown in Figure 6.6.

6.2.2 Spherical K-Means Initialization

We now apply the proposed Spherical K-Means initialization to the AVIRIS data set. We will again use the Euclidean distance NMF algorithm with $k = 12$ and the same random initialization for $\mathbf{H}$. However, we will use the centroids obtained after approximately 50 iterations of Spherical K-Means Clustering, $C^{(50)}$ to seed the basis $\mathbf{W}$.

Figure 6.7 shows the $k = 12$ centroids that result after 50 iterations of Spherical K-Means clustering. Since these 12 profiles represent "average" profiles representing similar locations, we do expect to see continuity maintained. Further, these profiles are much smoother than even the original spectral profiles – in grouping (clustering) similar profiles together and taking an average we have substantially reduced the error inherent in AVIRIS data. These are now the 12 profiles that will be used to initialize the profile basis $\mathbf{W}$.

We also hope that the clustering obtained will have some correspondence with the physical locations where each of the 40,000 spectral profiles were obtained from. Figure 6.9 graphically shows the clusterings obtained. In this figure, we are confined to a palette of 12 colors, each corresponding to a different cluster – each of the 40,000 pixels have the shading that represents the cluster they belong to. For example, all pixels that are nearly white (like those in the body near the upper right hand corner

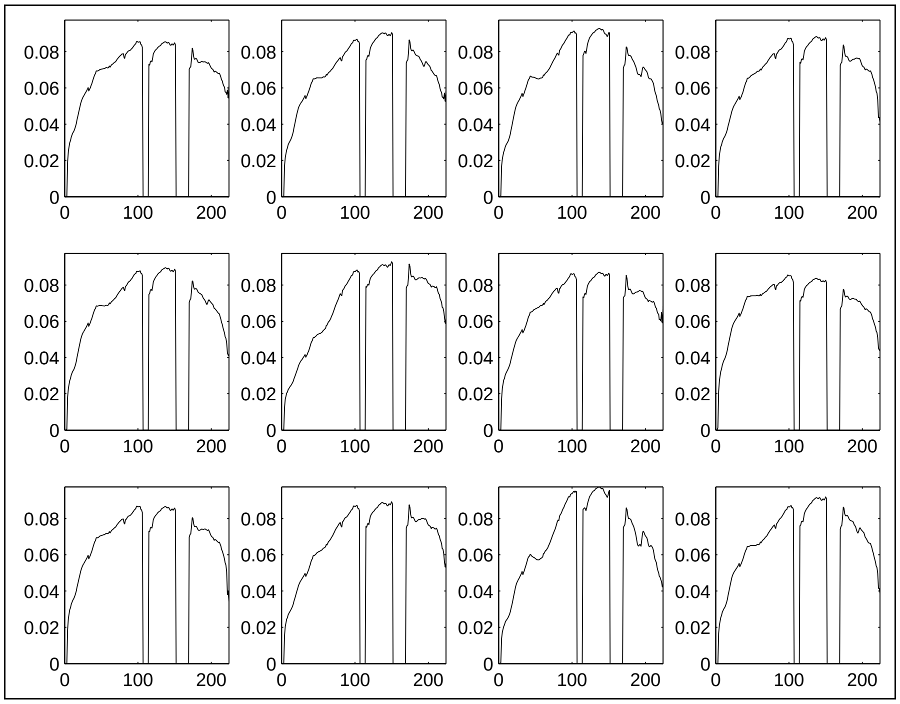


Figure 6.7: Centroids ($k = 12$) resulting from Spherical K-Means clustering of AVIRIS data set.

of Figure 6.9) were determined to be most similar by Spherical K-Means. Compare this picture with the one in Figure 6.8 which corresponds to the image of the original AVIRIS data set for a representative spectral band as obtained by ENVI [41]. Here one can clearly see that some of the original structures, such as large bodies of ore and the roadbed running through the valley in the center of the image, were returned as a result of clustering.

Using this Spherical K-Means initialization, 300 iterations of the Euclidean distance NMF algorithm yielded the basis profiles shown in Figure 6.10. These basis profiles should be compared to those obtained using a random initialization of $\mathbf{W}$ in Figure 6.5. These new profiles are much smoother (preserve continuity better) than those obtained using a random initialization. The reader should also note that these profiles are not the same as the centroid profiles of Figure 6.7. The profiles in Figure 6.10 resemble the centroids because they were originally initialized with them, however, the two sets of

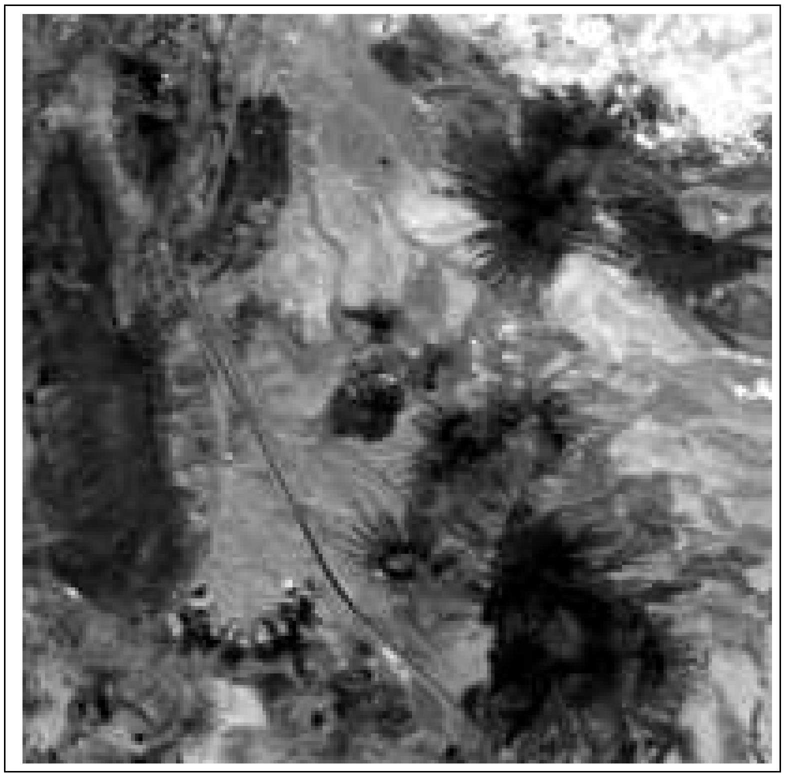


Figure 6.8: Original AVIRIS data set representation.

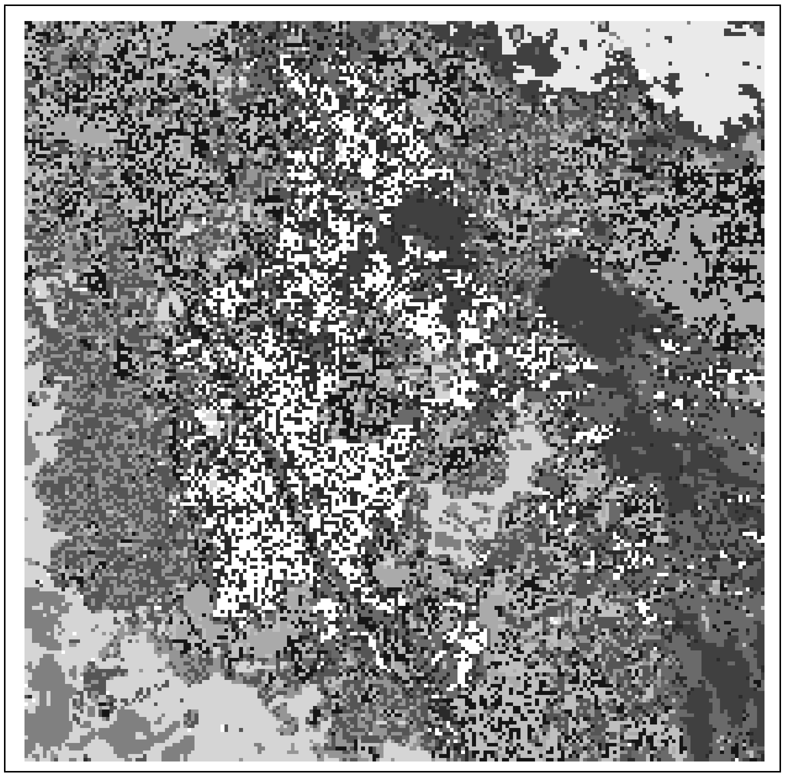


Figure 6.9: AVIRIS data set representation using Spherical K-Means ($k = 12$).

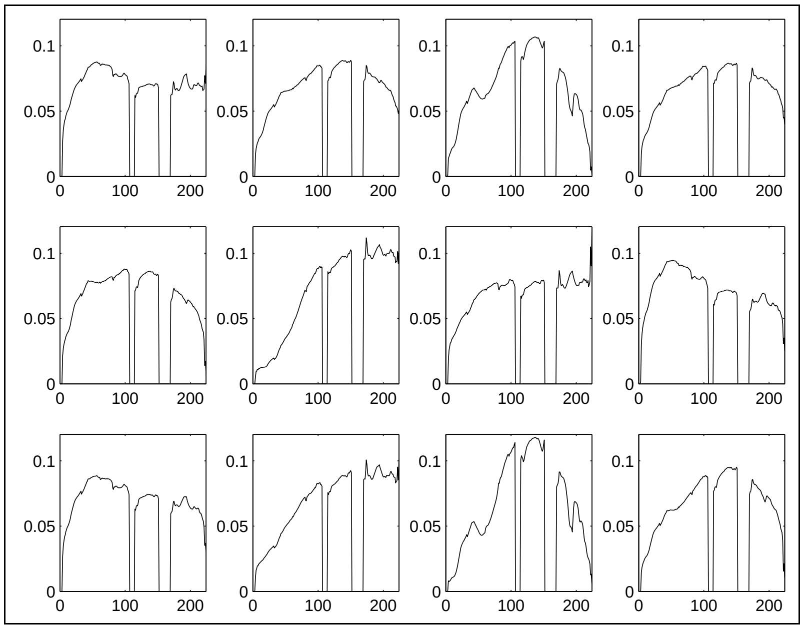


Figure 6.10: NMF basis profiles with the Spherical K-Means initialization.

profiles differ by their emphasis on certain features (marked by peaks and valleys in the spectral profiles).

Lastly, we comment about the error made by this rank 12 NMF approximation using the two different initializations. Both errors are shown in Figure 6.6. As expected, the Spherical K-Means initialization begins with significantly less error than its random counterpart. However, the specification of this restricted left factor slows down long-term convergence as shown in Chapter 5. What is more interesting to note about Figure 6.6 is that, after falling behind, the Spherical K-Means initialization catches up to the less restrictive random initialization. We conclude this section by remarking that these two approximations are very effective, reducing the rank from 196 to 12 with a relative Frobenius error of less than 2.5%.

6.3 Unmixing and Related Work

Lastly, we wish to emphasize how the initialization-factorization approach with Spherical K-Means and NMF proposed here differs from previous work done on AVIRIS images. Over the past decade, the increasingly wide-spread use of AVIRIS data has prompted several researchers to look into the area of data compaction. A more interesting problem is that of "unmixing" – overcoming the limited pixel resolution by determining what smaller sub-elements make up a pixel.

In the literature (see [3], [4], [21], and [20]), the spectral profiles of these sub-elements are called "endmembers". These endmembers are usually chosen from a library of known profiles taken from ground samples of different types of vegetation, minerals, metals, etc. For each endmember, the resulting abundance plot is generated, where each pixel of the plot signifies the abundance of that endmember in the pixel. The abundance of an endmember is necessarily non-negative. Consequently, Non-Negative Least Squares (NNLS) offers a very convenient way of obtaining the best non-negative least squares solution y to $\|Ey - x\|^2$, where E is the endmember library matrix whose columns are each endmembers.

Our work presents a new approach to this problem because we do not need to assume that a library of endmembers is already in place. Ours is an unsupervised method of actually deriving the k endmembers that best represent the given data when added together (with unequal weights). The resulting endmembers are stored in the basis matrix $\mathbf{W}$. Our technique also does not require additional NNLS computations because each abundance plot is already stored in a row of the coefficient matrix $\mathbf{H}$. This method only requires the specification of the parameter k, for which a method was introduced in Section 5.4.

One shortcoming of our method is that the columns of $\mathbf{H}$ do not sum to 1. Ideally, we would like the i-th element of column h_j to correspond to the percentage

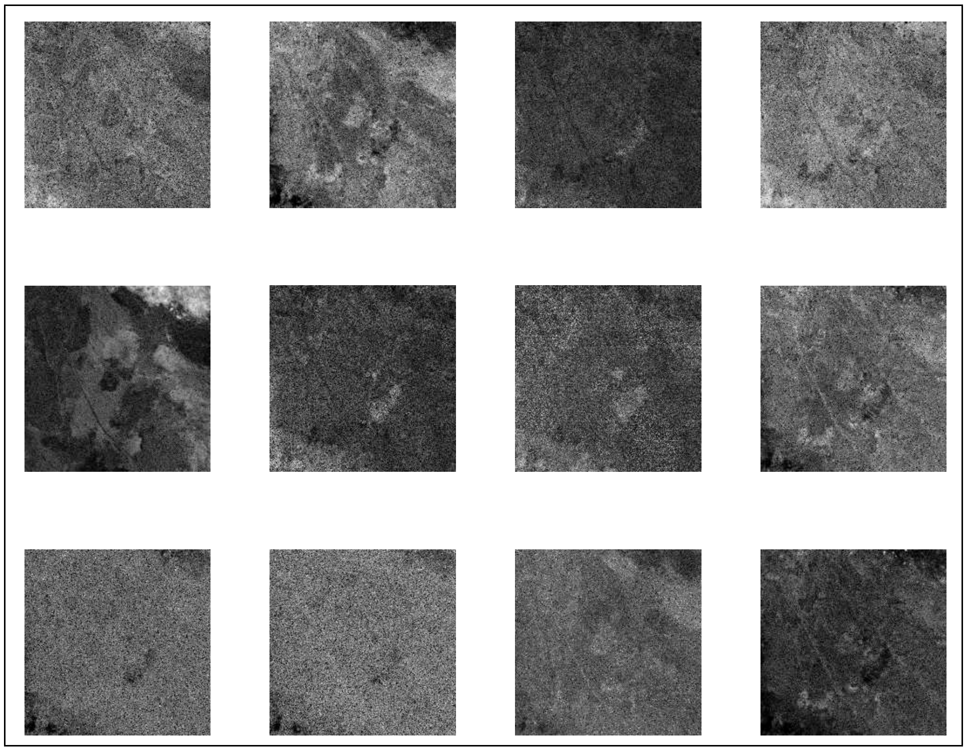


Figure 6.11: NMF coefficient matrix **H** with random initialization.

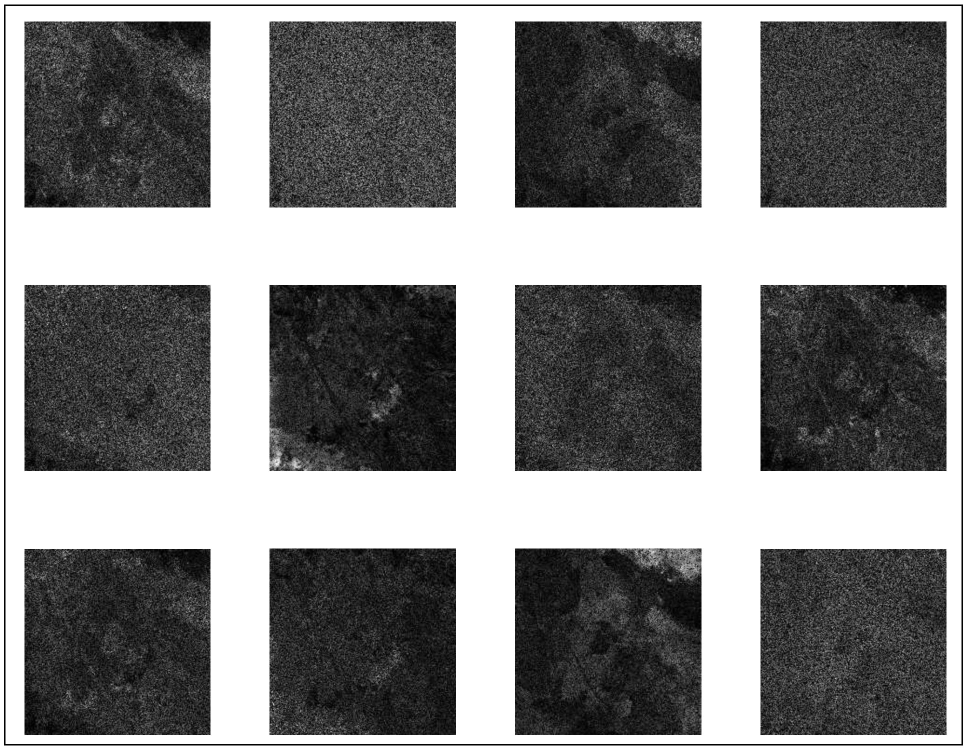


Figure 6.12: NMF coefficient matrix **H** with the Spherical K-Means initialization.

of pixel j that is made up of endmember i. The necessary scaling is not analogous to the normalization shown in Theorem 1 which applies to the rows of $\mathbf{H}$. Currently, each element in a column of $\mathbf{H}$ corresponds to the abundance of endmember i when viewed relative to all other endmembers.

To illustrate the concept of an abundance plot, we have shown the left factor $\mathbf{H}$ that resulted from the random and Spherical K-Means initializations of this chapter in Figure 6.11 and Figure 6.12, respectively. At first glance, some of these 12 abundance plots (each corresponding to the abundance of one of the 12 basis profiles, or endmembers) may appear to be entirely random. However, these abundance plots correspond to endmembers which are found throughout the image and whose endmembers probably resemble other endmembers associated with a random looking abundance plot. Further, the abundance plot is not entirely random because the ore body in the lower left is usually left black (corresponding to consisting of almost none of that particular endmember).

Even more interesting is the appearance of features such as the roadbed through the center of the image. For example, in looking at the second row of Figure 6.11, we see that the first endmember in the row shows the roadbed as with dark pixels while the last endmember in the row shows the roadbed as light pixels. Here we may conclude that the roadbed is significantly made up of the last endmember in the row while lacking in the first endmember of the row.

Before moving on to concluding remarks we emphasize that the spectral profiles (endmembers) that we have obtained may not make complete sense to us. In general, we are more familiar with the human faces and so we can recognize noses and ears when they appear in a face basis. However, an expert in remote sensing could likewise immediately identify the features corresponding with the spectral profile of water and could consequently make some skilled interpretation of the basis obtained. In this way, efficient Non-negative Matrix Factorization techniques may be applied to data from a variety of fields and yield both a basis and a coefficient matrix whose interpretation must then ultimately come from an expert in that field.

Chapter 7

Summary and Future Work

7.1 Summary and Conclusions

In this document we have explored two iterative techniques: Non-negative Matrix Factorization (NMF) and Spherical K-Means. We have shown how NMF overcomes the inadequacies of traditional rank reduction techniques of enforcing the non-negativity of both factors in the resulting low-dimensional representation. We have also illustrated, for a data set of facial images, the special structure of the output of NMF – eyes, ears, lips, et cetera. We have also shown the analogous representation obtained using the Spherical K-Means factorization.

Next, we took a closer look at the computational implementations of both NMF and Spherical K-Means and proved a theorem allowing the normalization of the NMF basis. For both techniques, we examined the algorithms traditionally used and explored related issues such as the convergence, computational complexity and initialization of these algorithms. These results were illustrated using a collection of facial images. In presenting these results and addressing the computational considerations, we motivated the structured initialization of NMF to help improve both the structure and speed of current NMF algorithms.

We also proved several results addressing the relationship between the NMF factorization process and the output of the Spherical K-Means. These results suggested using the Spherical K-Means clustering output as a possible structured initialization of

NMF algorithms. We explored this initialization, examining the progression of the resulting basis. We also found that this structured initialization resulted in less short-term error than current random initializations but that this initialization was too restrictive to maintain this advantage in the long-term (i.e. after several hundred iterations). However, the results obtained suggested using this structured initialization technique as a relatively efficient determinant of the rank of the desired NMF approximation. This suggested application was illustrated in Chapter 5 using two of the data sets of this document.

We have also shown preliminary results for NMF on AVIRIS remote sensing data. These results emphasize the benefit of using a structured initialization (in this case with the Spherical K-Means clustering output) for NMF problems. In the end, NMF allowed us to find a basis whose profiles emphasize local spectral features. These profiles are viewed as particular physical surface types by an expert in remote sensing, in a manner analogous to the way we view the NMF output for facial collections to be specific facial features. We have used the spectral basis obtained to unmix the low-resolution remote sensing data to determine at what level each of these features is present at a particular location. We hope that these results motivate the application of NMF to other non-standard types of data.

7.2 Future Work

Much work remains to be done in the area of Non-negative Matrix Factorizations. In Chapter 2 we noted that an NMF particular factorization is determined by the objective function which it seeks to minimize (maximize). There are many other available, and easily computable, distance measures that could feasibly be used to create an NMF objective function. An issue coupled with this idea of finding suitable objective functions is the actual implementation of the minimization or maximization of the objective function. Recall that many update strategies that maintain monotonicity are possible for each objective function. Because of the direct dependencies between objective function (theoretical error measure) and update strategy (computational implementation), with the exception of LNMF, no new NMF algorithms have been proposed since the original introduction of NMF in 1997.

In Chapter 4, we have seen the dramatic dissimilarity in both error and basis structure for different NMF algorithms. These differences suggest that other NMF algorithms could be developed to address other desirable properties of an NMF factorization. A thorough exploration of alternative objective functions and their potential update strategies is needed. The new updates that would follow from this would allow a user to select an NMF algorithm tailored to their own data set and/or goals.

The specific details of the convergence of even the current NMF algorithms has yet to be theoretically examined. In our experience, behavior of the monotonic decrease of the three objective functions used in practice varied greatly from data set to data set and so a general (non data specific) result may not be possible. However, a proper examination of the behavior of NMF algorithms may be able to characterize the large amount of work done by NMF in early iterations and the subtle shifting that follows in later iterations. This analysis would be a tremendous benefit to practical NMF implementation because it might allow us to significantly reduce the number of operations required for later iterations. The idea here would be to prove a technique similar to the "Comparison Refinement" for Spherical K-Means shown in Chapter 3. This refinement would exist in the much finer (but still discrete) space of the two NMF factors, $\mathbf{W}$ and $\mathbf{H}$, and would only require computations for those factor elements where data is actively moving to/from. Current NMF algorithm iterations are by nature $\mathcal{O}(mnr)$, and so such a contribution would be of tremendous practical value.

As the NMF theory evolves, we also believe that it will be critical to create alternate formulations of the NMF problem than the EM approach currently used. Alternate formulations could suggest more direct methods of obtaining a constrained factorization. For example, the Euclidean distance objective function could have alternately been posed as an optimization problem with a linear feasibility region and nonlinear objective function. If discretization (as a result of, for example, a limited palette of colors for images) is further considered, the combinatorial properties exploited by discrete optimization techniques could become increasingly useful.

Much work remains to be done on the initialization of NMF algorithms. In Chapter 5 we introduced one particular structured NMF initialization using the centroids of Spherical K-Means clustering. The results obtained show that structured initialization does provide certain (particularly short-term) benefits. Our results also suggest that there are other structured initializations which may do even better while still retaining some freedom for subsequent NMF iterations. However it is unclear if such an initialization will be easier to come by than actually computing standard NMF iterations.

Along this line, the geometrical interpretation of Spherical K-Means may also be useful for making minor modifications to the collection of centroids in order to obtain a "wider" cone in the positive orthant. Further, we would like to develop an "NMF objective function-update strategy pair" that has an even nicer relationship with the properties of the optimal Spherical K-Means clustering. For example, an NMF algorithm based on the angles between a data vector and its approximation (resulting in a linear objective function and nonlinear feasibility region) might be well suited for the centroid initialization. Such an NMF algorithm might offer insight into what modifications to the Spherical K-Means centroids should be applied to condition them to be a better initialization for any NMF algorithm.

Finally, the physical data set in Chapter 6 motivates the application of NMF to a variety of nonstandard types of non-negative data. For each different type of data, we may have differing goals for computing a Non-negative Matrix Factorization. One possible way to address these varying goals is to tailor objective function-update strategy pairs to the specific desired output structure. Such an approach would lead to a seemingly “endless” number of NMF algorithms customized for a particular type of data. We believe there are many, as of yet untested, types of data whose NMF output is significantly more valuable than that obtained for current standard data sets. We intend to further pursue the NMF output for AVIRIS remote sensing data as well as other data arising from physical phenomena.

Much work remains to be done in the area of Non-negative Matrix Factorizations. We have made some contributions, but feel that we have discovered more questions than answers. We hope that the problems raised here will inspire others to address (especially from a mathematical viewpoint) some of the interesting topics touched on. We predict that with the increasingly widespread use of large structured data sets, the constrained optimization techniques of NMF will continue to be in high demand by data analysts in a variety of fields.

Bibliography

[1] M.W. Berry and M. Browne. Understanding Search Engines: Mathematical Modeling and Text Retrieval. SIAM Press, 1999.

[2] M.W. Berry, S. Howard, and H. Tang. GTP: General Text Parser. Available at http://www.cs.utk.edu/lsi/. 2000.

[3] J.W. Boardman. Automating Spectral Unmixing of AVIRIS Data Using Convex Geometry Concepts. In AVIRIS Airborne Geoscience Workshop Proceedings, 1993.

[4] J.W. Boardman, F.A. Kruse, and R.O. Green. Mapping Target Signatures Via Partial Unmixing of AVIRIS Data. In AVIRIS Airborne Geoscience Workshop Proceedings, 1995.

[5] D. Boley. Principal direction divisive partitioning. Data Mining and Knowledge Discovery, 2(4):325–344, 1998.

[6] L. Bottou and Y. Bengio. Convergence properties of the K-means algorithms. In G. Tesauro, D. Touretzky, and T. Leen, editors, Advances in Neural Information Processing Systems, volume 7, pages 585–592. The MIT Press, 1995.

[7] P.S. Bradley and U.M. Fayyad. Refining initial points for K-Means clustering. In Proceedings of the 15th International Conference on Machine Learning, pages 91–99. Morgan Kaufmann, San Francisco, CA, 1998.

[8] M.T. Chu and R.E. Funderlic. The centroid decomposition: Relationships between discrete variational decompositions and SVDs. SIAM Journal on Matrix Analysis and Applications, Vol. 23, No. 4:1025–1044, 2002.

[9] Cranfield Cisi and Medline. Classic3 data set. Available at ftp://ftp.cs.cornell.edu/pub/smart/. 2002.

[10] S.C. Deerwester, S.T. Dumais, T.K. Landauer, G.W. Furnas, and R.A. Harshman. Indexing by latent semantic analysis. Journal of the American Society of Information Science, 41(6):391–407, 1990.

[11] M.M. Dempster, N.M. Laird, and D.B. Jain. Maximum likelihood from incomplete data via the EM algorithm. Journal of the Royal Statistical Society, 39:1–38, 1977.

[12] I.S. Dhillon, J. Fan, and Y. Guan. Efficient clustering of very large document collections. In R. Grossman, G. Kamath, and R. Naburu, editors, Data Mining for Scientific and Engineering Applications, Kluwer Academic Publishers, 2001.

[13] I.S. Dhillon, Y. Guan, and J. Kogan. Refining clusters in high dimensional text data. In Workshop on Clustering High Dimensional Data, Second SIAM International Conference on Data Mining, April 2002.

[14] I.S. Dhillon, Y. Guan, and J. Kogan. Iterative clustering of high dimensional text data augmented by local search. In Proceedings of the 2nd IEEE International Conference on Data Mining. Maebishi, Japan, December 2002.

[15] I.S. Dhillon and D.S. Modha. Concept decompositions for large sparse text data using clustering. Machine Learning, 42(1):143–175, 2001.

[16] C. Ding, X. He, H. Zha, and H. Simon. Adaptive dimension reduction for clustering high dimensional data. In Proceedings of ICDM 2002, pages 107–114, 2002.

[17] E. Forgy. Cluster analysis of multivariate data: Efficiency versus interpretability of classification. Biometrics, 21:768–780, 1965.

[18] W.B. Frakes and R. Baeza-Yates, editors. Information Retrieval: Data Structures and Algorithms. Prentice Hall, Englewood Cliffs, NJ, 1992.

[19] M.R. Garey and D.S. Johnson. Computers and Intractability: A Guide to the Theory of NP-Completeness. W.H. Freeman, New York, 1979.

[20] A.F.H. Goetz and B. Kindel. Understanding Unmixed AVIRIS Images in Cuprite, NV, Using Coincedent Hydice Data. In AVIRIS Airborne Geoscience Workshop Proceedings, 1996.

[21] A.F.H. Goetz and B. Kindel. Comparison of Unmixing Results Derived from AVIRIS, High and Low Resolution, and HYDICE Images at Cuprite, NV. In AVIRIS Airborne Geoscience Workshop Proceedings, 1999.

[22] G.H. Golub and C.F.Ṽan Loan. Matrix Computations. Johns Hopkins University Press: Baltimore, MD, 3rd edition, 1996.

[23] D. Guillamet, B. Schiele, and J. Vitri. Analyzing non-negative matrix factorization for image classification. In Proceedings to the 16th International Conference on Pattern Recognition (ICPR'02), volume II, pages 116–119. IEEE Computer Society, August 2002.

[24] D. Guillamet and J. Vitri. Determining a suitable metric when using non-negative matrix factorization. In Proceedings to the 16th International Conference on Pattern Recognition (ICPR'02), volume II, pages 128–131. IEEE Computer Society, August 2002.

[25] J.A. Hartigan. Clustering Algorithms. Wiley: New York, 1975.

[26] I.T. Jolliffe. Principal Component Analysis. Springer-Verlag, New York, 1986.

[27] T. Kawamoto, K. Hotta, T. Mishima, J. Fujiki, M. Tanaka, and T. Kurita. Estimation of single tones from chord sounds using non-negative matrix factorization. Neural Network World, 3/00:429–436, July, 2000.

[28] J. Kleinberg, C. Papadimitriou, and P. Raghavan. A microeconomic view of data mining. J. Data Mining and Knowledge Discovery, 1999.

[29] T.G. Kolda. Limited-memory matrix methods with applications. Technical Report CS-TR-3806, University of Maryland, College Park, 1997.

[30] T.G. Kolda and D.P. O'Leary. A semidiscrete matrix decomposition for latent semantic indexing information retrieval. ACM Transactions on Information Systems, 16(4):322–346, 1998.

[31] W. Lang. On generalizations of the Stirling number triangles. Journal of Integer Sequences, Vol. 3, 00.2.4, 2000.

[32] C.L. Lawson and R.J. Hanson. Solving Least Squares Problems. Prentice–Hall, Englewood Cliffs, 1974.

[33] D.D. Lee and H.S. Seung. Unsupervised learning by convex and conic coding. In M.C. Mozer, M.I. Jordan, and T. Petsche, editors, Advances in Neural Information Processing Systems, volume 9, pages 515–521. The MIT Press, 1997.

[34] D.D. Lee and H.S. Seung. Algorithms for non-negative matrix factorization. In Advances in Neural Information Processing Systems, volume 13, pages 556–562, 2000.

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

[36] S.Z. Li, X.W. Hou, and H.J. Zhang. Learning spatially localized, parts-based representation. In Proceedings of IEEE International Conference on Computer Vision and Pattern Recognition, December 2001.

[37] Mathworks Inc. MATLAB: Matrix Laboratory. Computer software, 2001.

[38] D.S. Modha and S. Spangler. Feature weighting in k-means clustering, 2003.

[39] NASA Jet Propulsion Laboratory. AVIRIS: Airborne Visible InfraRed Imaging Spectrometer. Available at http://popo.jpl.nasa.gov/html/aviris.overview.html. 2003.

[40] University of Stirling. Psychological image collection at stirling. Available at http://pics.psych.stir.ac.uk/. 2002.

[41] Research Systems Inc. ENVI: The Environment for Visualizing Images. Computer software, 2002.

[42] J.B. Rosen. Minimum and basic solutions to singular linear systems. J. of SIAM, Vol. 12, Issue 1:156–162, 1964.

[43] L.K. Saul and D.D. Lee. Multiplicative updates for classification by mixture models. In T.G. Dietterich, S. Becker, and Z. Ghahramani, editors, Advances in Neural Information Processing Systems 14. MIT Press: Cambridge, MA, 2002.

[44] A.S. Warner, A.F.H. Goetz, K.B. Heidebrecht, and E.L. Johnson. Measuring the Ability of Landsat 7 to Map Vegetative Fractions Using a Near-Simultaneous AVIRIS Underflight. In AVIRIS Airborne Geoscience Workshop Proceedings, 2000.

[45] S. Wild, J. Curry, and A. Dougherty. Motivating non-negative matrix factorizations. In Proceedings of 2003 SIAM Applied Linear Algebra Conference, 2003.

Appendix A

Computing Specifications

Throughout this document, we have implemented a variety of algorithms and techniques. All algorithms were encoded in MATLAB 6.1 [37] with floating point operation counts confirmed in older versions of MATLAB. AVIRIS data was viewed in ENVI (Environment for Visualizing Images) 3.6 [41]. Three primary machines were used to perform the more computationally intensive experiments:

- AMD XP 2.0 GHz CPU, 1.5 Gb DDR RAM
- 4x 400MHz CPU, 4.0 Gb shared RAM
- 2x 2.0 GHz Intel Xeon CPU, 4.0 GB shared RAM[1]

We include this information in order to place all of the reported results in the context of the machines used.

[1] Computer funded by LLNL ASCI-2 grant, award number B347880.