Matrix Factorization: Nonnegativity, Sparsity and Independence

Vamsi Potluru

Vamsi Krishna Potluru

Candidate

Computer Science

Department

This dissertation is approved, and it is acceptable in quality and form for publication:

Approved by the Dissertation Committee:

Thomas Hayes, Chairperson

Vince Calhoun

Terran Lane

Barak Pearlmutter

Matrix Factorization : Nonnegativity, Sparsity and Independence

by

Vamsi Krishna Potluru

M.S., Computer Science, University of New Mexico, 2008

DISSERTATION

Submitted in Partial Fulfillment of the
Requirements for the Degree of

Doctor of Philosophy
Computer Science

The University of New Mexico

Albuquerque, New Mexico

May, 2014

Dedication

Dattatreya

Acknowledgments

I would like to thank my advisors: Prof's Barak Pearlmutter, Vince Calhoun, Thomas Hayes and Shuang Luan. Also, I would like to acknowledge the members of the brain and computational (BCL) lab, Hamilton institute, MIALAB at the Mind Research Network (MRN), and the Mitsubishi Electric Research Labs (MERL).

Contents

List of Figures **x**

List of Tables **xvi**

1 Introduction **1**

1.0.1 Big data 4

1.0.2 Distributed systems 4

1.1 Background 6

1.1.1 Least Absolute Shrinkage and Selection Operator (LASSO) . . . 6

1.1.2 Support Vector Machine (SVM) 6

1.1.3 Nonnegative Least Squares (NNLS) 7

1.1.4 Nonnnegative Matrix Factorization (NMF) 8

1.2 Contributions of this dissertation 9

2 Nonnegativity — Structural connections **10**

2.1 Our Reduction from TNNLS to SVM 14

2.1.1 Implementation issues 15

2.2 Primal SVM solver 16

2.2.1 Bounds 17

2.3 Experiments 18

2.3.1 Random problems 20

2.3.2 Phantom tumor dataset 21

2.3.3 Real tumor datasets 22

2.3.4 Discussion 24

2.4 Dual SVM solver 24

2.5 Experiments 26

2.6 Connections between NMF and SVM 27

2.6.1 SVM as matrix factorization 27

2.6.2 NMF reduced to sequence of SVMs 30

2.6.3 Our algorithms — Sign-insensitive Kernel SVMs 30

2.6.4 Nonnegative Quadratic Programming 33

2.6.5 Decomposition 36

2.6.6 Soft Margin SVM 39

2.6.7 Bias 40

2.6.8 Fixed points 41

2.6.9 Asymptotic convergence 42

2.6.10 Adapting NMF algorithms for SVM 43

2.6.11 Power methods 43

2.7 Experiments 44

2.7.1 Proofs 46

2.8 Conclusions and Future Work 51

3 Sparsity — Matrix factorization 53

3.1 Preliminaries and Previous Work 54

3.1.1 Nonnegative Matrix Factorization 55

3.1.2 Sparse Nonnegative Matrix Factorization 56

3.2 The Sequential Sparse NMF Algorithm 58

3.2.1 Sparse-opt 59

3.2.2 Sequential Approach —Block Coordinate Descent 60

3.2.3 SSNMF Algorithm for Sparse NMF 61

3.3 Implementation Issues 62

3.4 Experiments and Discussion 63

3.4.1 Datasets 63

3.4.2 Comparing Performances of Core Updates 64

3.4.3 Comparing Overall Performances 65

3.4.4 Main Results 67

3.5 Connections to Related Work 68

3.6 Bi-Sparse NMF 69

3.7 Conclusions 70

4 Independence — A closer look 71

4.1 Review and critique of the presented evidence 71

4.2 Experiments on synthetic data: boxes 73
4.3 The statistical properties of synthetic data [28] 74
4.4 Boxes revisited 77
4.5 Sparsity and sources that are mixture of Gaussians 79
4.6 ICA of sources with mixture of Gaussians distribution 81
4.7 On the definition of sparsity 84
4.8 On the application of ICA to fMRI 84
4.9 Conclusions 85

5 Conclusions and Future Work **88**

Bibliography **90**

List of Figures

1.1 Estimated growth of global data. 5

1.2 Radiation dosage for cancer treatment involves the solution of NNLS problems. (Left) We plot 12 of 35 slices of a C-shaped tumor. The required dosage to target the tumor is shown in Red. (Right) The achieved dosage using NNLS is shown. 7

2.1 We illustrate two cases where SVM gives us no speed up(left) and the other case where it does(right). The approximate hyperplane which is output by the SVM solver is shown in blue and the threshold hyperplane whether we accept a point as a non-support is given by the line in green. Parameters C, ϵ have to be fed in to the SVM solver. A priori, we have to trade-off high values of C and low values of ϵ with computation time. The best values of these parameters with respect to computation time and correctness of solution depend on the distribution of the data. This is illustrated in the figure. We plot two dimensional points on a plane in each figure. If the data is too clustered along the maximum-margin hyperplane(left) and the values of C, ϵ and δ are not set properly, we might end up declaring that all points are potentially supports after running the SVM solver. A better distribution of data would result in the case(right) where we prune the number of potential supports thereby reducing the size of the original problem. 16

2.2 Dose profiles of proton and antiproton beams 19

2.3 We plot the running time for all 6 approaches. Lines with filled markers correspond to NNLS solvers and hollow markers correspond to NNLS solvers initialized by the OCAS solver and our reduction technique. The x-axis is indexed by i, which controls the size of the input matrix, which is $300i \times 200i$. 21

2.4 We plot running times for FNNLS and PQN-NNLS and their corresponding OCAS initialized solvers. The problem size is fixed at 1200×800 and we vary the density from 0.1 to 1.0. These are the mean times for 10 runs at each density level. 22

2.5 Running times on the different datasets using the solvers FNNLS, PQN-NNLS and RAND-PQN-NNLS and their corresponding OCAS initialized counterparts can be obtained by multiplying the corresponding entry with the scaling factor. Comparison between running times across different NNLS solvers should be taken with a grain of salt for the stopping criterion for each solver is potentially different. However, stopping criterion between a solver and its OCAS initialized solver are the same and thus can be compared. 23

2.6 (Left) Mean running times for each problem size where the elements of the matrix $\boldsymbol{W}$ and vector $\boldsymbol{x}$ are drawn uniformly at random from $[0, 1]$. The running times for the solvers should be taken with a grain of salt because of the different stopping criterion used. (Right) Running times versus objective values for our (FCD) algorithm and the competing FNNLS and PLB algorithms on the phantom tumor dataset. 27

2.7 Running times versus objective values for FCD and PLB are shown for the real tumor dataset 28

2.8 The figure shows a paraboloid of the two dimensional objective function generated by a random construction of the Gram matrix satisfying the structure in (2.27). MUSIK and M^3 algorithms [104] were applied to this problem starting at $\boldsymbol{\alpha} = [1, 1]^T$. As expected, both algorithms arrive at the unique solution of the convex problem, however they follow different routes and the nonnegative kernel SVM takes fewer steps. 35

2.9 The figure shows the average ratio of the number of iterations for M^3 to the number of iterations for MUSIK taken to achieve given tolerance on the same problem (up is good). Computation is done at error bar points, the lines connecting them are for the visual guide only. The larger the problem size the smaller the number of iterations the algorithm needs compared to M^3, which can be up to 4 times less. Since the running time per iteration is comparable for both algorithms 4 times improvement in iterations means 4 times faster. Even for 0.01% distance from the solution our algorithm is more than two times faster on reasonable sized problems. 37

2.10 Differences in convergence on the UCI breast cancer dataset for MUSIK, M^3 and integrated MUSIK (iMUSIK) algorithms. 38

2.11 Rate of convergence of multiplicative updates for breast cancer dataset using RBF kernel with $\sigma = 3$. i is the iteration number, ϵ_t is the training error, ϵ_g is the test error. The support vectors have been rearranged for visualization into active and inactive. 45

2.12 The objective function (2.14) versus training iteration number (log scale) on the USPS handwritten digits dataset for the M^3 and the MUSIK algorithms (down is better). 47

2.13 Percentage of the misclassification versus the training iteration number (log scale) on the USPS handwritten digits dataset for the M^3 and the MUSIK algorithms (down is better). 48

2.14 Convergence performance on a dataset containing negative and positive values (shown on the left side) with polynomial kernel of degree 3. MUSIK algorithm with semiNMF updates from Section 2.6.3 is called sMUSIK in the legend. 49

3.1 (Left) Features learned from the ORL dataset (Scikit-learn package was used) with various matrix factorization methods such as principal component analysis (PCA), independent component analysis (ICA), and dictionary learning. The relative merit of the various matrix factorizations depends on both the signal domain and the target application of interest. (Right) Features learned under the sparse NMF formulation where roughly half the features were constrained to lie in the interval $[0.2, 0.4]$ and the rest are fixed to sparsity value 0.7. This illustrates the flexibility that the user has in fine tuning the feature sparsity based on prior domain knowledge. White pixels in this figure correspond to the zeros in the features. 54

3.2 Mean running times for Sparse-opt and the Projection-Hoyer are presented for random problems. The x-axis plots the dimension of the problem while the y-axis has the running time in seconds. Each of the subfigures corresponds to a single sparsity value in $\{0.2, 0.4, 0.6, 0.8\}$. Each datapoint corresponds to the mean running time averaged over 40 runs for random problems of the same fixed dimension. 64

3.3 Running times for SSNMF and SSNMF+Proj algorithms for the CBCL face dataset with rank set to 49 and sparsity values ranging from 0.2 to 0.9 65

3.4 Running times for SSNMF and NMFSC and TPC algorithms on the synthetic dataset where the sparsity values range from 0.2 to 0.8 and number of features is 5. Note that SSNMF and NMFSC are over an order of magnitude faster than TPC. 66

3.5 Convergence plots for the ORL dataset with sparsity from $[0.1, 0.8]$ for the NMFSC and SSNMF algorithms. Note that we are an order of magnitude faster, especially when the sparsity is higher. 66

3.6 Running times for SSNMF and NMFSC algorithms for the sMRI dataset with rank set to 40 and sparsity values of α from 0.1 to 0.8. Note that for higher sparsity values we converged to a lower reconstruction error and are also noticeably faster than the NMFSC algorithm. 67

3.7 Features from (Left) NMFSC algorithm and (Right) SSNMF algorithm (Right) using the ORL face dataset for sparsity values $0.5, 0.6, 0.75$. Note that SSNMF algorithm gives a parts-based representation similar to the one recovered by NMFSC. 68

4.1 The excess kurtosis of a source as a function of the relative size of the active region. A Gaussian has zero excess kurtosis. Here as in Example 2 of the original paper [28]. The four vertical lines at correspond to the relative sizes of the small box, the medium box, the large box, and a very large box corresponding to the maximal kurtosis case. Note that the medium and large box experiments have near zero excess kurtosis, *i.e., kurtosis value matching that of a Gaussian*. In addition, the pdfs of these sources are bimodal (see inset figures), ensuring that ICA algorithms designed for unimodal super-Gaussian distributions such as Infomax and FastICA with standard parameter settings, will likely fail. At the bottom of the figure are the ISI values (see Equation (1)) for the various algorithms at those four points (see Table 1 for full list). Also note the best separation performance of Infomax and FastICA for the maximum kurtosis case, which corresponds to almost the *lowest* level of sparsity. . 76

4.2 The distribution of sources and mixtures for $\lambda = 30\%(M_2)$. We plot (A-C) the distribution of sources, and (D) the contour plot of mixtures for the case of $\lambda = 30\%(M_2)$. Contrary to the claim made in Daubechies et al., the sources have in fact very peaky and heavy-tailed distributions and are not at all close to a Gaussian distribution. For comparison purposes we also present Gaussian distribution curves (blue, A-B). 82

4.3 Sparsity measures for three different coordinate system origins z_0. Sparsity as measured with respect to different coordinate system origins z_0, as a function of the relative size of the active region. Remark that for a relative size of zero, only background samples are present and, thus, the mean of the mixture model coincides with the mean of the background (and the two sparsity measures correspond at this point). An analogous observation can be made for a relative size of one, now with respect to the activity (signal samples). 83

List of Tables

2.1 Misclassification rates (%) on the breast cancer and sonar datasets after convergence of the M^3, MUSIK (M) and Kernel Adatron (KA) algorithms. Polynomial kernels of degree 4 and 6 and Gaussian kernels of σ 1 and 3 were used. 46

4.1 Source estimates for the four cases indicated in Figure 1 as in Example 2 of the original paper [8]. The algorithms behave as one would expect if they are selecting for independence. For the bimodal or Gaussian-like cases, ICA-EBM and Infomax (sub) do well, and for the unimodal or maximum kurtosis or low sparsity case Infomax-super, FastICA and ICA-EBM all do extremely well. Numbers in boldface indicate when separation was good. 86

4.2 Tabulated results for the so-called [28] ICA "promotional material". Both Infomax (super) and FastICA do successfully separate (zero indicates perfect separation) the super-Gaussian sources s_a and s_b . Note the excess kurtosis is more than 3 for both sources. Numbers in boldface indicate when separation was good. 87

Chapter 1

Introduction

Matrix factorization arises in a wide range of application domains and is useful for extracting the latent features in the dataset. Examples include recommender systems, brain data analysis, and document clustering. Informally, given a matrix $\boldsymbol{X}$, matrix factorization seeks to approximate it as a product of factors as follows:

$$\boldsymbol{X} \approx \boldsymbol{W}\boldsymbol{H}$$

Sometimes the term "approximate matrix factorization" is used since there may not exist an exact factorization. In this dissertation, we are interested in matrix factorizations which impose the following requirements:

- low rank — The rank of the matrices $\boldsymbol{W}, \boldsymbol{H}$ are set to be much smaller than their larger dimension. This requirement is particularly useful for learning a lower dimensionality representation as given by matrix $\boldsymbol{H}$. Also, in practice, the low-rank requirement works well in recommender systems where only a few criteria are assumed to give us a good prediction model for user preferences.

- nonnegativity — The elements of the matrix factors are positive (or zero). This is a natural constraint on solutions to machine learning problems, for instance when modeling chemical concentrations in solutions, pixel intensities in images, or radiation dosages for cancer treatment. The nonnegativity constraint on the representation leads to an easier identification of the features, denoted by columns of matrix $\boldsymbol{W}$, present in the corresponding data for the features can only combine additively.

- sparsity — The matrix factors have few non-zeros elements. Sparsity can be imposed on either or both of the matrix factors. For instance, when sparsity is imposed on the matrix $\boldsymbol{W}$ for music data, we can recover "parts" of music such as notes and chords. On the other hand, when it is imposed on the matrix factor $\boldsymbol{H}$, it leads to sparse representations. Examples include learning dictionaries or overcomplete representations for various types of datasets arising from natural images, music, and text.

- independence —This assumes that the column vectors of matrix $\boldsymbol{X}$ are random vectors and that they have been generated by linear combinations of the underlying independent sources whose realizations are represented by the columns of matrix $\boldsymbol{W}$. Examples include source separation in the context of speech, music and brain signals.

We expound on the above requirements with examples from matrix factorization problems.

Low rank: Singular value decomposition (SVD) is a widely used matrix factorization technique in many applications including problems in signal processing and machine learning. The SVD of a matrix $\boldsymbol{X}$ is the following:

$$\boldsymbol{X} = \boldsymbol{U}\boldsymbol{D}\boldsymbol{V}^{\top}$$

where the matrices $\boldsymbol{U}, \boldsymbol{V}$ are unitary and $\boldsymbol{D}$ is diagonal. Let us assume, without loss of generality, that datapoints correspond to the columns of matrix $\boldsymbol{X}$. An interesting application of the SVD is in finding low-rank approximations to a given data matrix. Since a lot of interesting domains produce high-dimensional data, such as natural images, world wide web (WWW), and functional magnetic resonance imaging (fMRI), we would like to find a lower-dimensional representation which captures most of the information and is also easier to analyze. The Eckart-Young theorem states that for a given rank k, SVD gives the best rank-k approximation to the target matrix $\boldsymbol{X}$ under the Frobenius norm where we select the columns of $\boldsymbol{U}, \boldsymbol{V}$ corresponding to the largest k singular values given by $\boldsymbol{D}$. The Frobenius norm of a matrix $\boldsymbol{A}$ is simply $\sqrt{\mathbf{Tr}(\boldsymbol{A}^\top \boldsymbol{A})}$. More formally, compute the follow matrix $\tilde{\boldsymbol{X}}$:

$$\tilde{\boldsymbol{X}} \approx \boldsymbol{U}_k \boldsymbol{D}_k \boldsymbol{V}_k^\top$$

where we have selected the top k singular values of $\boldsymbol{D}$, denote by $\boldsymbol{D}_k$, and their corresponding columns in matrices $\boldsymbol{U}, \boldsymbol{V}$. The matrix $\tilde{\boldsymbol{X}}$ is the best rank-k approximation to matrix $\boldsymbol{X}$ according to Eckart-Young theorem. Note that the SVD of a matrix can be computed in cubic time [42].

Nonnegativity: Factoring a matrix, all of whose entries are nonnegative as a product of two low-rank nonnegative factors is a fundamental algorithmic challenge and is called nonnegative matrix factorization (NMF). This has arisen naturally in diverse areas such as image analysis [65], micro-array data analysis [58], document clustering [114], chemometrics [63], information retrieval [45] and biology applications [13]. For further applications, see the references in the following papers [1, 25]. Interestingly, adding the nonnegativity constraint substantially increases the complexity of the problem. In sharp contrast to the SVD problem, it is no longer convex. In practice, the algorithms for NMF despite guaranteeing only local convergence tend to be surprisingly useful in machine learning applications.

Sparsity: Sparsity is another commonly imposed assumption that arises naturally from the principle of parsimony: the simplest explanation is preferred. Sparsity is also motivated by evidence of neuronal coding efficiency and sparse coding in the nervous system. Sparse representations can help avoid the problem of overfitting while also leading to solutions that are easier to interpret. Applications of sparse signal processing methods include dictionary learning [77], speech separation [119], and feature learning [48].

Independence: Independent component analysis (ICA) [79, 14, 5, 16] is a widely used signal processing approach that has been applied to areas including speech separation, communications, and functional magnetic resonance (fMRI) data analysis. Given a set of linearly mixed observations, recovering the underlying components is an ill-defined problem. However, the assumption of independence among the sources turns out to be surprisingly powerful and effective for a wide range of problems in various practical domains. The requirement is to minimize the statistical dependence among the sources, or in other words, the components of matrix $\boldsymbol{H}$. This assumption on the factorization has been been shown to be successful in the "cocktail party problem" where given a mixture of speech sources, we are able to separate out the individual speakers. This paradigm has been used to analyze data from other domains such as medical imaging [14]. In particular, for functional magnetic resonance imaging (fMRI) data, applying independence assumption on the spatial maps of the decomposition results in biologically relevant features present in the brain activations.

1.0.1 Big data

Data sets are growing in size partly due to the ubiquitous availability of mobile devices, brain scanners, cameras, microphones, and wireless sensor networks. It has been noted that as of 2012, around 2.5 quintillion bytes of data were created (Wikipedia 2012: Big data). Examples include the 13 petabytes of data produced by the four main detectors at the Large Hadron Collider in 2010 and the 140 terabytes of data the Sloan digital sky survey (SDSS) has amassed starting from 2000. Other areas where big data is becoming common include brain datasets using fMRI, MEG or EEG scanners, text documents from world wide web, and gene datasets in biology applications. For illustration, we show the growth of internet data in Figure 1.1

1.0.2 Distributed systems

With the increase in sizes of datasets, the computer hardware has also been keeping pace by following Moore's law. Moore's law states that the number of transistors in integrated circuits doubles approximately every 2 years. However, this does not necessarily translate

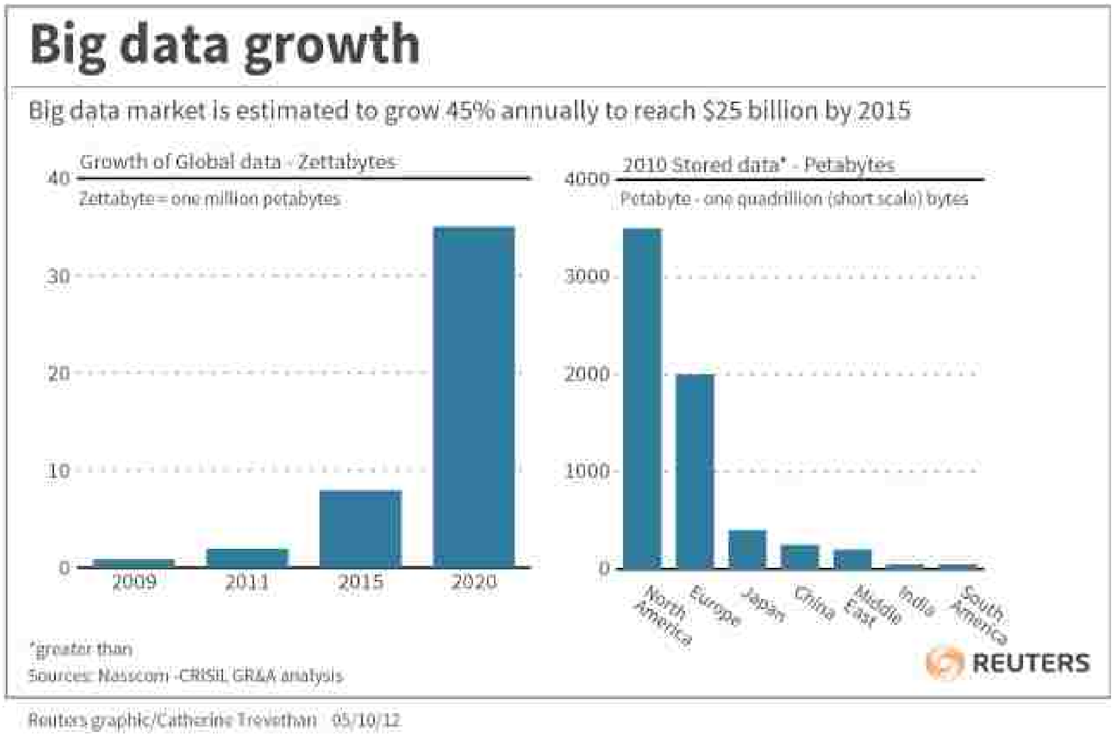


Figure 1.1: Estimated growth of global data.

to computation performance. In recent years, there has been a trend towards an increase in the number of computational cores available on a single chip. This necessitates the use of parallel programming to take advantage of these systems. Also, in order to reduce wall clock time for processing large datasets there has been an increased need for distributed systems. MapReduce has been introduced which can rapidly process large amounts of data in parallel on distributed cluster nodes. One such practical system is the open source Hadoop library written in Java [112]. To leverage these systems, we need to design efficient algorithms, with proven convergence guarantees, by leveraging the distributed architecture of the computational nodes. Also, the computational nodes can further leverage parallelization based on their multi-core architecture. One such project based on Hadoop is the Apache Mahout which attempts to build a scalable machine learning library.

1.1 Background

Nonnegative Quadratic programming (NQP) involves optimizing a quadratic objective function subject to nonnegative constraints. It is defined as follows:

$$\min_{x} \frac{1}{2}\boldsymbol{x}^{\top}\mathbf{A}\boldsymbol{x} + \boldsymbol{b}^{\top}\boldsymbol{x}$$
$$\boldsymbol{x} \geq \mathbf{0} \tag{1.1}$$

NQP encompasses a wide umbrella of important problems such as LASSO, Support Vector Machines (SVM), Nonnegative Least Squares(NNLS) and Nonnegative Matrix Factorization (NMF).

1.1.1 Least Absolute Shrinkage and Selection Operator (LASSO)

Let $\boldsymbol{W} \in R^{m\times n}$ be a matrix and $\boldsymbol{y} \in R^{m}$ a column vector. LASSO is formulated as follows:

$$\min_{\boldsymbol{\beta}} \frac{1}{2}\|\boldsymbol{y} - \boldsymbol{W}\boldsymbol{\beta}\|_2^2 + \lambda\|\boldsymbol{\beta}\|_1 \tag{1.2}$$

The LASSO formulation arises in linear regression with a sparsity regularization term to avoid the problem of over-fitting. It involves an L_1 term which is not differentiable making the development of efficient algorithms non-trivial. Quite a few algorithms have been developed over the years to solve this problem [109, 83, 59, 68, 9].

1.1.2 Support Vector Machine (SVM)

Let the set of labeled examples be $\{(\boldsymbol{x}_i, y_i)\}_{i=1}^{n}$, with binary class labels $y_i = \pm 1$ corresponding to two classes. The dual quadratic optimization problem for SVM [102] is given by minimizing the following loss function:

$$\min_{\boldsymbol{\alpha}} \frac{1}{2}\sum_{i,j=1}^{n} \alpha_i\alpha_j y_i y_j k(\boldsymbol{x}_i, \boldsymbol{x}_j) - \sum_{i=1}^{n}\alpha_i \tag{1.3}$$
$$\text{subject to } \alpha_i \geq 0, i \in \{1, \ldots, n\},$$

where $k(\boldsymbol{x}_i, \boldsymbol{x}_j)$ is a kernel that computes the inner product $\Phi(\boldsymbol{x}_i)^T\Phi(\boldsymbol{x}_j)$ in the space Φ by performing all operations only in the original data space on x_i and x_j, thus defining a Hilbert space Φ.

Recently, the cost of training of kernel SVM's has shifted the focus of the SVM community back to linear SVM for large scale applications. This has lead to the formulation of very efficient linear SVM solvers which converge to a ϵ precision solution in linear (in the number of training points) time as seen in the literature [40, 50].

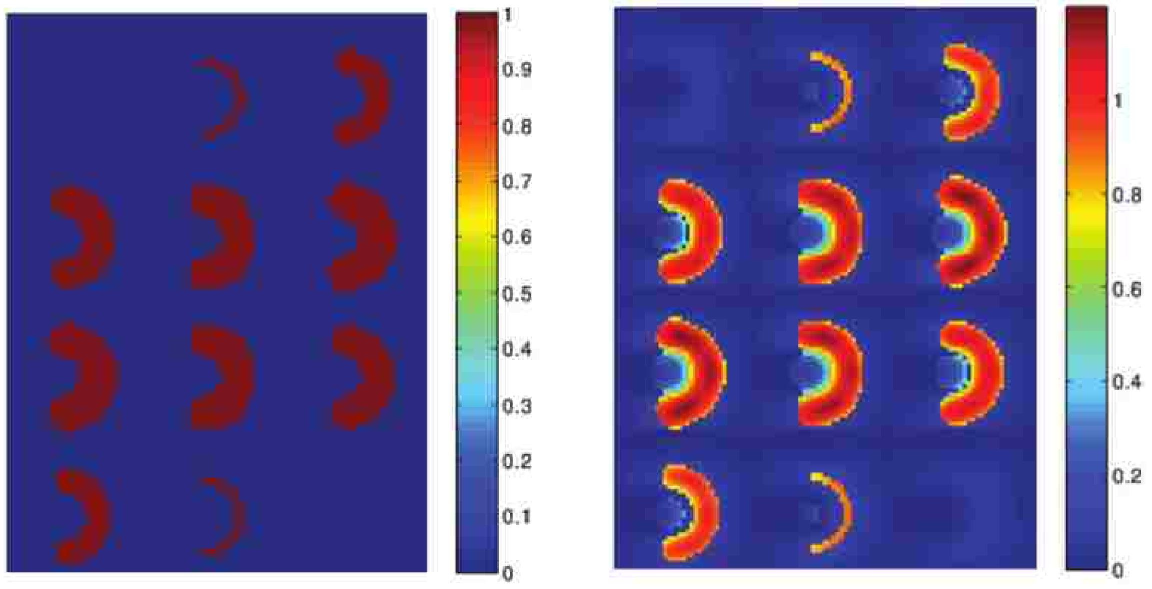


Figure 1.2: Radiation dosage for cancer treatment involves the solution of NNLS problems. (Left) We plot 12 of 35 slices of a C-shaped tumor. The required dosage to target the tumor is shown in Red. (Right) The achieved dosage using NNLS is shown.

1.1.3 Nonnegative Least Squares (NNLS)

Let $\boldsymbol{W} \in R^{m\times n}$ be a matrix and $\boldsymbol{y} \in R^m$ a column vector. The nonnegative least squares problem (NNLS) is to find a column vector $\boldsymbol{x} \in R^n$ which solves the following problem:

$$\min_{\boldsymbol{x}\geq\boldsymbol{0}} \frac{1}{2}\|\boldsymbol{y} - \boldsymbol{W}\boldsymbol{x}\|_2^2 \tag{1.4}$$

The NNLS problem and an algorithm to solve it was introduced in the early 70's [62]. NNLS problems frequently arise in practice and there are quite a few algorithms to solve them [62, 57, 11, 6]. Note that the nonnegativity constraint is natural in real problems, for instance, when we are modeling chemical concentrations, brain activations, or color intensities. Real world applications include target detection at subpixel level in remote sensing images [18], and resolving tags into genes in the SAGE datasets [118]. We illustrate the NNLS formulation on a radiation therapy problem in Figure 1.2.

1.1.4 Nonnegative Matrix Factorization (NMF)

Matrix factorization arises in quite a few applications where the assumption is that the data is generated from the linear combination of underlying features. Additionally, if we constrain the data, features and the representation to be nonnegative we arrive at the Nonnegative Matrix Factorization (NMF) problem. NMF has seen increasing applications in the last decade for nonnegativity is a natural constraint in a wide range of applications like gene analysis, document clustering and face recognition. We will consider the following version of the NMF problem, which measures the reconstruction error using the Frobenius norm [66].

$$\begin{aligned} &\min_{\mathbf{W},\mathbf{H}} \frac{1}{2}\|\mathbf{X}-\mathbf{WH}\|_F^2 \\ &\qquad \text{s.t. } \mathbf{W} \geq \mathbf{0}, \mathbf{H} \geq \mathbf{0} \\ &\qquad\qquad \|\boldsymbol{W}_j\|_2 = 1, \forall j \in \{1, \cdots, r\} \end{aligned} \tag{1.5}$$

where $\geq$ is element-wise. Note that we use subscript to denote column elements and superscript to denote row elements.

Besides the Frobenius norm, other measures have been proposed for the NMF problem [73, 38, 30]. Also, extensions to tensors have been studied in the literature [115, 24]. Convolutive formulations of NMF have been successfully applied to audio datasets [90, 105]. Bayesian treatment of the NMF problem have recently become popular [101, 17].

1.2 Contributions of this dissertation

Nonnegative matrix factorization (NMF) and support vector machines have a structural connection. We explored this connection to propose novel algorithms for both the NMF and SVM problems. In particular, we developed multiplicative updates for the SVM problem akin to the NMF problem. Also, we showed an explicit reduction from totally nonnegative least squares to a single class support vector machine. Treatment planning systems for radiation therapy can be modeled as nonnegative least squares (NNLS) problems. We exploited the reduction and the existence of fast SVM solvers to efficiently solve the NNLS problem. This enabled us to reduce the planning time for cancer treatment by an order of magnitude.

NMF is in general an ill-defined problem and additional constraints such as sparsity have been shown to result in more domain revelant features [41]. Many sparse formulations for NMF have been proposed. We consider one such model proposed by Hoyer. It has the benefit that it is easy to use. Hower, algorithms for it are slow. Therefore, we proposed a novel algorithm based on block coordinate methods. We showed that the algorithm is fast on real-world datasets and can give us an order of magnitude improvement.

Independent component analysis (ICA) [5, 16, 79, 14] has enjoyed success on a wide range of domains like speech separation, fMRI brain analysis, and MEG/EEG source extraction. Given signal mixtures, recovering the underlying components is an ill-defined problem. However, the assumption of independence among the sources turns out to be a surprisingly powerful and effective for a wide range of problem domains in practice. In particular, we tackle the question of understanding the relationship between independence and sparsity in the context of fMRI data. Recent work by Daubechies [28] suggested that sparsity is the driving force behind the success of ICA algorithms and not independence as was previously thought. We reexamined the experimental work and concluded that the evidence does not support the conclusion reached.

Chapter 2

Nonnegativity — Structural connections

Let us revisit the three nonnegative problems comprised of support vector machines, nonnegative least squares and nonnegative matrix factorization.

Support Vector Machines (SVM): SVMs are now routinely used for many classification problems in machine learning [102] due to their ease of use and ability to generalize. In the basic case, the input data, corresponding to two groups, is mapped into a higher dimensional space, where a maximum-margin hyperplane is computed to separate them. The "kernel trick" is used to ensure that the mapping into higher dimensional space is never explicitly calculated. This can be formulated as a non-negative quadratic programming (NQP) problem and there are efficient algorithms to solve it [94].

Given labeled training examples $\{(\boldsymbol{x}_i, y_i)\}_{i=1}^n$, with binary class labels $y_i = \pm 1$ corresponding to two classes. The primal formulation of the binary class linear SVM is:

$$\min_{\boldsymbol{w}} P(\boldsymbol{w}) := \frac{1}{2}\|\boldsymbol{w}\|_2^2 + C\sum_{i=1}^{n} \max\{0, 1 - y_i(\langle \boldsymbol{w}, \boldsymbol{x}_i \rangle)\} \tag{2.1}$$

where C is a regularization constant. The corresponding dual formulation for linear SVM [102] is given by minimizing the following loss function:

$$\min_{\boldsymbol{\alpha}} \frac{1}{2}\sum_{i,j=1}^{n} \alpha_i\alpha_j y_i y_j k(\boldsymbol{x}_i, \boldsymbol{x}_j) - \sum_{i=1}^{n} \alpha_i \tag{2.2}$$

$$\text{subject to } \; 0 \le \alpha_i \le C, \; i \in \{1, \dots, n\},$$

where $k(\boldsymbol{x}_i, \boldsymbol{x}_j)$ is a kernel that computes the inner product $\Phi(\boldsymbol{x}_i)^T\Phi(\boldsymbol{x}_j)$ in the space Φ by performing all operations only in the original data space on x_i and x_j, thus defining a Hilbert space Φ.

An SVM can be trained using variants of the gradient descent method applied to the NQP. Although these methods can be quite efficient [27], their drawback is that they require a manually-tuned and problem specific learning rate. Subset selection methods are an alternative approach to solving the SVM NQP problem [94]. At a high level, they work by splitting the arguments of the quadratic function at each iteration into two sets: a fixed set, where the arguments are held constant, and a working set of the variables being optimized in the current iteration. These methods, though efficient in space and time, still require a heuristic to exchange arguments between the working and the fixed sets.

An alternative algorithm for solving the general NQP problem has been applied to SVMs [104]. The algorithm, called M^3, uses multiplicative updates to iteratively converge to the solution. It does not require any heuristics, such as setting the learning rate or choosing how to split the argument set. Multiplicative updates in the M^3 algorithm are formulated for the general NQP problem and then applied to SVM as a special case. It was also demonstrated [103] that M^3 can solve soft-margin SVMs and the sum constraint can be accounted for. However, accounting for the sum constraint requires choosing a parameter, which defeats the original intention of creating a parameter free SVM algorithm.

Nonnegative Matrix Factorization (NMF): We present a brief introduction to NMF mechanics with the notation that is standard in NMF literature. NMF is a tool to split a given nonnegative data matrix into a product of two nonnegative matrix factors [67]. The constraint of nonnegativity (all elements are ≥ 0) usually results in a parts-based representation and is different from other factorization techniques which result in more holistic representations (e.g. PCA and VQ).

Given a nonnegative $m \times n$ matrix $\boldsymbol{X}$, we want to represent it with a product of two nonnegative matrices $\boldsymbol{W}, \boldsymbol{H}$ of sizes $m \times r$ and $r \times n$ respectively:

$$\boldsymbol{X} \approx \boldsymbol{W}\boldsymbol{H}. \tag{2.3}$$

Lee and Seung [67] describe two simple multiplicative updates for $\boldsymbol{W}$ and $\boldsymbol{H}$ which work well in practice. These correspond to two different cost functions representing the quality of approximation. Here, we use the Frobenius norm for the cost function. The cost function and the corresponding multiplicative updates are:

$$E = \frac{1}{2}\|\boldsymbol{X} - \boldsymbol{W}\boldsymbol{H}\|_F^2 \tag{2.4}$$

$$\boldsymbol{W} = \boldsymbol{W} \odot \frac{\boldsymbol{X}\boldsymbol{H}^T}{\boldsymbol{W}\boldsymbol{H}\boldsymbol{H}^T}, \tag{2.5}$$

$$\boldsymbol{H} = \boldsymbol{H} \odot \frac{\boldsymbol{W}^T\boldsymbol{X}}{\boldsymbol{W}^T\boldsymbol{W}\boldsymbol{H}}, \tag{2.6}$$

where $\|.\|_F$ denotes the Frobenius norm and the operator $\odot$ represents element-wise multiplication. Division is also element-wise. It should be noted that the cost function to be minimized is convex in either $\boldsymbol{W}$ or $\boldsymbol{H}$ but not in both [67]. In [67] it is proved that when the algorithm iterates using the updates (2.5) and (2.6), $\boldsymbol{W}$ and $\boldsymbol{H}$ monotonically decrease the cost function.

The slightly mysterious form for the above updates can be understood as described in [67]. A gradient descent (additive) update for $\boldsymbol{H}$ is given by:

$$\boldsymbol{H} = \boldsymbol{H} + \boldsymbol{\eta} \odot (\boldsymbol{W}^T\boldsymbol{X} - \boldsymbol{W}^T\boldsymbol{W}\boldsymbol{H}) \tag{2.7}$$

If the learning rate given by the matrix elements of $\boldsymbol{\eta}$ be all set to some small positive number then this is the conventional gradient descent. However, setting the learning rate matrix as follows:

$$\boldsymbol{\eta} = \frac{\boldsymbol{H}}{\boldsymbol{W}^T\boldsymbol{W}\boldsymbol{H}} \tag{2.8}$$

gives us the NMF updates. We note the multiplicative factors for the updates correspond to the negative component of the derivative divided element-wise by the positive component of the derivative respectively.

NMF problem was extended by Ding et al. [31] to semi-NMF, where data matrix $\boldsymbol{X}$ and one of the factors $\boldsymbol{W}$ were allowed to have real elements. Ding et al. [31] derive and provide convergence guarantees for the multiplicative updates:

$$\boldsymbol{W} = \boldsymbol{X}\boldsymbol{H}^T(\boldsymbol{H}\boldsymbol{H}^T)^{-1} \tag{2.9}$$

$$\boldsymbol{H} = \boldsymbol{H} \odot \sqrt{\frac{[\boldsymbol{W}^T\boldsymbol{X}]^+ + [\boldsymbol{W}^T\boldsymbol{W}]^-\boldsymbol{H}}{[\boldsymbol{W}^T\boldsymbol{W}]^+\boldsymbol{H} + [\boldsymbol{W}^T\boldsymbol{X}]^-}} \tag{2.10}$$

Nonnegative Least Squares (NNLS): Let $\boldsymbol{W} \in R^{m\times n}$ be a matrix and $\boldsymbol{y} \in R^m$ a column vector. The nonnegative least squares problem (NNLS) is to find a column vector $\boldsymbol{x} \in R^n$ which solves the following problem:

$$\min_{\boldsymbol{x}\geq\boldsymbol{0}} \frac{1}{2}\|\boldsymbol{y} - \boldsymbol{W}\boldsymbol{x}\|_2^2 \tag{2.11}$$

If we additionally constrain all the elements of matrix $\boldsymbol{W}$ and vector $\boldsymbol{x}$ to be nonnegative, we get the totally nonnegative least squares (TNNLS) problem [81]. TNNLS has been applied to compressive sensing by OGrady and Rickard [89] who showed that nonnegativity is enough to recover a sufficiently sparse signal. Bruckstein et al. [12] have explored the connection between uniqueness of nonnegative sparse solutions of underdetermined systems of equations while Donoho and Tanner [33] explored thresholds for recovery of sparse solutions via L_1 minimization. But why is there this connection between nonnegative entries and sparsity?

Many algorithms have been developed over the years to solve the NNLS problem. A brief history of these can be found in Kim et al. [57]. For instance, the NNLS algorithm of Lawson and Hanson [62] was modified by Bro and De Jong [11] and was called FAST-NNLS (FNNLS). However, FNNLS requires the computation of matrix-matrix product (of the input matrix) and can be expensive for large-scale problems. This was ameliorated in the case of multiple right hand sides by Benthem and Keenan [6] and was called FCNNLS. We do not consider FCNNLS in this work for we are solving NNLS problems with a single right hand side. Recently, advances in fast randomized projections have lead to the development of a randomized algorithm for NNLS which involves first employing a randomized Hadamard transform to construct a smaller NNLS problem. This is then solved by a standard NNLS solver [8].

Background: In [44], an isomorphism was established between sparse separation and ϵ-SVM regression and used it to kernelize sparse separation. Similarly, a connection between LASSO and SVM's was established [70] and further exploited for the kernel version of LASSO. Furthermore, the kernel adatron (KA) algorithm for solving SVM [27] resurfaced in solving the NNLS problem [39]. Coordinate descent methods have been applied to solve the SVM and NMF problems [50, 49]. Recently, an equivalence between L2-SVM and LASSO has been shown [53]. That is given an L2-SVM problem, we can construct a LASSO problem with an equivalent solution and vice-versa. Further connections between the various NQP problems can be found in [53].

Outline: For the Totally Nonnegative Least Squares (TNNLS) problem which is a special case of NNLS, we show that it can be reduced to a single-class SVM problem. This enables us to tie the number of supports (sparsity) of the single-class SVM problem to the nonnegativity constraint of the NNLS problem [96]. We exploit this reduction to propose two algorithms based on a primal and a dual SVM solver. Efficiency of the proposed framework is demonstrated on real-world datasets arising from problems in radiation therapy treatment.

Also, we show a structural connection between NMF and SVM enabling us to propose multiplicative updates for SVM [97]. In this chapter, we reformulate the dual SVM problem as a matrix factorization problem and demonstrate a connection to the non-negative matrix factorization (NMF) algorithm [67]. NMF employs multiplicative updates and is very successful in practice due to its independence from the learning rate parameter, low computational complexity, and ease of implementation.

2.1 Our Reduction from TNNLS to SVM

We describe a general framework for reducing TNNLS to SVM. In particular, we show that the TNNLS problem can be reduced to solving a hard-margin single class dual SVM problem. Let $\boldsymbol{D}$ denote the diagonal matrix whose diagonal elements are given by the vector $\frac{1}{\boldsymbol{W}^T\boldsymbol{x}}$. Also, let $\boldsymbol{h} = \boldsymbol{D}\boldsymbol{z}$. Then,

$$\begin{aligned}
G(\boldsymbol{z}) &= \frac{1}{2}\|\boldsymbol{x} - \boldsymbol{W}\boldsymbol{D}\boldsymbol{z}\|_2^2 \\
&= \frac{1}{2}\boldsymbol{z}^T(\boldsymbol{W}\boldsymbol{D})^T(\boldsymbol{W}\boldsymbol{D})\boldsymbol{z} - \boldsymbol{x}^T\boldsymbol{W}\boldsymbol{D}\boldsymbol{z} + \frac{1}{2}\boldsymbol{x}^T\boldsymbol{x} \\
&= \frac{1}{2}\boldsymbol{z}^T(\boldsymbol{W}\boldsymbol{D})^T(\boldsymbol{W}\boldsymbol{D})\boldsymbol{z} - \boldsymbol{1}^T\boldsymbol{z} + \frac{1}{2}\boldsymbol{x}^T\boldsymbol{x}
\end{aligned}$$

Ignoring the $\frac{1}{2}\boldsymbol{x}^T\boldsymbol{x}$, which does not change the location of the minimum, we see that it is an instance of the SVM objective in equation 2.2, with $\boldsymbol{v}$ corresponding to $\boldsymbol{z}$ and $\boldsymbol{s}_i$ corresponding to $\boldsymbol{W}_i\boldsymbol{D}_{ii}$. We have a single class maximum margin classifier passing through origin where the datapoints given by $\{\boldsymbol{W}_i D_{ii}\}_i^n$ lie in the positive orthant. Since, the primal version of TNNLS corresponds to the dual of a single class SVM, the dual of TNNLS corresponds to the primal of the single class SVM. Geometrically, this corresponds to finding a maximum margin hyperplane which gives us the set of supports. Or in other words, it gives us the zero elements of the vector $\boldsymbol{z}$ or equivalently $\boldsymbol{h}$. In practice, we find a superset of supports because we use a SVM solver to find an approximate maximum margin hyperplane.

2.1.1 Implementation issues

Algorithms for solving the SVM problem can be split into primal or dual depending on the version of the problem they solve. In this paper, we use a primal SVM solver to find an approximate maximum-margin hyperplane. This gives us a subset of nonsupport vectors. We solve for the remaining entries by invoking an exact NNLS solver. Note that, if we had a dual SVM solver, we could directly use it to solve the TNNLS problem. However, since we are using solvers whose performance scales as $O(\log(1/\epsilon))$, it might be preferable to get an approximate solution for say $\epsilon = 0.001$ and get the exact solution by using some other exact solver. This depends on how much we care about the accuracy of the solution and is application dependent.

Recently, a lot of fast approximate solvers have been proposed to solve the linear SVM problem. OCAS by [40] is based on a cutting plane algorithm and is one of the state-of-the art solvers. It very quickly approximates a maximum-margin hyperplane and its running time is linear in the size of the input samples (see [40] or [50] for details). For getting an approximate maximum-margin hyperplane, we use the OCAS solver of [40]. However, since we are using an approximate SVM solver, we find a subset of the zeros and have to solve for the smaller problem by using an exact NNLS solver.

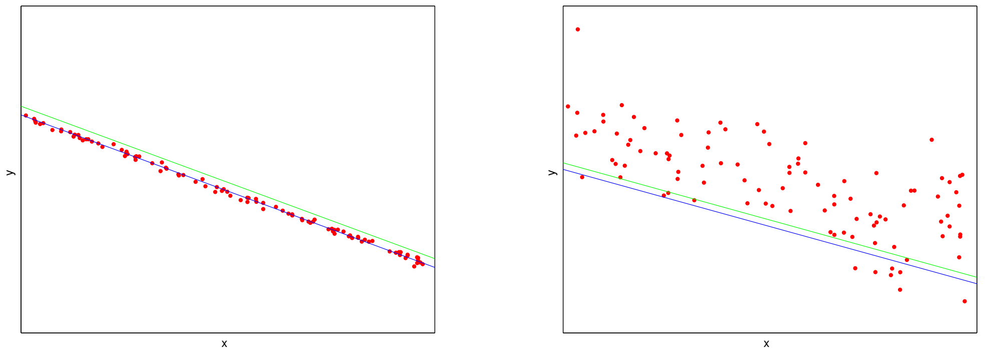


Figure 2.1: We illustrate two cases where SVM gives us no speed up(left) and the other case where it does(right). The approximate hyperplane which is output by the SVM solver is shown in blue and the threshold hyperplane whether we accept a point as a non-support is given by the line in green. Parameters C, ϵ have to be fed in to the SVM solver. A priori, we have to trade-off high values of C and low values of ϵ with computation time. The best values of these parameters with respect to computation time and correctness of solution depend on the distribution of the data. This is illustrated in the figure. We plot two dimensional points on a plane in each figure. If the data is too clustered along the maximum-margin hyperplane(left) and the values of C, ϵ and δ are not set properly, we might end up declaring that all points are potentially supports after running the SVM solver. A better distribution of data would result in the case(right) where we prune the number of potential supports thereby reducing the size of the original problem.

2.2 Primal SVM solver

If we use a primal hard-margin SVM solver and find an approximate hyperplane, say $\boldsymbol{w}$ then $\boldsymbol{w}$ satisfies the condition: $1 - P(\boldsymbol{w}^*)/P(\boldsymbol{w}) \leq \epsilon$ where $\boldsymbol{w}^*$ denotes the optimal hyperplane and ϵ is tolerance to which we solve the problem. The way we set a coefficient to zero is if it's corresponding input s_i satisfies the condition $\langle \boldsymbol{w}^*, \boldsymbol{s}_i \rangle > 1$. Since we don't have the vector $\boldsymbol{w}^*$ and have access to only $\boldsymbol{w}, \epsilon$, we instead use the following test function: $\langle \boldsymbol{w}, \boldsymbol{s}_i \rangle > 1 + \delta$ where δ is a function of ϵ and the data. By using the primal SVM solver, we are in fact solving the dual version of the TNNLS problem. Informally, this corresponds to finding the zeros of the solution vector. Once, we have all the zeros of the solution vector, the rest can be found by any least squares solver. However, if we end up with a subset of the zeros, as we do in this paper, we need to solve for the rest of the solution vector by using an NNLS solver. In practice, we find that ϵ set in the range

$[10^{-4}, 10^{-6}]$ is a good compromise between speed and accuracy as seen in the experiment section 2.3. We don't actually give a formula for computing δ but found 10ϵ to be a good heuristic in practice.

In this paper, we use a soft-margin SVM solver. This results in the issue of selecting the soft-margin parameter C. If we had used a hard margin SVM solver, this would not an issue as $C = \infty$. However, in the case of soft-margin SVM we need to set it. Ideally, we want C to be as large as possible. We found that C in the range $[10, 100]$ is a good choice for a wide range of problem sizes as shown in experiment section 2.3.

We illustrate the issue of setting the parameters of C, ϵ and δ in Figure 2.1. If the data is not close to the maximum separating hyperplane then we need not solve the SVM to high accuracy and can get by with a rough solution. However, if the data is highly clustered as in the case of Figure 2.1 then, we need to set ϵ high and this requires higher computation time for the SVM solver. It might be that the reduction can take more time than if we solved it directly. At the moment, we don't have a nice way to resolve this question.

If we don't treat the SVM solver as a black box as we do now, we can do something smarter by checking "progress" at each iteration and can come up with heuristics as to when to switch to exact solver.

2.2.1 Bounds

The soft margin parameter C can be set in a precise manner if we solve the primal SVM problem exactly. Notice that the soft-margin SVM formulation (2.1) has a dual formulation (2.21) where the parameter C only appears as an upper-bound constraint. Let $L = \max(\frac{\boldsymbol{W}^T\boldsymbol{x}}{diag(\boldsymbol{W}^T\boldsymbol{W})})$ where the function *diag* outputs the diagonal of a given input matrix. Setting C to be any value greater than L would make the single-class soft-margin SVM problem with nonnegative inputs equivalent to its hard-margin formulation. However, since we solve the soft-margin to only ϵ precision, it becomes tricky as to what the optimal value for parameter C should be.

We have the following bound on the primal solver objective:

$$(1-\epsilon)P(\boldsymbol{w}) \leq P(\boldsymbol{w}^*) \leq P(\boldsymbol{w})$$

We can find a nonnegative vector $\boldsymbol{f}$ from the above inequality such that if $\langle \boldsymbol{w} - \boldsymbol{f}, \boldsymbol{s}_i \rangle > 1$ implies that $\langle \boldsymbol{w}^*, \boldsymbol{s}_i \rangle > 1$ for all inputs s_i. In practice, we found this approach for estimating the supports conservative.

2.3 Experiments

In this section, we are going to present the results of applying various NNLS algorithms to different datasets. For the PQN-NNLS solver, we use the code supplied by the authors [57]. The default settings for the solver were used. The randomized NNLS solver code is based on the algorithms in [4] and [57].

We ran all the experiments on a machine with 2.2Ghz cpu power and 32GB of physical memory with 8 cores. The number of threads was set to 1 to ensure that we are using a single core.

Besides random data sets, we also applied our TNNLS solver to data sets that arise from Gamma Knife radiosurgery [21] and particle radiation therapy [61].

Gamma knife radiosurgergy has been a well-known treatment modality for many brain tumors and functional disorders. It uese γ-rays emitted from radioactive ^{60}Co sources to eradicate tumors and eliminate them. These sources are placed in a hemispherical, circular or linear array and their γ-ray beams are focused on a single point, creating a spherical high dose volume. Generally speaking, the goal of Gamma Knife is to use these spherical high dose volume to create a radiation dose distribution where the high dose regions are conformed to the targeted tumors. The problem is a typical TNNLS problem, where each column of the matrix W is a spherical high dose volume, and vector x is the ideal dose distribution, and the vector h is the weighting (i.e., “beam-on” time) of each high dose volume. Natually, everything is nonnegative in the problem. Our experimental results on Gamma Knife radiosurgery are in Section 2.3.2.

Another type of medical problem that we experimented with is the particle radiation therapy, where charged particles such as protons and carbon ions are used to irradiate tumors. This problem is similar to Gamma Knife radiosurgery, because the goal is to use particles beams to cover a targeted tumor to achieve an ideal dose distribution. Figure 2.2 shows the profiles of proton and carbon ions in comparison to X-rays. As can be seen, the dose profiles of protons and carbon ions display a distinct localized peak, called a Bragg Peak. The Bragg Peak makes TNNLS modeling particularly suitable for planning particle therapy, where the goal is to find the weighting for each particle beam to created a distribution as close to the target distribution as possible. Our experimental results on carbon therapy are shown in Section 2.3.3.

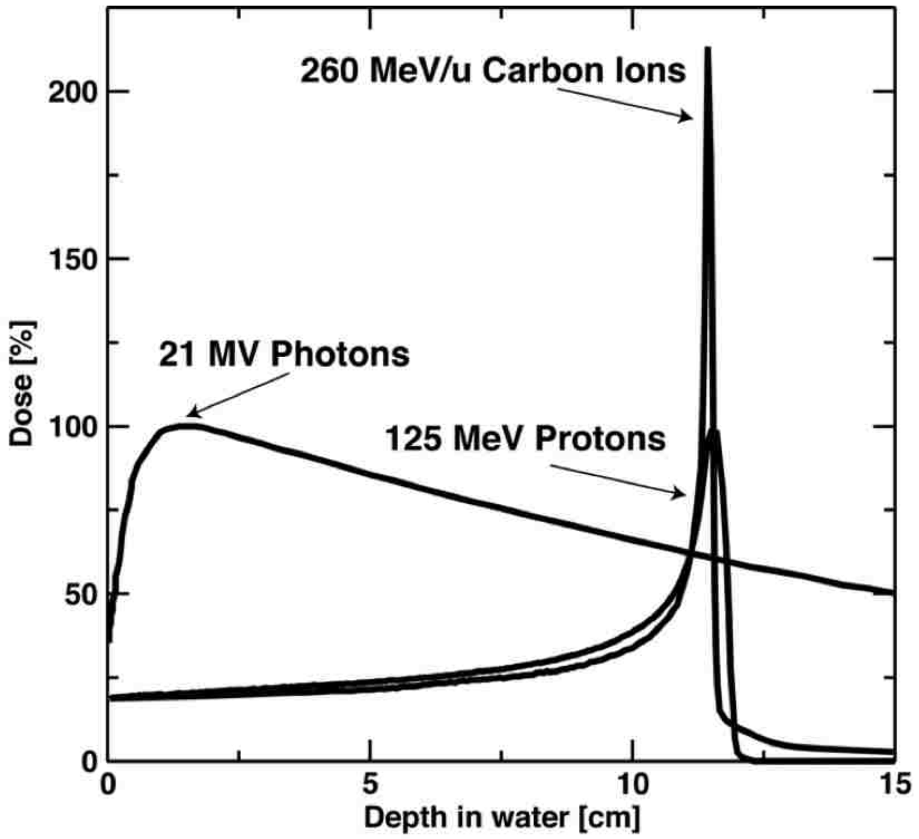


Figure 2.2: Dose profiles of proton and antiproton beams

2.3.1 Random problems

We evaluate the performance of the algorithm on randomly generated problems by varying size, aspect ratio and sparsity of the input data.

Size

We have applied the algorithm to a suite of 200 randomly generated problems of varying size. This is done by sampling the entries of $\boldsymbol{W}, \boldsymbol{x}$ uniformly from $[0, 1]$. We set the size of the matrix $\boldsymbol{W}$ to be $300i \times 200i$, where i ranges from 1 to 40. And, for each size, we create 5 randomly generated problems.

First, we applied the NNLS algorithms FNNLS [11], RAND-PQN-NNLS [8] and PQN-NNLS [57] to solve this set of problems. The running times for these algorithms are shown in Figure 2.3 with solid markers.

Next, we applied our reduction to obtain the SVM problem, and used the OCAS algorithm to find most of the zeros of the solution vector, and finally solved for the nonzeros by giving it to each of the 3 NNLS solvers mentioned above. The objective value obtained for the exact solver and the corresponding OCAS initialized solver match up to 6 significant digits. The parameters in OCAS initialized solvers are set as $(C, \epsilon) = (10, 10^{-4})$ for all problems. We plot the running time for the 3 OCAS initialized algorithms in Figure 2.3 using hollow markers. For each size, we plot the mean of running times. Note that these running times are for the entire procedure of reduction, running the SVM solver, and exact solution of the smaller NNLS problem. Except for the smallest cases, the OCAS initialized solver beats the corresponding exact solver. For larger matrices, the figure shows at least an order of magnitude improvement in the running times.

Aspect ratio

We did a similar analysis on a suite of 30 randomly generated problems of varying aspect ratio. The number of rows is set to 12000 and the number of columns is $1200 \times i$ where i

goes from 1 to 6. For each i, we generate 5 random instances. The parameters in OCAS initialized solvers are set as $(C, \epsilon) = (10, 10^{-4})$. The running times for all the six solvers are shown in Figure 2.3.

Sparsity

We also compared the running times for two of the solvers by varying the sparsity of the input matrix $\boldsymbol{W}$. We choose a fixed sized problem of size 1200×800 and varied the sparsity from 0.1 to 1.0 in increments of 0.1. Note that 1.0 corresponds to all the elements being nonzero. The parameters in OCAS initialized solvers are set as $(C, \epsilon) = (100, 10^{-6})$. The plots of running times for the two solvers FNNLS and PQN-NNLS and their OCAS initialized solvers are shown in Figure 2.3.1.

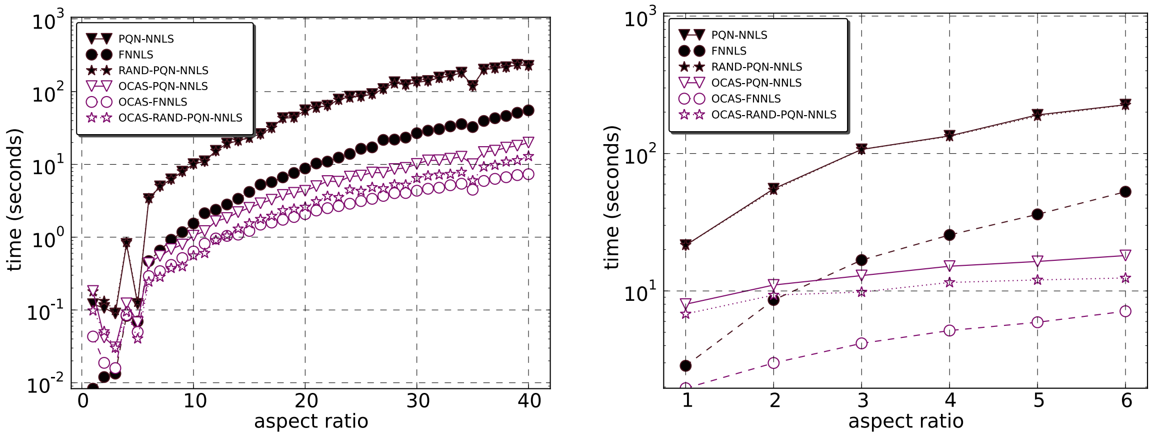


Figure 2.3: We plot the running time for all 6 approaches. Lines with filled markers correspond to NNLS solvers and hollow markers correspond to NNLS solvers initialized by the OCAS solver and our reduction technique. The x-axis is indexed by i, which controls the size of the input matrix, which is $300i \times 200i$.

2.3.2 Phantom tumor dataset

We also applied our TNNLS solver to a data set from a phantom commonly used for benchmarking radiosurgery treatment planning systems [76]. The phantom contains a C-shaped tumor surrounding a spherical critical structure and simulates a spine tumor case.

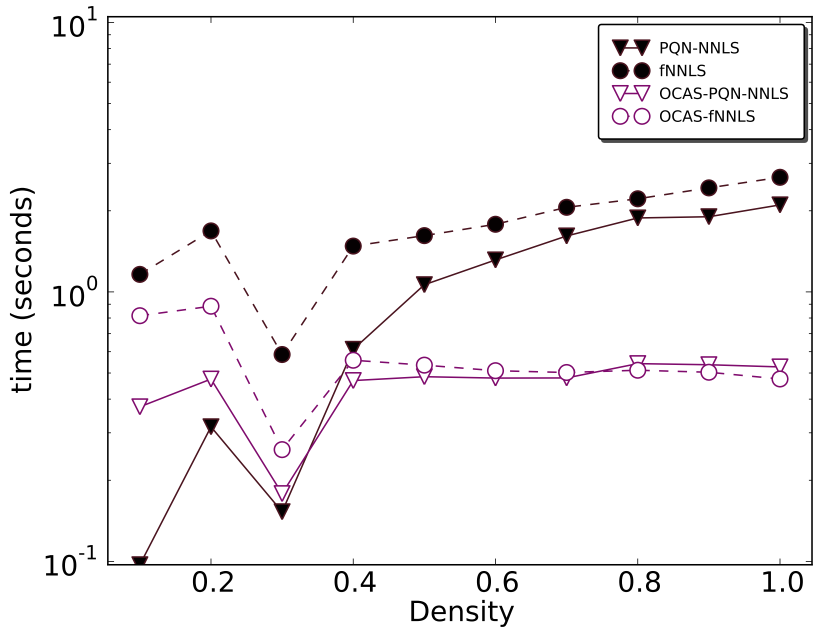


Figure 2.4: We plot running times for FNNLS and PQN-NNLS and their corresponding OCAS initialized solvers. The problem size is fixed at 1200×800 and we vary the density from 0.1 to 1.0. These are the mean times for 10 runs at each density level.

In this data set, the size of the input matrix W is 42875×20268 and the input vector x is of size 42875. Clinically, each column of the matrix W represents the radiation energy distribution deposited by a "shot" of radiation in Gamma Knife radiosurgery. The matrix x represents the ideal radiation energy deposition as prescribed by the physician. The sought variable h denotes the beam-on time each shot (i.e., a column of W) to create a radiation dose distribution that is as close to the ideal as possible. All solvers have the same objective value up to 6 significant digits. The parameters in OCAS initialized solvers are set as $(C, \epsilon) = (10, 10^{-4})$. The running times are shown in Figure 2.5.

2.3.3 Real tumor datasets

Besides randomly generated data, we also applied our TNNLS solver to two real radiation therapy data sets, both obtained from the German Cancer Research Center (DKFZ), of Heidelberg, Germany. The first of these is a skull base tumor case that was treated with

Dataset	FNNLS	PQN-NNLS	OCAS-FNNLS	OCAS-PQN-NNLS	Scaling factor
Phantom tumor	18.69	78.66	**1.0**	5.48	186s
Skull-base tumor	2.21	43.16	**1.0**	9.01	906s
Prostate tumor	1.46	2.04	**1.0**	1.35	134s

Figure 2.5: Running times on the different datasets using the solvers FNNLS, PQN-NNLS and RAND-PQN-NNLS and their corresponding OCAS initialized counterparts can be obtained by multiplying the corresponding entry with the scaling factor. Comparison between running times across different NNLS solvers should be taken with a grain of salt for the stopping criterion for each solver is potentially different. However, stopping criterion between a solver and its OCAS initialized solver are the same and thus can be compared.

carbon ion therapy. In this data set, the size of input matrix W is 227920×6505 and the input vector x is 227920. Just like the dataset in Section 2.3.2, the columns of W represent the radiation energy distribution of an ion beam, while the vector x represents the prescription. The goal of the optimization is to calculate the beam-on time for each individual beam. Note that the default setting of the soft margin to 10 didn't converge to the exact solution, so we used 100 for this dataset. The objective values match upto 6 significant digits as before. The parameters in OCAS initialized solvers are set as $(C, \epsilon) = (100, 10^{-4})$. The running times are shown in Table 2.5.

The second real data set is a prostate carcinoma case that was treated using two opposing beams. In this data set, the input matrix is 8284×7388. The parameters in the OCAS initialized solvers are set as $(C, \epsilon) = (100, 10^{-5})$.

2.3.4 Discussion

The speed up in the case of Real Tumor dataset is only around 2 times compared to the magnitude improvement we get in the case of Random and Phantom tumor datasets for the FNNLS solver. This is to be expected because the input matrix is "tall" and our algorithm does better when we are dealing with "fat" matrices. We have used the exact solver from [57] in combination with the randomized algorithm of [8]. Other exact solvers can also be used.

Note, the speedup is not uniform across the various problems. As, we noted in section "approximate solvers", this depends on the spread of the data. If the datapoints are not clustered along the maximum margin hyperplane, we can solve it pretty quickly using the approximate SVM solver. However, for cases, where this is not true, the running time for our solver is increased.

In the case where the matrix $\boldsymbol{W}$ is sparse, we found that a more aggressive setting for the parameters C, ϵ was required.

2.4 Dual SVM solver

We propose a coordinate descent scheme to solve NNLS. Our method is similar to the successful approach by Hsieh et al. [50] for solving linear SVM, which has been recently generalized to Nonnegative Quadratic Programming (NQP) by Nesterov [87]. Earlier, Franc, Hlavac and Navara [39] proposed a coordinate descent algorithm for NNLS; however, their approach of applying coordinate descent for solving NNLS is not optimized for large datasets. In particular, they compute $\mathbf{W}^\top\mathbf{W}$ which can be expensive. Experiments indicate that we converge quickly to a usable solution.

We optimize one coordinate at a time similar to the previous coordinate descent approach [39]. However, our method avoids the expensive computation of the matrix product $\mathbf{W}^\top\mathbf{W}$. (Since we are updating only one coordinate at a time, computing the full gradient information is unnecessary.) The plain version of our **F**rugal **C**oordinate **D**escent algorithm (**FCD**) is presented in Algorithm 1.

```
Algorithm 1 FCD($\boldsymbol{x}, \mathbf{W}, \mathbf{h}$)
(If $\mathbf{h}$ is not specified, let $\mathbf{h} = 0$.)
Let $\boldsymbol{z} = \sum_i \mathbf{W}_i h_i$.
repeat
    for $i = 1, \ldots, n$ do
        $G = \langle \mathbf{W}_i, \boldsymbol{x} - \boldsymbol{z} \rangle$
        if $h_i = 0$ then
            $G \leftarrow min(G, 0)$
        end if
        if $G \neq 0$ then
            $\boldsymbol{z} \leftarrow \boldsymbol{z} + (\max(h_i - \frac{G}{\|\mathbf{W}_i\|^2}, 0) - h_i)\mathbf{W}_i$
            $h_i \leftarrow \max(h_i - \frac{G}{\|\mathbf{W}_i\|^2}, 0)$
        end if
    end for
until convergence
Output: Vector $\boldsymbol{h}$.
```

The convergence condition of the algorithm can be specified in a couple of different ways. One of them is to specify the stopping threshold of relative change in the norm of the current solution or objective value across outer iterations of the algorithm. Another is to explicitly set the number of outer loops or total computation time. Finally, one could use an approximate satisfiability of KKT conditions of the NNLS problem depending on the required precision of the solution. The proof of convergence and its rate have been previously discussed [87].

There are two important cases for NNLS corresponding to “tall and thin” $(m \gg n)$ and “short and fat” $(m \ll n)$. Some of the algorithms compute the matrix product $\mathbf{W}^\top\mathbf{W}$ $(O(mn^2))$ while others work with $\mathbf{W}$ directly. Our algorithm is especially suitable when the matrix $\mathbf{W}$ is not thin.

We suggest three modifications that could potentially further speed up our algorithm.

They are random permutations [87], shrinking [54], and random projections [8].

2.5 Experiments

In this section, we compare our algorithm with two NNLS solvers called PLB [57] and FNNLS [11]. First, we applied our algorithm FCD and the competing solvers on various synthetic datasets ranging in size from 300×200 to 9000×6000. Next, we consider a large dataset obtained from a phantom commonly used for benchmarking radiosurgery treatment planning systems by Luan et al. [76]. The size of the input matrix W is 42875×20268. Also, we consider a skull base tumor case that was treated with carbon ion therapy which was obtained from the German Cancer Research Center (DKFZ), of Heidelberg, Germany. The size of the input matrix W is 227920×6505. Clinically, each column of the matrix W represents the radiation energy distribution deposited by a "shot" of radiation in Gamma Knife radiosurgery. The matrix x represents the ideal radiation energy deposition as prescribed by the physician. The sought variable h denotes the beam-on time need for each shot (i.e., a column of W) to create a radiation dose distribution that is as close to the ideal as possible. The results of running times for the synthetic and the phantom datasets are shown in Figure 2.6. Similarly, the running times versus objective values for the real tumor dataset is shown in Figure 2.7.

Our algorithm was implemented in MATLAB (`http://www.mathworks.com`) similar to the PLB algorithm. We used the default settings for the competing algorithm as given by the implementation. All of our experiments were run on a 3.2 Ghz Intel machine with 24GB of RAM and the number of threads set to one.

We note that our algorithm converges rapidly to within 1% of final value very fast. This accuracy is good enough in practice for radiation dosage calculations.

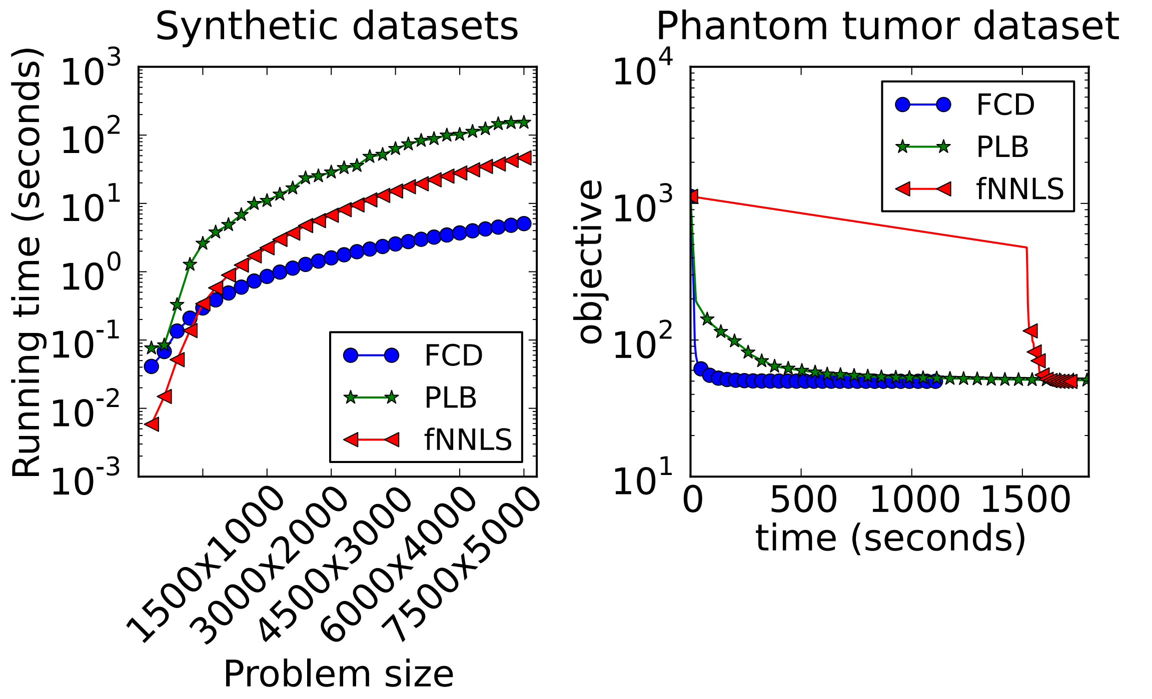


Figure 2.6: (Left) Mean running times for each problem size where the elements of the matrix $\boldsymbol{W}$ and vector $\boldsymbol{x}$ are drawn uniformly at random from $[0, 1]$. The running times for the solvers should be taken with a grain of salt because of the different stopping criterion used. (Right) Running times versus objective values for our (FCD) algorithm and the competing FNNLS and PLB algorithms on the phantom tumor dataset.

2.6 Connections between NMF and SVM

In this section, we will formalize some insights in to the similarities between the NMF and SVM problems. In particular, we will first show how to view SVM as a matrix factorization. Secondly, we will show how the steps in the popular alternate updates scheme for NMF can be reduced to single class SVM problems.

2.6.1 SVM as matrix factorization

Let the set of labeled examples be $\{(\boldsymbol{x}_i, y_i)\}_{i=1}^{N}$, with binary class labels $y_i = \pm 1$ corresponding to two classes, denoted by A and B respectively. Let the mapping $\Phi(\boldsymbol{x}_i)$ be the representation of the input datapoint $\boldsymbol{x}_i$ in space Φ, where we denote the space by the name of the mapping function performing the transformation. We now consider the problem of

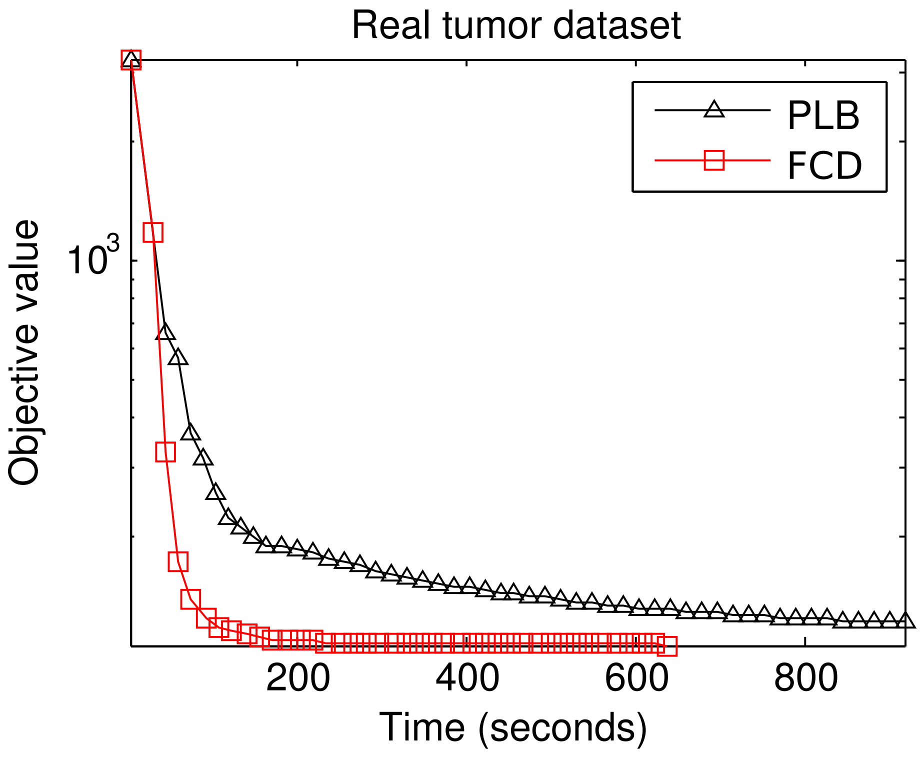


Figure 2.7: Running times versus objective values for FCD and PLB are shown for the real tumor dataset .

computing the maximum margin hyperplane for SVM in the case where the classes are linearly separable and the hyperplane passes through origin (We will relax this constraint presently.).

The dual quadratic optimization problem for hard-margin SVM [102] is given by minimizing the following loss function:

$$S(\boldsymbol{\alpha}) = \frac{1}{2}\sum_{i,j=1}^{n} \alpha_i \alpha_j y_i y_j k(\boldsymbol{x}_i, \boldsymbol{x}_j) - \sum_{i=1}^{n} \alpha_i \qquad (2.12)$$
$$\text{subject to } \alpha_i \geq 0,\ i \in \{1, \ldots, n\},$$

where $k(\boldsymbol{x}_i, \boldsymbol{x}_j)$ is a kernel that computes the inner product $\Phi(\boldsymbol{x}_i)^T \Phi(\boldsymbol{x}_j)$ in the space Φ by performing all operations only in the original data space on x_i and x_j, thus defining a Hilbert space Φ.

The first sum can be split into three terms: two terms contain kernels of elements that belong to the same respective class (one term per class), and the third contains only the kernel between elements of the two classes. This rearrangement of terms allows us to drop class labels y_i, y_j from the objective function. Denoting $k(\boldsymbol{x}_i, \boldsymbol{x}_j)$ with k_{ij} and defining $\rho_{ij} = \alpha_i \alpha_j k_{ij}$ for conciseness, we have:

$$\min_{\boldsymbol{\alpha}} \frac{1}{2} \left(\sum_{ij \in A} \rho_{ij} - 2 \sum_{\substack{i \in B \\ j \in A}} \rho_{ij} + \sum_{ij \in B} \rho_{ij} \right) - \sum_{i=1}^{n} \alpha_i \tag{2.13}$$
$$\text{subject to } \alpha_i \geq 0, i \in \{1..n\}.$$

Noticing the square and the fact that $k_{ij} = \Phi(\boldsymbol{x}_i)^T \Phi(\boldsymbol{x}_j)$ we rewrite the problem as:

$$\min_{\boldsymbol{\alpha}} \frac{1}{2} \|\Phi(\boldsymbol{X}_A)\boldsymbol{\alpha}_A - \Phi(\boldsymbol{X}_B)\boldsymbol{\alpha}_B\|_2^2 - \sum_{i \in \{A,B\}} \alpha_i \tag{2.14}$$
$$\text{subject to } \alpha_i \geq 0,$$

where the matrices $\boldsymbol{X}_A, \boldsymbol{X}_B$ contain the datapoints corresponding to groups A and B respectively with the stacking being column-wise. The map Φ applied to a matrix corresponds to mapping each individual column vector of the matrix using Φ and stacking them to generate the new matrix. The vectors $\boldsymbol{\alpha}_A$ and $\boldsymbol{\alpha}_B$ contain coefficients of the support vectors of the two groups A and B respectively. We will use the vector $\boldsymbol{\alpha}$ to denote the concatenation of vectors $\boldsymbol{\alpha}_A, \boldsymbol{\alpha}_B$. Expression (2.14) is a form of matrix factorization problem and resembles NMF with an additional term in the objective [67]. The above formulation enables other metrics $D(\Phi(\boldsymbol{X_A})\boldsymbol{\alpha}_A || \Phi(\boldsymbol{X}_B)\boldsymbol{\alpha}_B)$ than least squares for SVM such as more general Bregman divergence [29]. However, to be computationally efficient the metric used has to admit the use of the kernel trick. The matrices $\boldsymbol{X}_A, \boldsymbol{X}_B$ contain the datapoints corresponding to groups A and B respectively with the stacking being column-wise. The map Φ applied to a matrix corresponds to mapping each individual column vector of the matrix using Φ and stacking them to generate the new matrix. The vectors $\boldsymbol{\alpha}_A$ and $\boldsymbol{\alpha}_B$ contain the dual variables of the two groups A and B respectively. We will use the vector $\boldsymbol{\alpha}$ to denote the concatenation of vectors $\boldsymbol{\alpha}_A, \boldsymbol{\alpha}_B$. Expression (2.14) is a form of matrix factorization problem and resembles NMF with an additional term in the objective [67]. The above formulation enables other metrics $D(\Phi(\boldsymbol{X_A})\boldsymbol{\alpha}_A || \Phi(\boldsymbol{X}_B)\boldsymbol{\alpha}_B)$

than least squares for SVM such as more general Bregman divergence [29]. However, to be computationally efficient the metric used has to admit the use of the kernel trick.

2.6.2 NMF reduced to sequence of SVMs

We prove that NMF can be reduced to solving a sequence of hard-margin single class SVM problems. We are now in a position to sketch the reduction. The vectors $\boldsymbol{x}$ and $\boldsymbol{h}$ are the corresponding column vectors of the matrices $\boldsymbol{X}$ and $\boldsymbol{H}$. Let $\boldsymbol{D}$ denote the diagonal matrix whose diagonal is given by the vector $\frac{1}{\boldsymbol{W}^T\boldsymbol{x}}$. Also, let $\boldsymbol{h} = \boldsymbol{D}\boldsymbol{z}$. Then,

$$\begin{aligned} G(\boldsymbol{z}) &= \frac{1}{2}\|\boldsymbol{x} - \boldsymbol{W}\boldsymbol{D}\boldsymbol{z}\|_2^2 \\ &= \frac{1}{2}\boldsymbol{z}^T(\boldsymbol{W}\boldsymbol{D})^T(\boldsymbol{W}\boldsymbol{D})\boldsymbol{z} - \boldsymbol{x}^T\boldsymbol{W}\boldsymbol{D}\boldsymbol{z} + \frac{1}{2}\boldsymbol{x}^T\boldsymbol{x} \\ &= \frac{1}{2}\boldsymbol{z}^T(\boldsymbol{W}\boldsymbol{D})^T(\boldsymbol{W}\boldsymbol{D})\boldsymbol{z} - \boldsymbol{1}^T\boldsymbol{z} + \frac{1}{2}\boldsymbol{x}^T\boldsymbol{x} \end{aligned}$$

Ignoring the $\frac{1}{2}\boldsymbol{x}^T\boldsymbol{x}$, which does not change the location of the minimum, we see that it is an instance of the SVM objective in equation 2.13, with $\boldsymbol{\alpha}$ corresponding to $\boldsymbol{z}$ and $\boldsymbol{s}_i$ corresponding to $\boldsymbol{W}_i\boldsymbol{D}_{ii}$ with the kernel function being linear. We have a single class maximum margin classifier passing through origin where the datapoints given by $\{\boldsymbol{W}_iD_{ii}\}_i^n$ lie in the positive orthant. The NMF problem can be written as a sequence of sub-problems as follows:

$$\begin{gathered} \min_{\boldsymbol{H}} \frac{1}{2}\|\mathrm{vec}(\boldsymbol{X}) - (\boldsymbol{I} \otimes \boldsymbol{W})\mathrm{vec}(\boldsymbol{H})\|_F^2 \\ \text{s.t.}\boldsymbol{H} \geq \boldsymbol{0} \\ \min_{\boldsymbol{W}} \frac{1}{2}\|\mathrm{vec}(\boldsymbol{X}^T) - (\boldsymbol{I} \otimes \boldsymbol{H})\mathrm{vec}(\boldsymbol{W}^T)\|_F^2 \\ \text{s.t.}\boldsymbol{W} \geq \boldsymbol{0} \end{gathered}$$

2.6.3 Our algorithms — Sign-insensitive Kernel SVMs

In this section, we derive two new updates for solving SVM's with sign-insensitive kernels based on NMF. A sign-insensitive kernel is one whose output can be either positive, negative or zero. One of them follows immediately by appealing to the semi-NMF formulation. The other is derived using the idea from NMF updates of Lee and Seung [67].

semi-NMF SVM

We differentiate the objective (2.14) with respect to $\boldsymbol{\alpha}_A$:

$$\begin{aligned}\frac{\partial S}{\partial \boldsymbol{\alpha}_A} &= \Phi(\boldsymbol{X}_A)^T\Phi(\boldsymbol{X}_A)\boldsymbol{\alpha}_A - (\Phi(\boldsymbol{X}_A)^T\Phi(\boldsymbol{X}_B)\boldsymbol{\alpha}_B + \mathbf{1}) \\ &= K(\boldsymbol{X}_A, \boldsymbol{X}_A)\boldsymbol{\alpha}_A - (K(\boldsymbol{X}_A, \boldsymbol{X}_B)\boldsymbol{\alpha}_B + \mathbf{1}) \end{aligned} \tag{2.15}$$

We slightly abuse notation to define kernel for matrices as follows: $K(\boldsymbol{C}, \boldsymbol{D})$ is given by the matrix whose $(i, j)^{th}$ element is given by the inner product of i^{th} and j^{th} datapoints of matrices $\boldsymbol{C}$,$\boldsymbol{D}$ respectively in the feature space Φ for all values of (i, j) in range. We note that the derivative has a positive and a negative component. We use the following notation to represent kernel matrices:

$$\begin{aligned} K(\boldsymbol{X}_A, \boldsymbol{X}_B) &= \boldsymbol{K}_{AB} \\ K(\boldsymbol{X}_A, \boldsymbol{X}_A) &= \boldsymbol{K}_A \end{aligned}$$

and their decomposition into $\boldsymbol{K}^+$ and $\boldsymbol{K}^-$:

$$K_{ij}^+ = \begin{cases} K_{ij} & K_{ij} > 0, \\ 0 & \text{otherwise,} \end{cases} \qquad K_{ij}^- = \begin{cases} |K_{ij}| & K_{ij} < 0, \\ 0 & \text{otherwise.} \end{cases}$$

Similarly, we take the derivative with respect to $\boldsymbol{\alpha}_B$. Recalling the updates for semi-NMF (Equation (2.9)), we write down the multiplicative updates for problem (2.14):

$$\begin{aligned} \boldsymbol{\alpha}_A &= \boldsymbol{\alpha}_A \odot \sqrt{\frac{\boldsymbol{K}_{AB}^+\boldsymbol{\alpha}_B + \boldsymbol{K}_A^-\boldsymbol{\alpha}_A + \mathbf{1}}{\boldsymbol{K}_A^+\boldsymbol{\alpha}_A + \boldsymbol{K}_{AB}^-\boldsymbol{\alpha}_B}} \\ \boldsymbol{\alpha}_B &= \boldsymbol{\alpha}_B \odot \sqrt{\frac{\boldsymbol{K}_{BA}^+\boldsymbol{\alpha}_A + \boldsymbol{K}_B^-\boldsymbol{\alpha}_B + \mathbf{1}}{\boldsymbol{K}_B^+\boldsymbol{\alpha}_B + \boldsymbol{K}_{BA}^-\boldsymbol{\alpha}_A}} \end{aligned} \tag{2.16}$$

where $\mathbf{1}$ is an appropriately sized vector of ones and $\odot$ denotes the Hadamard product as before. The proof of the updates directly follows from the proof of semiNMF updates [31, 84].

MUSIK

If instead of using semi-NMF formulation, we use NMF to derive the updates, i.e. updating by the ratio of the negative to the positive part of the gradient, we get the following:

$$\begin{aligned}\boldsymbol{\alpha}_A &= \boldsymbol{\alpha}_A \odot \frac{\boldsymbol{K}_{AB}^{+}\boldsymbol{\alpha}_B + \boldsymbol{K}_{A}^{-}\boldsymbol{\alpha}_A + \mathbf{1}}{\boldsymbol{K}_{A}\boldsymbol{\alpha}_A + \boldsymbol{K}_{AB}^{-}\boldsymbol{\alpha}_B} \\ \boldsymbol{\alpha}_B &= \boldsymbol{\alpha}_B \odot \frac{\boldsymbol{K}_{BA}^{+}\boldsymbol{\alpha}_A + \boldsymbol{K}_{B}^{-}\boldsymbol{\alpha}_B + \mathbf{1}}{\boldsymbol{K}_{B}^{+}\boldsymbol{\alpha}_B + \boldsymbol{K}_{BA}^{-}\boldsymbol{\alpha}_A}\end{aligned} \tag{2.17}$$

In these updates, we note that the split is not done as in the previous section. Instead the kernel matrix is split as follows:

$$K_{ij}^{+} = \begin{cases} K_{ij} & K_{ij} > 0, \\ K_{ij} + D_{ii} & i = j, \\ 0 & \text{otherwise,} \end{cases}$$

$$K_{ij}^{-} = \begin{cases} |K_{ij}| & K_{ij} < 0, \\ |K_{ij}| + D_{ii} & i = j, \\ 0 & \text{otherwise.} \end{cases}$$

In other words the new split when defined in terms of the old split looks like:

$$\begin{aligned}\boldsymbol{K}_{\text{new}}^{+} &= \boldsymbol{K}^{+} + \boldsymbol{D} \\ \boldsymbol{K}_{\text{new}}^{-} &= \boldsymbol{K}^{-} + \boldsymbol{D} \\ \boldsymbol{K} &= \boldsymbol{K}_{\text{new}}^{+} - \boldsymbol{K}_{\text{new}}^{-} \\ &= \boldsymbol{K}^{+} - \boldsymbol{K}^{-}\end{aligned}$$

where matrix $\boldsymbol{D}$ is a nonnegative diagonal matrix. We note that in practice, we explicitly work with the matrices in the old split even though we are using the new split. This is compensated for in the new updates. The construction of matrix $\boldsymbol{D}$ is as follows:

$$[\boldsymbol{D}_A]_{ii} = \max(0, \sum_{j \neq i} \left[\boldsymbol{K}_A^-\right]_{ij} - \left[\frac{\boldsymbol{K}_{AB}^- \boldsymbol{\alpha}_B}{\boldsymbol{\alpha}_A}\right]_i). \tag{2.18}$$

$$[\boldsymbol{D}_B]_{ii} = \max(0, \sum_{j \neq i} \left[\boldsymbol{K}_B^-\right]_{ij} - \left[\frac{\boldsymbol{K}_{BA}^- \boldsymbol{\alpha}_A}{\boldsymbol{\alpha}_B}\right]_i). \tag{2.19}$$

This ensures that $\boldsymbol{K}_{\text{new}}^-$ becomes positive semi-definite. At each new iteration of the updates, we choose $\boldsymbol{D}$ adaptively using eq 2.18 and the new updates are given by:

$$\begin{aligned}
\boldsymbol{\alpha}_A &= \boldsymbol{\alpha}_A \odot \frac{\boldsymbol{K}_{AB}^+ \boldsymbol{\alpha}_B + \boldsymbol{K}_A^- \boldsymbol{\alpha}_A + \mathbf{1} + \boldsymbol{D}_A \boldsymbol{\alpha}_A}{\boldsymbol{K}_A \boldsymbol{\alpha}_A + \boldsymbol{K}_{AB}^- \boldsymbol{\alpha}_B + \boldsymbol{D}_A \boldsymbol{\alpha}_A} \\
\boldsymbol{\alpha}_B &= \boldsymbol{\alpha}_B \odot \frac{\boldsymbol{K}_{BA}^+ \boldsymbol{\alpha}_A + \boldsymbol{K}_B^- \boldsymbol{\alpha}_B + \mathbf{1} + \boldsymbol{D}_B \boldsymbol{\alpha}_B}{\boldsymbol{K}_B^+ \boldsymbol{\alpha}_B + \boldsymbol{K}_{BA}^- \boldsymbol{\alpha}_A + \boldsymbol{D}_B \boldsymbol{\alpha}_B}
\end{aligned} \tag{2.20}$$

This condition is required for convergence properties of the updates. We defer the proof to the appendix. We note that in the case of nonnegative kernels i.e. kernels which output a nonnegative value for all valid inputs, the split can be done trivially by having $\boldsymbol{K}^-$ set to zero and $\boldsymbol{K}^+$ set to the original kernel matrix.

We call this new algorithm **Multiplicative Updates for sign-insensitive Kernel** SVM (MUSIK). We note that besides solving SVM problem, this formulation presents multiplicative updates for semi-NMF alternative to Ding et al. [31]. Further, it positions us to extend to the general, soft-margin, biased SVM (Sections 2.6.6 and 2.6.7).

2.6.4 Nonnegative Quadratic Programming

It is well known that the dual formulation (2.13) can be represented as a quadratic programming problem with a nonnegativity constraint on alphas [102]:

$$F(\boldsymbol{\alpha}) = \frac{1}{2}\boldsymbol{\alpha}^T \boldsymbol{A} \boldsymbol{\alpha} - \mathbf{1}^T \boldsymbol{\alpha}, \tag{2.21}$$

where $\boldsymbol{A}$ is the Gram matrix of data points whose values are scaled by corresponding label products ($A_{ij} = y_i y_j K(\boldsymbol{x}_i, \boldsymbol{x}_j)$) and $\mathbf{1}$ denotes an appropriately sized vector of ones. A more general form of quadratic programming can be written as:

$$F(\boldsymbol{\alpha}) = \frac{1}{2}\boldsymbol{\alpha}^T \boldsymbol{A}\boldsymbol{\alpha} + \boldsymbol{b}^T\boldsymbol{\alpha}. \tag{2.22}$$

This problem is called Nonnegative Quadratic Programming (NQP) when the nonnegativity constraint is enforced on $\boldsymbol{\alpha}$. SVM is a special case of NQP.

Parameter free multiplicative updates for NQP have been previously introduced [104]. For the special case of SVM the updates from [104] have the following form:

$$\boldsymbol{\alpha} = \boldsymbol{\alpha} \odot \frac{\mathbf{1} + \sqrt{\mathbf{1} + 4(\boldsymbol{A}^+\boldsymbol{\alpha}) \odot (\boldsymbol{A}^-\boldsymbol{\alpha})}}{2(\boldsymbol{A}^+\boldsymbol{\alpha})}, \tag{2.23}$$

where $\boldsymbol{A}^+$ and $\boldsymbol{A}^-$ are defined as:

$$A^+_{ij} = \begin{cases} A_{ij} & A_{ij} > 0, \\ 0 & \text{otherwise}, \end{cases} \tag{2.24}$$

$$A^-_{ij} = \begin{cases} |A_{ij}| & A_{ij} < 0, \\ 0 & \text{otherwise}. \end{cases} \tag{2.25}$$

Reformulated SVMs for which we have derived multiplicative updates in Section 2.6.3, can be represented as NQP with a special form of $\boldsymbol{A}$ and $\boldsymbol{\alpha}$:

$$\tilde{\boldsymbol{\alpha}} = \begin{bmatrix} \boldsymbol{\alpha}_A & \boldsymbol{\alpha}_B \end{bmatrix}^T \tag{2.26}$$

$$\tilde{\boldsymbol{A}} = \begin{bmatrix} K(\boldsymbol{X}_A, \boldsymbol{X}_A) & -K(\boldsymbol{X}_A, \boldsymbol{X}_B) \\ -K(\boldsymbol{X}_B, \boldsymbol{X}_A) & K(\boldsymbol{X}_B, \boldsymbol{X}_B) \end{bmatrix} \tag{2.27}$$

The block structure of $\tilde{\boldsymbol{A}}$ allows for a clear and easy split of this matrix into $\tilde{\boldsymbol{A}}^+$ and $\tilde{\boldsymbol{A}}^-$ after which it is clear that the multiplicative update of NQP (2.23) is different from the updates in (2.17).

In order to highlight that difference we generated a random matrix $\boldsymbol{A}$ of form (2.27) for dimension 2 and solved the problem using the method introduced in Section 2.6.3 and the update (2.23), introduced in [104]. Convergence paths for both algorithms are shown in Figure 2.8. The figure shows a paraboloid of the two dimensional objective function generated by a random construction of the Gram matrix satisfying the structure in (2.27). MUSIK and M^3 algorithms [104] were applied to this problem starting at $\boldsymbol{\alpha} = [1, 1]^T$. As expected, both algorithms arrive at the unique solution of the convex problem, however they follow different paths and MUSIK takes fewer steps.

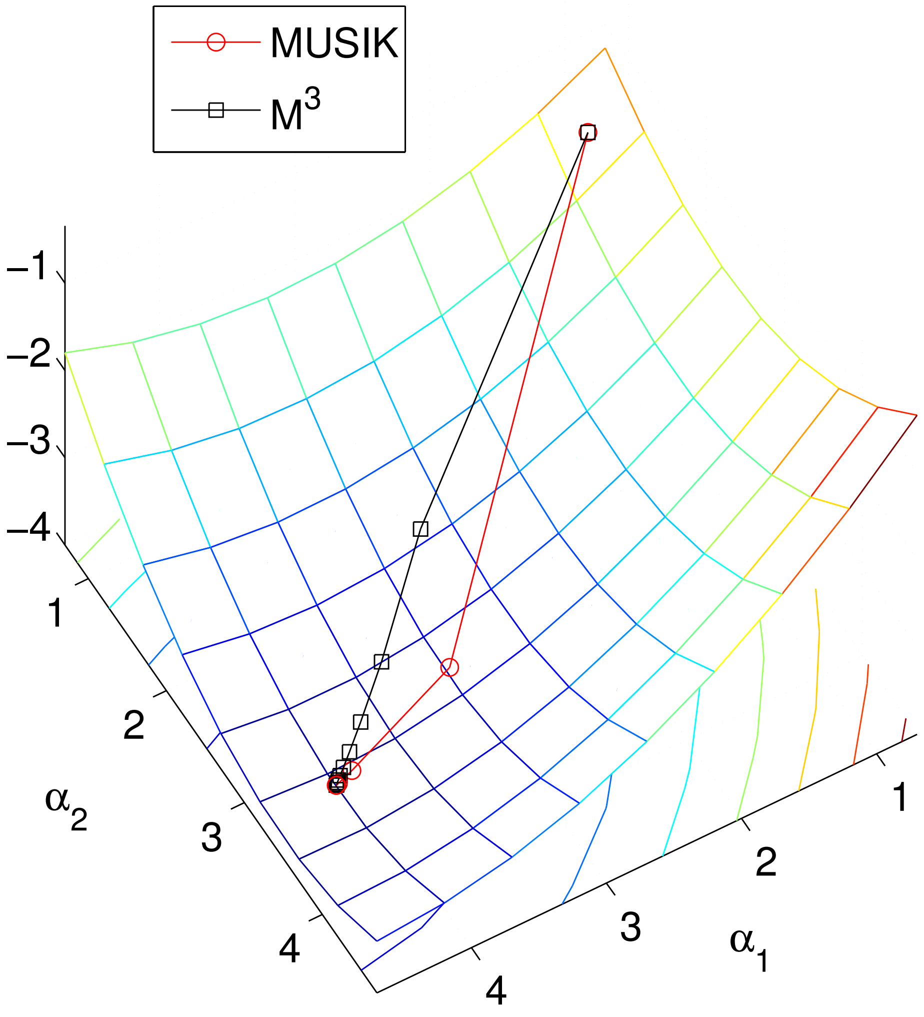


Figure 2.8: The figure shows a paraboloid of the two dimensional objective function generated by a random construction of the Gram matrix satisfying the structure in (2.27). MUSIK and M^3 algorithms [104] were applied to this problem starting at $\boldsymbol{\alpha} = [1, 1]^T$. As expected, both algorithms arrive at the unique solution of the convex problem, however they follow different routes and the nonnegative kernel SVM takes fewer steps.

Figure 2.8 demonstrates the differences between the methods on a single problem case.

In order to have an aggregate measure of the difference we have implemented the following simulation. We have randomly constructed 100 positive definite matrices $\tilde{\boldsymbol{A}}$ with the structure required by our algorithm (2.27) (recall that the structure comes from the requirement of the kernel to be nonnegative) for each dimension from the following list: (16, 32, 64, 128, 256, 512, 1024, 2048). Equal number of data points of each class was assumed. For each of these matrices we have solved the QP problem (2.21) using `quadprog` function of Matlab. All 800 problems were constructed to be well conditioned and solvable by this function. Knowing the exact solution to a given problem we ran both MUSIK and M^3 until they were within the given percent of the solution (convergence tolerances of $1\%, 0.1\%$ and 0.01% were used). Although the absolute value of this percent depends on the distance of the optimum from the base hyperplane it is not an issue in our case due to the shift $\boldsymbol{b}$ being equal to $\mathbf{1}$ for all the problems. For each problem we have computed the ratio of the number of iterations it took the M^3 algorithm to reach within the given percent of the solution to the number of iterations it took MUSIK to finish. Results of this simulation are displayed in Figure 2.9.

2.6.5 Decomposition

As we show in the previous section MUSIK updates converge faster than M^3. In part this is due to the better asymptotic bound on the convergence rate which we discuss in Section 2.6.9. However, the next feature that improves the convergence rate is splitting $\boldsymbol{\alpha}$ into parts. Separately updating two groups of alphas is similar to decomposition techniques [91], only the way we set the problem does not require any additional heuristics.

In order to demonstrate that decomposition affects the performance in our multiplicative updates, we compare it with MUSIK algorithm in which elements of $\boldsymbol{\alpha}$ are updated

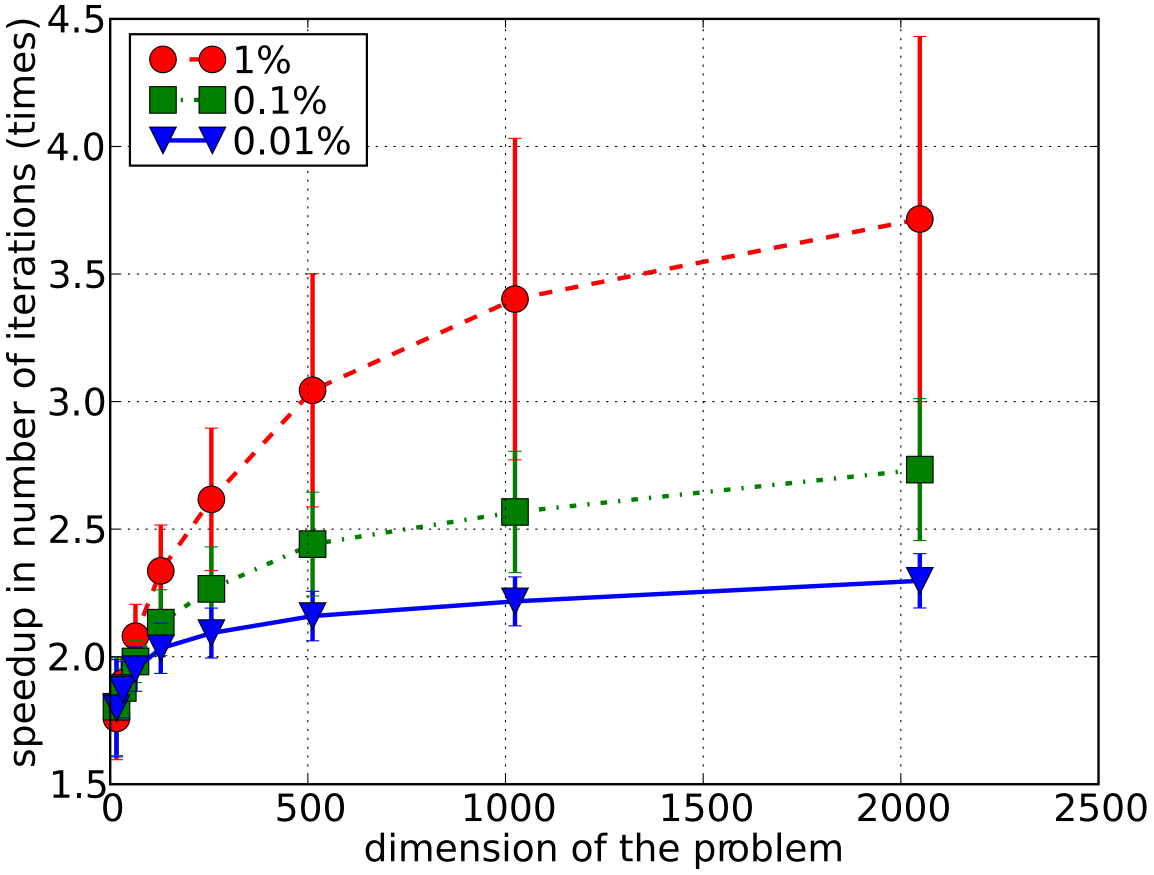


Figure 2.9: The figure shows the average ratio of the number of iterations for M^3 to the number of iterations for MUSIK taken to achieve given tolerance on the same problem (up is good). Computation is done at error bar points, the lines connecting them are for the visual guide only. The larger the problem size the smaller the number of iterations the algorithm needs compared to M^3, which can be up to 4 times less. Since the running time per iteration is comparable for both algorithms 4 times improvement in iterations means 4 times faster. Even for 0.01% distance from the solution our algorithm is more than two times faster on reasonable sized problems.

simultaneously:

$$\begin{bmatrix} \boldsymbol{\alpha}_A \\ \boldsymbol{\alpha}_B \end{bmatrix} = \begin{bmatrix} \boldsymbol{\alpha}_A \\ \boldsymbol{\alpha}_B \end{bmatrix} \odot \frac{\begin{bmatrix} \boldsymbol{K}_A^- & \boldsymbol{K}_{AB}^+ \\ \boldsymbol{K}_{BA}^+ & \boldsymbol{K}_B^- \end{bmatrix} \begin{bmatrix} \boldsymbol{\alpha}_A \\ \boldsymbol{\alpha}_B \end{bmatrix} + \mathbf{1}}{\begin{bmatrix} \boldsymbol{K}_A^+ & \boldsymbol{K}_{AB}^- \\ \boldsymbol{K}_{BA}^- & \boldsymbol{K}_B^+ \end{bmatrix} \begin{bmatrix} \boldsymbol{\alpha}_A \\ \boldsymbol{\alpha}_B \end{bmatrix}} \tag{2.28}$$

We call this modification of the algorithm integrative MUSIK (iMUSIK).

Figure 2.10 shows objective function, training error and testing error as a function of iteration number for MUSIK, iMUSIK and M^3 algorithms. Asymptotically the fastest convergence is exhibited by the MUSIK algorithm and the iMUSIK algorithm fall between MUSIK and M^3. The difference between iMUSIK and MUSIK is only due to the decomposition. Decomposition improves the convergence rate as improved updated parameters are used when updating the remaining parameters.

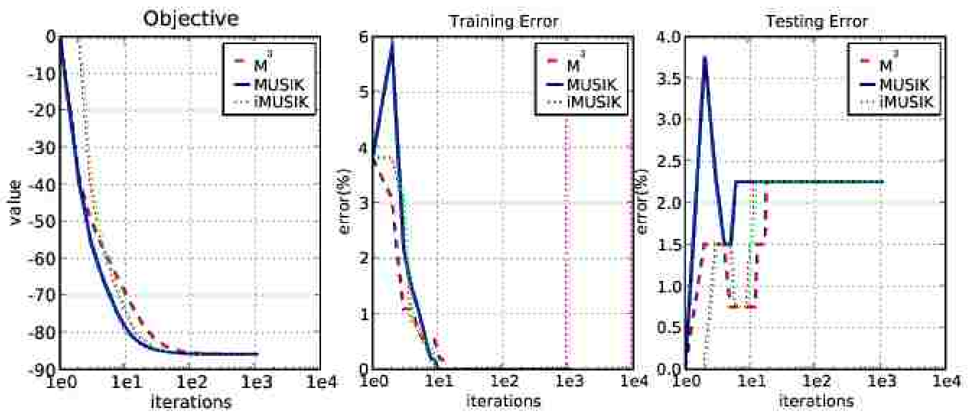


Figure 2.10: Differences in convergence on the UCI breast cancer dataset for MUSIK, M^3 and integrated MUSIK (iMUSIK) algorithms.

If we start making the size of the subsets updated at once smaller, we arrive at chunking algorithms of which SMO [94] represents the extreme case. In the extreme case we can update only a single element of $\boldsymbol{\alpha}$ per iteration. In this case we end up with multiplicative variant of the Kernel Adatron (KA) algorithm [27].

KA is a simple gradient ascent procedure for learning support vectors with adaptive learning rate. It has a learning rate parameter which needs to be set. The updates for kernel adatron are as follow:

$$\alpha_i = \alpha_i + \eta_i(1 - y_i(\sum_j K(x_i, x_j)y_j\alpha_j)) \tag{2.29}$$

where η_i is the learning rate parameter. In the case of support vector machines it is set as $\eta_i = \frac{1}{K(x_i,x_i)}$. If we instead set the learning rate to be

$$\eta_i = \frac{\alpha_i}{\boldsymbol{C}_i\alpha} \tag{2.30}$$

we obtain a multiplicative algorithm for KA through MUSIK updates. Note that the matrix $\boldsymbol{C}$ corresponds to the matrix in the denominator of the updates in equation (2.28) and we subscript it to denote the corresponding row vector. We get the multiplicative updates of MUSIK done sequentially. Kernel adatron (KA) belongs to the class of subset methods and can be shown equivalent to the popular SMO algorithm [55].

When heuristics are used to choose which α_i to update KA demonstrates very fast convergence. Thus it is expected that multiplicative KA with heuristics is considerably faster than MUSIK. However, the attractive feature of M^3 and MUSIK is the absence of hyper-parameters, a feature that is removed by the need to use heuristics in multiplicative KA algorithm.

Also, the KA algorithm can be adapted to solve the NMF problem. This was indeed done by applying sequential updates to solve nonnegative least squares problem (NNLS) [39]. This was subsequently adapted for solving NMF [117].

2.6.6 Soft Margin SVM

We can extend the multiplicative updates to incorporate upper bound constraints of the form $\alpha_i \leq l$ where l is a constant as follows:

$$\alpha_i = \min\{\alpha_i, l\} \tag{2.31}$$

These are referred to as box constraints, since they bound α_i from both above and below.

The dual problem for soft margin SVM is given by:

$$\min_{\boldsymbol{\alpha}} \frac{1}{2} \sum_{i,j=1}^{n} \alpha_i \alpha_j y_i y_j k(\boldsymbol{x}_i, \boldsymbol{x}_j) - \sum_{i=1}^{n} \alpha_i \tag{2.32}$$
$$\text{subject to } 0 \leq \alpha_i \leq l, i \in \{1..n\}.$$

The parameter l is a regularization term, which provides a way to avoid overfitting. We note that this objective differs from hard margin SVM (2.13) only in box constraints. Soft margin SVM involves box constraints and that can be handled by the above formulation. At each update of $\boldsymbol{\alpha}$, we implement a step given by (2.31) to ensure the box constraint is satisfied.

2.6.7 Bias

SVM with a bias term is given by the following formulation:

$$S(\boldsymbol{\alpha}) = \frac{1}{2}\sum_{i,j=1}^{n} \alpha_i\alpha_j y_i y_j k(\boldsymbol{x}_i, \boldsymbol{x}_j) - \sum_{i=1}^{n}\alpha_i \tag{2.33}$$
$$\text{subject to } \alpha_i \geq 0, \sum_i y_i\alpha_i = 0,\ i \in \{1, \ldots, n\}.$$

We can incorporate bias into MUSIK by considering the following modifications as shown in Keerthi et al. [56]. We introduce a weight variable λ and rewrite the equality constraint $\sum_i y_i\alpha_i = 0$ as the following two equality constraints :

$$\sum_{i\in A}\alpha_i = \lambda, \sum_{j\in B}\alpha_j = \lambda$$

Let us introduce new variables $\beta_k = \alpha_k/\lambda$ for all k and we obtain the following new objective :

$$S_1(\boldsymbol{\beta}, \lambda) = \frac{\lambda^2}{2}\sum_{i,j=1}^{n} \beta_i\beta_j y_i y_j k(\boldsymbol{x}_i, \boldsymbol{x}_j) - 2\lambda \tag{2.34}$$
$$\text{s.t. } \beta_i \geq 0, \sum_{i\in A}\beta_i = 1, \sum_{i\in B}\beta_i = 1.$$

First, we optimize λ keeping the vector β fixed and then alternate by optimizing β keeping λ fixed. Optimizing with respect to λ gives :

$$\lambda = \frac{2}{\sum_i\sum_j \beta_i\beta_j y_i y_j k(\boldsymbol{x}_i, \boldsymbol{x}_j)} \tag{2.35}$$

We substitue this value of λ in the above formulation to get the new objective :

$$S_2(\boldsymbol{\beta}) = \frac{1}{2}\sum_{i,j=1}^{n} \beta_i\beta_j y_i y_j k(\boldsymbol{x}_i, \boldsymbol{x}_j) \tag{2.36}$$
$$\text{s.t. } \beta_i \geq 0, \sum_{i\in A}\beta_i = 1, \sum_{i\in B}\beta_i = 1.$$

We update the β vector corresponding to each group alternatingly and derive the following updates similar to Eggert and Körner [37]:

$$\begin{aligned} \boldsymbol{\beta}_A &= \boldsymbol{\beta}_A \odot \frac{\boldsymbol{K}_{AB}\boldsymbol{\beta}_B + \mathbf{1}\boldsymbol{\beta}_A^T\boldsymbol{K}_A\boldsymbol{\beta}_A}{\boldsymbol{K}_A\boldsymbol{\beta}_A + \mathbf{1}\boldsymbol{\beta}_A^T\boldsymbol{K}_{AB}\boldsymbol{\beta}_B} \\ \boldsymbol{\beta}_B &= \boldsymbol{\beta}_B \odot \frac{\boldsymbol{K}_{BA}\boldsymbol{\beta}_A + \mathbf{1}\boldsymbol{\beta}_B^T\boldsymbol{K}_B\boldsymbol{\beta}_B}{\boldsymbol{K}_B\boldsymbol{\beta}_B + \mathbf{1}\boldsymbol{\beta}_B^T\boldsymbol{K}_{BA}\boldsymbol{\beta}_A} \end{aligned} \tag{2.37}$$

when there is normalization involved. The updates are not guaranteed to be non-increasing but in practice converge to global optimum – an observation similar to [37]. The updates assume that the kernels are nonnegative. A nonnegative kernel is one whose output is always nonnegative irrespective of its input. Similar to MUSIK algorithm, the updates can be extended to general kernels.

2.6.8 Fixed points

We show that the updates have fixed points wherever the objective function $S(\boldsymbol{\alpha})$ achieves its minimum value. Let $\boldsymbol{\alpha}^*$ be the global minimum. Let us consider the coefficients corresponding to group A. At such a point, we either have that each α_A^i is greater than zero and derivative of objective with respect to α_A^i vanishes or it is zero and derivative is greater than or equal to zero. The first condition applies to the positive elements of $\boldsymbol{\alpha}_A^*$ with the requirement that their corresponding terms in the gradient be zero. The derivatives of these terms are given by:

$$\begin{aligned} \left.\frac{\partial S}{\partial \alpha_A^i}\right|_{\boldsymbol{\alpha}_A^*} &= (K(\boldsymbol{X}_A, \boldsymbol{X}_A)\boldsymbol{\alpha}_A^*)_i - (K(\boldsymbol{X}_A, \boldsymbol{X}_B)\boldsymbol{\alpha}_B^*)_i - 1 \\ &= -(\boldsymbol{K}_{AB}^+\boldsymbol{\alpha}_B^*)_i - (\boldsymbol{K}_A^-\boldsymbol{\alpha}_A^*)_i - 1 + (\boldsymbol{K}_A\boldsymbol{\alpha}_A^*)_i + (\boldsymbol{K}_{AB}^-\boldsymbol{\alpha}_B^*)_i \end{aligned} \tag{2.38}$$

This condition applies to the support vectors. For non-support vectors corresponding to them being zero we have the second condition. Fixed points occur when one of the following two conditions hold. Either the element to be updated is greater than zero and multiplicative factor is unity or the element is zero. We can see that in the case of the element being non-zero the multiplicative factor is indeed one. Similar analysis can be done for coefficients corresponding to group B. Thus the updates have fixed points wherever the objective reaches its minimum value. We note that at the fixed point M^3 and MUSIK are the same.

2.6.9 Asymptotic convergence

The M^3 algorithm [104] observed a rapid decay of non-support vector coefficients and did an analysis of rates of asymptotic convergence. They perturb one of the non-support vector coefficients, say $\boldsymbol{\alpha}_i$ away from the fixed point to some nonzero value $\delta\boldsymbol{\alpha}_i$ and fix all the remaining values. Applying their multiplicative update from (2.23) gives a bound on the asymptotic rate of convergence.

Let $d_i = K(\boldsymbol{x}_i, \boldsymbol{w})/\sqrt{K(\boldsymbol{w}, \boldsymbol{w})}$ denote the perpendicular distance in the feature space from $\boldsymbol{x}_i$ to the maximum margin hyperplane and $d = \min_i d_i = 1/\sqrt{K(\boldsymbol{w}, \boldsymbol{w})}$ denote the one-sided margin to the maximum-margin hyperplane. Also, $l_i = \sqrt{K(\boldsymbol{x}_i, \boldsymbol{x}_i)}$ denotes the distance of $\boldsymbol{x}_i$ to the origin in the feature space and $l = \max_i l_i$ denote the largest such distance. The following bound on the asymptotic rate of convergence $\gamma_i^{M^3}$ was established:

$$\gamma_i^{M^3} \leq [1 + \frac{1}{2}\frac{(d_i - d)d}{l_i l}]^{-1} \tag{2.39}$$

We do a similar analysis for rate of asymptotic convergence of the multiplicative updates of the MUSIK algorithm in the case of nonnegative kernels. We perturb one of the non-support vector coefficients fixing all the other coefficients and apply the multiplicative update. This enables us to calculate a bound on rate of convergence. A bound on the asymptotic rate of convergence in terms of geometric quantities is given as follows:

$$\gamma_i^{MUSIK} \leq [1 + \frac{(d_i - d)d}{l_i l}]^{-1} \tag{2.40}$$

The proof sketch can be found in appendix. It is for non-negative kernels, but we note that they constitute the majority of the popular and widely used kernels. We note that our bound is tighter compared to the M^3 algorithm as $\gamma_i^{MUSIK} \leq \gamma_i^{M^3}$.

2.6.10 Adapting NMF algorithms for SVM

Multiplicative updates are not the only way to solve NMF-type problems. For example Lin [75] shows a fast projected-gradient algorithm for solving NMF. Zdunek and Cichoki [117] , Dhillon and Sra [29] etc give more algorithms for solving NMF. Projected gradient algorithm can be used for solving SVM with a slight modification to the algorithm. The derivative has to be modified and the rest of the algorithm of updating dual vectors $\boldsymbol{\alpha}$ corresponding to group A and group B alternatively remains.

We will show how to adapt the Landweber method for solving NMF [117] to solve the SVM problem.

Taking the gradient as given in equation 2.15, we can update the dual variables as follows:

$$\boldsymbol{\alpha}_A = \boldsymbol{\alpha}_A - \boldsymbol{\eta} \odot \boldsymbol{d} \tag{2.41}$$

$$\boldsymbol{\eta} = \frac{2}{\boldsymbol{K}_A \mathbf{1}} \tag{2.42}$$

$$\boldsymbol{\alpha}_A = \max(\mathbf{0}, \boldsymbol{\alpha}_A), \tag{2.43}$$

where $\boldsymbol{d}$ corresponds to derivative in 2.15 and $\max$ is applied to two vectors element-wise. Similarly, we update the dual variables corresponding to group B given by the vector $\boldsymbol{\alpha}_B$.

2.6.11 Power methods

Following the work in [99], we can increase the convergence speed of the algorithms by raising the multiplicative factor in the updates by a power greater than one. In the original work, it was applied to NMF as the Adaptive Overrelaxed NMF(ANMF) algorithm. Given a cost function $C(\alpha)$ over nonnegative α, we can define its positive and negative component of the derivative by $pd = \frac{\partial C(\boldsymbol{\alpha})^+}{\partial \boldsymbol{\alpha}_i}$, $nd = \frac{\partial C(\boldsymbol{\alpha})^-}{\partial \boldsymbol{\alpha}_i}$ respectively. The multiplicative updates can now be written as follows:

$$\boldsymbol{\alpha}_i = \boldsymbol{\alpha}_i(\frac{nd}{pd})^\gamma \tag{2.44}$$

where γ is a real number greater than one. This is applied to MUSIK and we get faster convergence as expected.

2.7 Experiments

In order to demonstrate practical applicability of theoretical properties proved in previous section, we test the above updates on two real world problems consisting of breast cancer dataset and aspect-angle dependent sonar signals from the UCI Repository [88]. They contain 683 and 208 labelled examples respectively. The breast cancer dataset was split into 80% and 20% for training and test sets respectively. The sonar dataset was equally divided into test and training sets. The support vectors were all initialized to one. Different kernels involving polynomial and radial basis functions were applied to the dataset. Misclassification rates on the test datasets after 750 iterations are shown in Table 2.1. They match previously reported error rates on this dataset [104]. The rate of convergence of support vectors is shown in Figure 2.11.

These results support our derivations and demonstrate that the algorithm can be used for training SVM with non-negative kernels. However, since the problem is convex and there exists a unique solution all correct algorithms will converge to the same solution and arrive at similar classification error rates.

In the following we test the MUSIK algorithm on a medium sized problem of USPS handwritten digits data set. It contains 7291 training examples. We consider the binary class problem with all the samples having digit '2' as labels belong to one and all the rest to another. This was compared with the state of the art multiplicative updates for NQP from [104].

For the experiments we have normalized the USPS dataset to lie in the range $[-1, 1]$ and smoothed it with a 2×2 Gaussian kernel. The non-negative kernel used for the

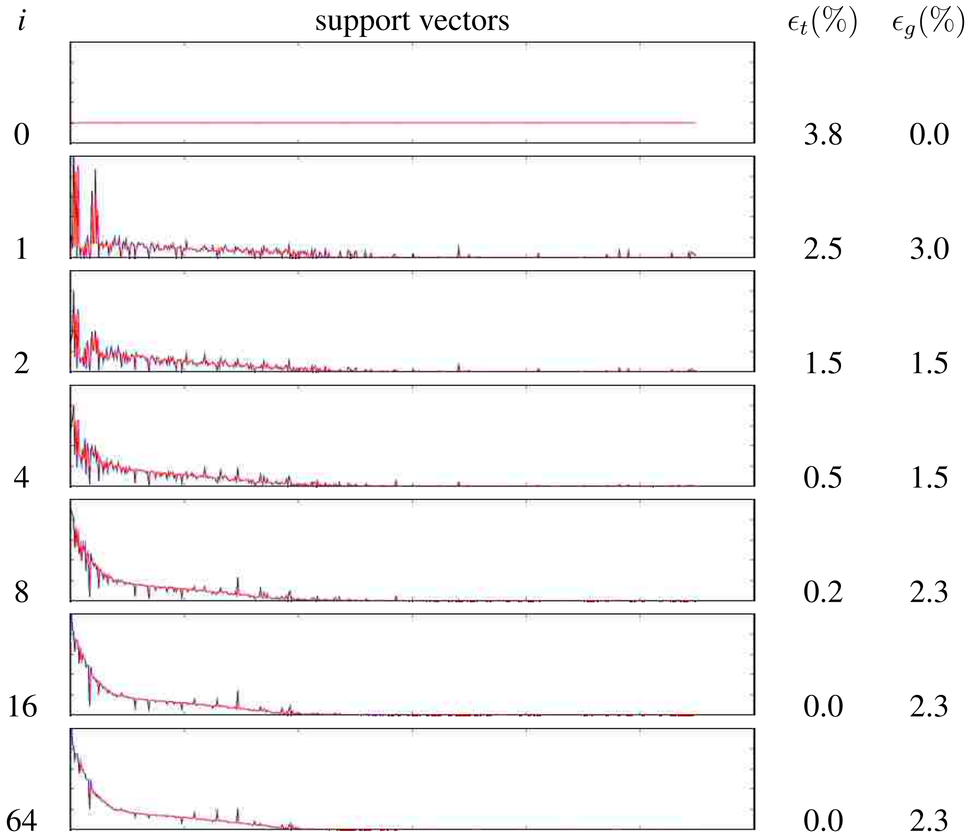


Figure 2.11: Rate of convergence of multiplicative updates for breast cancer dataset using RBF kernel with $\sigma = 3$. i is the iteration number, ϵ_t is the training error, ϵ_g is the test error. The support vectors have been rearranged for visualization into active and inactive.

experiment was the Gaussian radial basis function $K(\boldsymbol{x}_i, \boldsymbol{x}_j) = e^{\|\boldsymbol{x}_i - \boldsymbol{x}_j\|^2/2\sigma^2}$, with $\sigma = 6.0$. The slack penalty was set to 10.

Our algorithm is slightly faster per iteration due to an extra square root and multiplication per training pattern in the M^3 algorithm. We ignore that slight difference and plot the objective function per iteration of both algorithms on the USPS data set in Figure 2.12. The result agrees with the theoretically shown upper bound and the simulations from Figure 2.9.

Figure 2.13 shows misclassification rate on the training samples using MUSIK and M^3 algorithm.

To test the MUSIK algorithm with sign-insensitive kernel we generate an artificial dataset with 50 samples of each class. We compare convergence speed of M^3, MUSIK,

Kernel		Breast			Sonar		
		M^3	M	KA	M^3	M	KA
Poly	4	2.26	2.26	2.26	9.62	9.62	9.62
	6	3.76	3.76	3.76	10.58	10.58	10.58
Gaussian	3	2.26	2.26	2.26	11.53	11.53	11.53
	1	0.75	0.75	0.75	7.69	7.69	7.69

Table 2.1: Misclassification rates (%) on the breast cancer and sonar datasets after convergence of the M^3, MUSIK (M) and Kernel Adatron (KA) algorithms. Polynomial kernels of degree 4 and 6 and Gaussian kernels of σ 1 and 3 were used.

and MUSIK with semiNMF updates. Results are shown in Figure 2.14.

2.7.1 Proofs

In the following subsections we show how to prove the updates are non-increasing and bound the rate of convergence.

Multiplicative updates and convergence

We now derive the update rules for the dual variables $\boldsymbol{\alpha}$. Let us denote the matrices $\Phi(\boldsymbol{X}_A), \Phi(\boldsymbol{X}_B)$ by the matrices $\boldsymbol{M}, \boldsymbol{N}$ and the vectors $\boldsymbol{\alpha}_A, \boldsymbol{\alpha}_B$ by $\boldsymbol{u}, \boldsymbol{v}$ respectively.

The objective is now given by:

$$F(\boldsymbol{v}) = \frac{1}{2}\|\boldsymbol{M}\boldsymbol{u} - \boldsymbol{N}\boldsymbol{v}\|_2^2 - \sum_i \boldsymbol{v}_i \tag{2.45}$$

We define an auxillary function $G(\boldsymbol{v}, \boldsymbol{v}^t)$ with the properties that $G(\boldsymbol{v}, \boldsymbol{v}) = F(\boldsymbol{v})$ and $G(\boldsymbol{v}, \boldsymbol{v}^t) \geq F(\boldsymbol{v})$. The multiplicative update rule is found at each iteration by minimizing the auxiliary function :

$$\boldsymbol{v}^{t+1} = \arg\min_{\boldsymbol{v}} G(\boldsymbol{v}, \boldsymbol{v}^t) \tag{2.46}$$

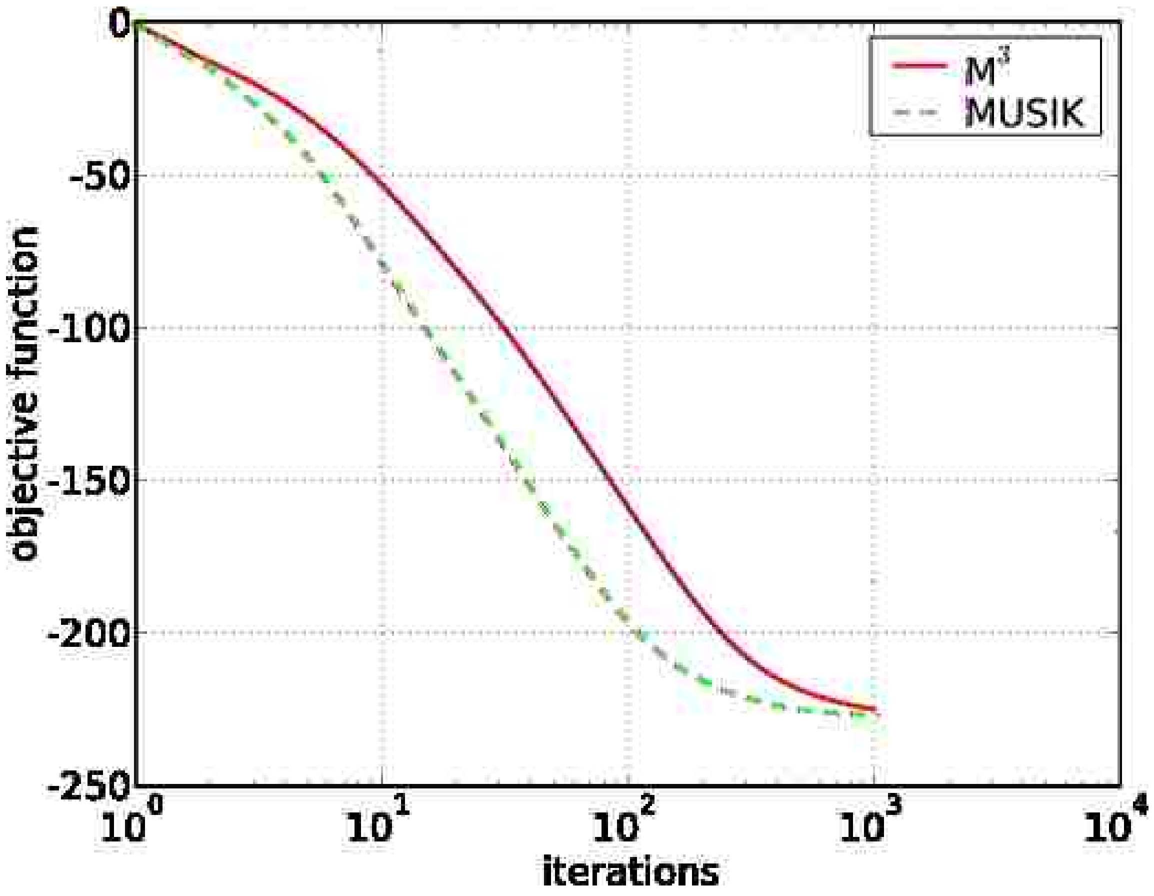


Figure 2.12: The objective function (2.14) versus training iteration number (log scale) on the USPS handwritten digits dataset for the M^3 and the MUSIK algorithms (down is better).

We know that this does not increase the objective function F ,as we have

$$F(\boldsymbol{v}^{t+1}) \leq G(\boldsymbol{v}^{t+1}, \boldsymbol{v}^t) \leq G(\boldsymbol{v}^t, \boldsymbol{v}^t) = F(\boldsymbol{v}^t) \tag{2.47}$$

Define G as follows:

$$\begin{aligned} G(\boldsymbol{v}, \boldsymbol{v}^t) &= F(\boldsymbol{v}^t) + (\boldsymbol{v} - \boldsymbol{v}^t)\nabla F(\boldsymbol{v}^t) \\ &+ \frac{1}{2}(\boldsymbol{v} - \boldsymbol{v}^t)L(\boldsymbol{v}^t)(\boldsymbol{v} - \boldsymbol{v}^t) \end{aligned} \tag{2.48}$$

where the diagonal matrix $L(\boldsymbol{v}^t)$ is defined as

$$L_{ab}(\boldsymbol{v}^t) = \delta_{ab}\frac{(K_{BA}^{-}\boldsymbol{u} + K_B^{+}\boldsymbol{v}^t)_a}{\boldsymbol{v}_a^t} \tag{2.49}$$

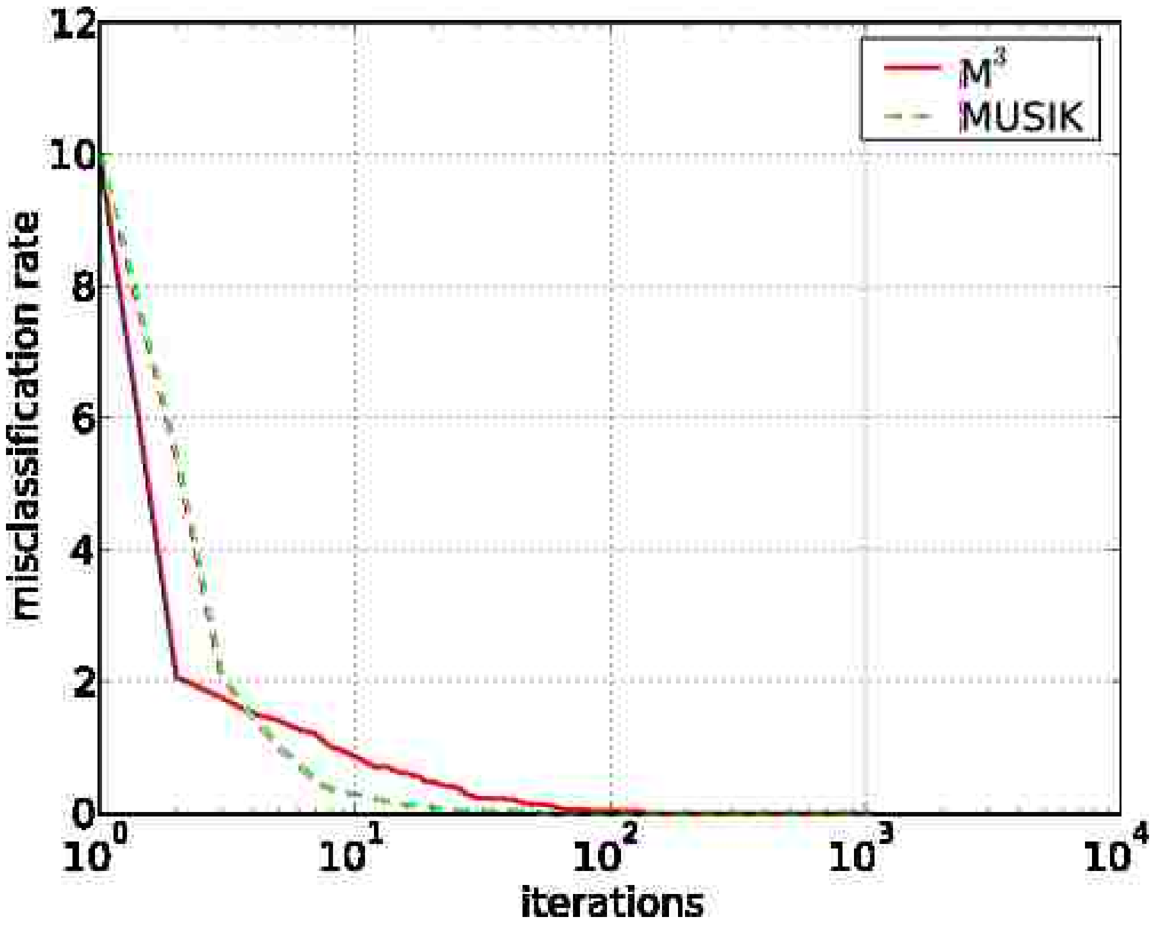


Figure 2.13: Percentage of the misclassification versus the training iteration number (log scale) on the USPS handwritten digits dataset for the M^3 and the MUSIK algorithms (down is better).

We see that $G(\boldsymbol{v}, \boldsymbol{v}) = F(\boldsymbol{v})$ trivially. The second property that $G(\boldsymbol{v}, \boldsymbol{v}^t) \geq F(\boldsymbol{v})$ is satisfied if

$$0 \leq (\boldsymbol{v} - \boldsymbol{v}^t)^T [L(\boldsymbol{v}^t) - K_B](\boldsymbol{v} - \boldsymbol{v}^t) \tag{2.50}$$

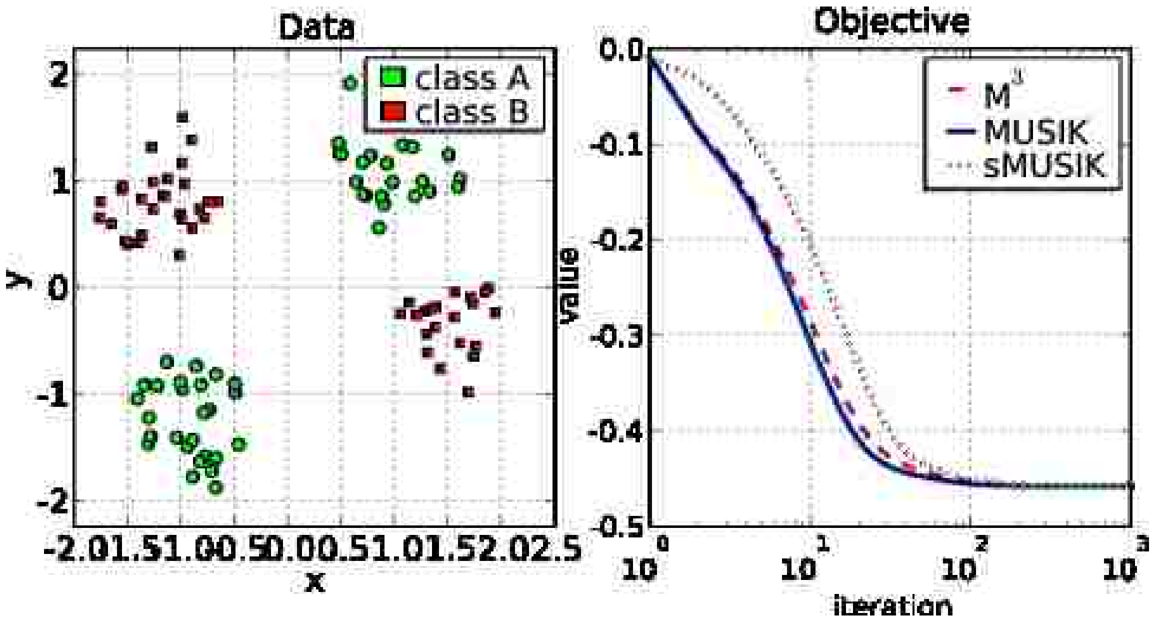


Figure 2.14: Convergence performance on a dataset containing negative and positive values (shown on the left side) with polynomial kernel of degree 3. MUSIK algorithm with semiNMF updates from Section 2.6.3 is called sMUSIK in the legend.

This can be split into three parts as follows:

$$L - K_B = L_1 + L_2 + L_3 \tag{2.51}$$

$$L_1 = diag(\frac{K_{BA}^{-}\boldsymbol{u}}{\boldsymbol{v}}) \tag{2.52}$$

$$L_2 = diag(\frac{(K_B^{+}\boldsymbol{v}^t)}{\boldsymbol{v}^t}) - K_B^{+} \tag{2.53}$$

$$L_3 = K_B^{-} \tag{2.54}$$

We have $L_1 + L_3$ to be positive semidefinite by construction in Section 2.6.3. If L_2 can be shown to positive semidefinite then the sum is positive semidefinite. L_2 is shown to be true using the argument in [67] which is as follows:

$$\boldsymbol{Q}_{ab}(\boldsymbol{v}^t) = \boldsymbol{v}_a^t (L_2(\boldsymbol{v}^t))_{ab} \boldsymbol{v}_b^t \tag{2.55}$$

$$\boldsymbol{\nu}^T \boldsymbol{Q} \boldsymbol{\nu} = \sum_{ab} \boldsymbol{\nu}_a \boldsymbol{Q}_{ab} \boldsymbol{\nu}_b \tag{2.56}$$

$$= \sum_{ab} (K_B^+)_{ab} \boldsymbol{v}_a^t \boldsymbol{v}_b^t [\frac{1}{2}\boldsymbol{\nu}_a^2 + \frac{1}{2}\boldsymbol{\nu}_b^2 - \boldsymbol{\nu}_a \boldsymbol{\nu}_b] \tag{2.57}$$

$$= \frac{1}{2} \sum_{ab} (K_B^+)_{ab} \boldsymbol{v}_a^t \boldsymbol{v}_b^t (\boldsymbol{\nu}_a - \boldsymbol{\nu}_b)^2 \tag{2.58}$$

$$\geq 0 \tag{2.59}$$

We select the minimum of G. This is found by setting the gradient of G to zero.

$$\begin{aligned} \boldsymbol{v}^{t+1} &= \boldsymbol{v}^t - \frac{\boldsymbol{v}^t}{K_{BA}^- \boldsymbol{u} + K_B^+ \boldsymbol{v}} \odot (K_B \boldsymbol{v}^t - K_{BA} \boldsymbol{u} - \mathbf{1}) \\ &= \boldsymbol{v}^t \odot \frac{K_{BA}^+ \boldsymbol{u} + K_B^- \boldsymbol{v} + \mathbf{1}}{K_{BA}^- \boldsymbol{u} + K_B^+ \boldsymbol{v}} \end{aligned} \tag{2.60}$$

This is the update rule for $\boldsymbol{v}$ and similarly we can derive the update rule for $\boldsymbol{u}$.

Convergence rate

Let the fixed point be $\boldsymbol{\alpha}^*$. Let us denote $K(\boldsymbol{X}_A, \boldsymbol{X}_A)\boldsymbol{\alpha}_A^*$ by $\boldsymbol{z}^+$ and $K(\boldsymbol{X}_A, \boldsymbol{X}_B)\boldsymbol{\alpha}_B^*$ by $\boldsymbol{z}^-$. If we choose an ith non-support vector coefficient from $\boldsymbol{\alpha}_A$, then we have $\boldsymbol{z}_i^+ - \boldsymbol{z}_i^- \geq 1$.

Let the multiplicative factor be denoted by γ_i. We then have:

$$\frac{1}{\gamma_i} = \frac{\boldsymbol{z}_i^+}{\boldsymbol{z}_i^- + 1} \tag{2.61}$$

$$= 1 + \frac{\boldsymbol{z}_i^+ - \boldsymbol{z}_i^- - 1}{\boldsymbol{z}_i^- + 1} \tag{2.62}$$

$$\geq 1 + \frac{K(\boldsymbol{x}_i, \boldsymbol{w}) - 1}{\boldsymbol{z}_i^+} \tag{2.63}$$

where we have $\boldsymbol{w} = \sum_i \alpha_i^* x_i y_i$ is the normal vector to the maximum margin hyperplane.

We used the following:

$$\begin{aligned}\boldsymbol{z}_i^+ - \boldsymbol{z}_i^- &= \sum_{j\in A} K(\boldsymbol{x}_i, \boldsymbol{x}_j)\boldsymbol{\alpha}_j^* - \sum_{k\in B} K(\boldsymbol{x}_i, \boldsymbol{x}_k)\boldsymbol{\alpha}_k^* \\ &= K(\boldsymbol{x}_i, \boldsymbol{w}) \qquad (2.64)\end{aligned}$$

We now obtain a bound on the denominator:

$$\begin{aligned}\boldsymbol{z}_i^+ &= \sum_{j\in A} K(\boldsymbol{x}_i, \boldsymbol{x}_j)\boldsymbol{\alpha}_j^* \qquad (2.65)\\ &\le \max_{k\in A} K(\boldsymbol{x}_i, \boldsymbol{x}_k) \sum_{j\in A} \boldsymbol{\alpha}_j^* \qquad (2.66)\\ &\le \sqrt{K(\boldsymbol{x}_i, \boldsymbol{x}_i)} \max_{k\in A} \sqrt{K(\boldsymbol{x}_k, \boldsymbol{x}_k)} K(\boldsymbol{w}, \boldsymbol{w}) \qquad (2.67)\end{aligned}$$

We used the Cauchy-Schwartz inequality for kernels and an upper bound for the sum of vector α_A^*.

We do a similar analysis by perturbing an ith non-support vector coefficient from group B. Combining the analysis, we have a lower bound as follows:

$$\frac{1}{\gamma_i} \ge 1 + \frac{K(\boldsymbol{x}_i, \boldsymbol{w}) - 1}{\sqrt{K(\boldsymbol{x}_i, \boldsymbol{x}_i)} \max_k \sqrt{K(\boldsymbol{x}_k, \boldsymbol{x}_k)} K(\boldsymbol{w}, \boldsymbol{w})}$$

2.8 Conclusions and Future Work

We showed a reduction from TNNLS to a single-class SVM. This gave us insight into the connection between nonnegativity and sparsity and further enabled us to propose an efficient algorithm to solve the TNNLS problem. The new algorithm is simple to implement and involves combining an SVM solver (such as OCAS) with an exact NNLS solver. We showed its application to random problems, as well as to two real examples of dose calculation in radiation therapy. Also, we showed that nonnegativity corresponds to sparsity depending on how many elements lie on the maximum-margin hyperplane. This explains the connection between nonnegativity and sparsity posed in the work on compressive sensing by [89]. Also, the running time depended on the spread of data and we would like to explore when it makes sense to use the reduction and how to set the parameters in the

SVM to best trade-off between the approximate SVM solver and an exact NNLS solver. Our approach seems to be more suitable for “fat” matrices, where the number of rows and columns are similar. The NNLS solver used in [76] used advanced techniques such as multi-threading and vector commands, and has only floating point precision, while our TNNLS solver has double precision. A similar speed up is conceivable if solvers used in this paper were implemented in a similar manner. Also, we adapted a dual SVM solver to solve the NNLS problem and similar to the primal SVM solver, it gave us a speedup over the state-of-the-art algorithms.

We have derived simple multiplicative update rules for solving the maximum-margin classifier problem in SVMs. No additional parameter tuning is required and the convergence is guaranteed. In practice the method converges within a few iterations. Extensions to multiple kernel learning are left as future work. The updates could also be used as part of a subset method which could potentially speed up MUSIK algorithm. MUSIK shares the utility of M^3 algorithm in that it is easy to implement in higher-level languages like MATLAB with application to small datasets. It also shares the drawback of M^3 in its inability to directly set a variable to zero. However, we have shown MUSIK to have an asymptotically faster rate of convergence compared to M^3 algorithm and we believe this provides a motivation for further research in multiplicative updates for support vector machines. Also the derivation was constructed in such a way that it highlights the connection between SVM and NMF. We also show a connection to the Kernel Adatron algorithm. Sequential updates similar to ones in KA have been used to solve the NMF problem and it would be interesting if heuristics used in KA can be imported to solve NMF-type problems. Since multiplicative updates emerge in different settings and algorithms it might be interesting to find the pattern of when such updates are possible and how to automatically derive them. Our presentation of NMF and SVM correspondence can be considered a step towards this direction.

Chapter 3

Sparsity — Matrix factorization

Matrix factorization arises in a wide range of application domains and is useful for extracting the latent features in the dataset (Figure 3.1). In particular, we consider matrix factorizations which impose the following requirements:

- nonnegativity
- low-rankedness
- sparsity

Nonnegativity is a natural constraint when modeling data with physical constraints such as chemical concentrations in solutions, pixel intensities in images and radiation dosages for cancer treatment. Low-rankedness is useful for learning a lower dimensionality representation. Sparsity is useful for modeling the conciseness of the representation or that of the latent features. Imposing all these requirements on our matrix factorization leads to the sparse nonnegative matrix factorization (SNMF) problem.

SNMF enjoys quite a few formulations [7, 48, 47, 43, 85, 58, 92, 93] with successful applications to single-channel speech separation [100] and micro-array data analysis [58, 92].

However, algorithms [48, 43] for solving SNMF which utilize the mixed norm of L_1/L_2 as their sparsity measure are slow and do not scale well to large datasets. Thus, we develop an efficient algorithm to solve this problem and has the following ingredients:

- A theoretically efficient projection operator ($O(m \log m)$) to enforce the user-defined sparsity where m is the dimensionality of the feature vector as opposed to the previous approach [48].
- Novel sequential updates which provide the bulk of our speedup compared to the previously employed batch methods [48, 43].

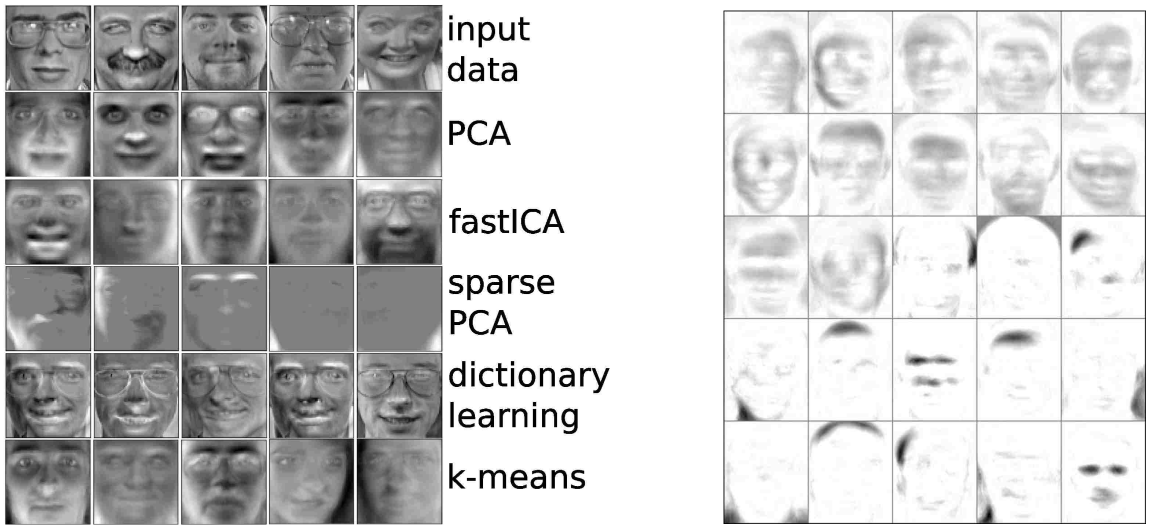


Figure 3.1: (Left) Features learned from the ORL dataset (Scikit-learn package was used) with various matrix factorization methods such as principal component analysis (PCA), independent component analysis (ICA), and dictionary learning. The relative merit of the various matrix factorizations depends on both the signal domain and the target application of interest. (Right) Features learned under the sparse NMF formulation where roughly half the features were constrained to lie in the interval $[0.2, 0.4]$ and the rest are fixed to sparsity value 0.7. This illustrates the flexibility that the user has in fine tuning the feature sparsity based on prior domain knowledge. White pixels in this figure correspond to the zeros in the features.

3.1 Preliminaries and Previous Work

In this section, we give an introduction to the nonnegative matrix factorization (NMF) and SNMF problems. Also, we discuss some widely used algorithms from the literature to solve them.

Both these problems share the following problem and solution structure. At a high-level, given a nonnegative matrix $\mathbf{X}$ of size $m \times n$, we want to approximate it with a product of two nonnegative matrices $\mathbf{W}, \mathbf{H}$ of sizes $m \times r$ and $r \times n$, respectively:

$$\mathbf{X} \approx \mathbf{W}\mathbf{H}. \tag{3.1}$$

The nonnegative constraint on matrix $\mathbf{H}$ makes the representation a conical combination of features given by the columns of matrix $\mathbf{W}$. In particular, NMF can result in sparse representations, or a parts-based representation, unlike other factorization techniques such as principal component analysis (PCA) and vector quantization (VQ). A common theme in the algorithms proposed for solving these problems is the use of alternating updates to the matrix factors, which is natural because the objective function to be minimized is convex in $\mathbf{W}$ and in $\mathbf{H}$, separately, but not in both together. Much effort has been focused on optimizing the efficiency of the core step of updating one of $\mathbf{W}, \mathbf{H}$ while the other stays fixed.

3.1.1 Nonnegative Matrix Factorization

Factoring a matrix, all of whose entries are nonnegative, as a product of two low-rank nonnegative factors is a fundamental algorithmic challenge. This has arisen naturally in diverse areas such as image analysis [65], micro-array data analysis [58], document clustering [114], chemometrics [63], information retrieval [45] and biology applications [13]. For further applications, see the references in the following papers [1, 25].

We will consider the following version of the NMF problem, which measures the reconstruction error using the Frobenius norm [66]:

$$\min_{\mathbf{W},\mathbf{H}} \frac{1}{2}\|\mathbf{X} - \mathbf{W}\mathbf{H}\|_F^2 \text{ s.t. } \mathbf{W} \geq \mathbf{0},\ \mathbf{H} \geq \mathbf{0},\ \|\boldsymbol{W}_j\|_2 = 1,\ \forall j \in \{1, \cdots, r\} \tag{3.2}$$

where $\geq$ is element-wise. We use subscripts to denote column elements. Simple multiplicative updates were proposed in [66] to solve the NMF problem. This is attractive for the following reasons:

- Unlike additive gradient descent methods, there is no arbitrary learning rate parameter that needs to be set.

- The nonnegativity constraint is satisfied automatically, without any additional projection step.
- The objective function converges to a limit point and the values are non-increasing across the updates [66].

Algorithm 2 is an example of the kind of multiplicative update procedure used in the literature [66]. The algorithm alternates between updating the matrices $\mathbf{W}$ and $\mathbf{H}$ (we have only shown the updates for $\mathbf{H}$—those for $\mathbf{W}$ are analogous).

Algorithm 2 nnls-mult($\mathbf{X}, \mathbf{W}, \mathbf{H}$)

```
1: repeat
2:     $\mathbf{H} = \mathbf{H} \odot \frac{\mathbf{W}^\top \mathbf{X}}{\mathbf{W}^\top \mathbf{W} \mathbf{H}}$.
3: until convergence
4: Output: Matrix $\mathbf{H}$.
```

Here, $\odot$ indicates element-wise (Hadamard) product and matrix division is also element-wise. To remove the scaling ambiguity, the norm of columns of matrix $\mathbf{W}$ are set to unity. Also, a small constant, say 10^{-9}, is added to the denominator in the updates to avoid division by zero.

Besides multiplicative updates, other algorithms have been proposed to solve the NMF problem based on projected gradient [75], block pivoting [60], sequential constrained optimization [23] and greedy coordinate-descent [49].

3.1.2 Sparse Nonnegative Matrix Factorization

The nonnegative decomposition is in general not unique [34]. Furthermore, the features may not be parts-based if the data resides well inside the positive orthant. To address these issues, sparseness constraints have been imposed on the NMF problem.

Sparse NMF can be formulated in many different ways. From a user point of view, we can split them into two classes of formulations: explicit and implicit. In explicit versions of SNMF [48, 43], one can set the sparsities of the matrix factors $\boldsymbol{W}, \boldsymbol{H}$ directly. On the other hand, in implicit versions of SNMF [58, 92], the sparsity is controlled via a regularization parameter and is often hard to tune to specified sparsity values a priori. However, the algorithms for implicit versions tend to be faster compared to the explicit versions of SNMF.

In this paper, we consider the explicit sparse NMF formulation proposed by Hoyer [48]. To make the presentation easier to follow, we first consider the case where the sparsity is imposed on one of the matrix factors, namely the feature matrix $\boldsymbol{W}$—the analysis for the symmetric case where the sparsity is instead set on the other matrix factor $\boldsymbol{H}$ is analogous. The case where sparsity requirements are imposed on both the matrix factors is dealt with in the Appendix. The sparse NMF problem formulated [48] with sparsity on matrix $\boldsymbol{W}$ is as follows:

$$\begin{aligned} \min_{\mathbf{W},\mathbf{H}} f(\boldsymbol{W},\boldsymbol{H}) =& \frac{1}{2}\|\mathbf{X}-\mathbf{W}\mathbf{H}\|_F^2 \text{ s.t. } \mathbf{W}\geq\mathbf{0},\mathbf{H}\geq\mathbf{0}, \\ & \|\boldsymbol{W}_j\|_2 = 1,\ \text{sp}(\boldsymbol{W}_j)=\alpha,\ \forall j\in\{1,\cdots,r\} \end{aligned} \tag{3.3}$$

Sparsity measure for a d-dimensional vector $\boldsymbol{x}$ is given by:

$$\text{sp}(\boldsymbol{x}) = \frac{\sqrt{d}-\|x\|_1/\|x\|_2}{\sqrt{d}-1} \tag{3.4}$$

The sparsity measure (3.4) defined above has many appealing qualities. Some of which are as follows:

- The measure closely models the intuitive notion of sparsity as captured by the L_0 norm. So, it easy for the user to specify sparsity constraints from prior knowledge of the application domain.
- Simultaneously, it is able to avoid the pitfalls associated with directly optimizing the L_0 norm. Desirable properties for sparsity measures have been previously explored [51] and it satisfies all of these properties for our problem formulation. The properties can be briefly summarized as: (a) Robin Hood — Spreading the energy from larger coordinates to smaller ones decreases sparsity, (b) Scaling — Sparsity

is invariant to scaling, (c) Rising tide — Adding a constant to the coordinates decreases sparsity, (d) Cloning — Sparsity is invariant to cloning, (e) Bill Gates — One big coordinate can increase sparsity, (f) Babies — coordinates with zeros increase sparsity.

- The above sparsity measure enables one to limit the sparsity for each feature to lie in a given range by changing the equality constraints in the SNMF formulation (3.3) to inequality constraints [43]. This could be useful in scenarios like fMRI brain analysis, where one would like to model the prior knowledge such as sizes of artifacts are different from that of the brain signals. A sample illustration on a face dataset is shown in Figure 3.1 (Right). The features are now evenly split into two groups of local and global features by choosing two different intervals of sparsity.

A gradient descent-based algorithm called Nonnegative Matrix Factorization with Sparseness Constraints (NMFSC) to solve SNMF was proposed [48]. Multiplicative updates were used for optimizing the matrix factor which did not have sparsity constraints specified. In [43] two new algorithms were proposed which also solved this problem by sequential cone programming and utilized general purpose solvers like MOSEK (`http://www.mosek.com`). We will consider the faster one of these called tangent-plane constraint (TPC) algorithm. However, both these algorithms, namely NMFSC and TPC, solve for the whole matrix of coefficients at once. In contrast, we propose a block coordinate-descent strategy which considers a sequence of vector problems where each one can be solved in closed form efficiently.

3.2 The Sequential Sparse NMF Algorithm

We present our algorithm which we call **S**equential **S**parse **NMF** (**SSNMF**) to solve the SNMF problem as follows:

First, we consider a problem of special form which is the building block (Algorithm 3) of our SSNMF algorithm and give an efficient, as well as exact, algorithm to solve it.

Second, we describe our sequential approach (Algorithm 4) to solve the subproblem of SNMF. This uses the routine we developed in the previous step. Finally, we combine our routines developed in the previous two steps along with standard solvers (for instance Algorithm 2) to complete the SSNMF Algorithm (Algorithm 5).

3.2.1 Sparse-opt

Sparse-opt routine solves the following subproblem which arises when solving problem (3.3):

$$\max_{\boldsymbol{y}\geq 0} \boldsymbol{b}^\top \boldsymbol{y} \text{ s.t. } \|\boldsymbol{y}\|_1 = k, \|\boldsymbol{y}\|_2 = 1 \tag{3.5}$$

where vector $\boldsymbol{b}$ is of size m. This problem has been previously considered [48], and an algorithm to solve it was proposed which we will henceforth refer to as the Projection-Hoyer. Similar projection problems have been recently considered in the literature and solved efficiently [36, 20].

Observation 1. *For any i, j, we have that if $b_i \geq b_j$, then $y_i \geq y_j$.*

Let us first consider the case when the vector $\boldsymbol{b}$ is sorted. Then by the previous observation, we have a transition point p that separates the zeros of the solution vector from the rest.

Observation 2. *By applying the Cauchy-Schwarz inequality on $\mathbf{y}$ and the all ones vector, we get $p \geq k^2$.*

The Lagrangian of the problem (3.5) is :

$$L(\boldsymbol{y}, \mu, \lambda, \boldsymbol{\gamma}) = \boldsymbol{b}^\top \boldsymbol{y} + \mu\left(\sum_{i=1}^{m} y_i - k\right) + \frac{\lambda}{2}\left(\sum_{i=1}^{m} y_i^2 - 1\right) + \boldsymbol{\gamma}^\top \boldsymbol{y}$$

Setting the partial derivatives of the Lagrangian to zero, we get by observation 1:

$$\sum_{i=1}^{m} y_i = k, \sum_{i=1}^{m} y_i^2 = 1$$

$$b_i + \mu(p) + \lambda(p) y_i = 0, \forall i \in \{1, 2, \cdots, p\}$$

$$\gamma_i = 0, \forall i \in \{1, \cdots, p\}$$

$$y_i = 0, \forall i \in \{p+1, \cdots, m\}$$

where we account for the dependence of the Lagrange parameters λ, μ, and γ on the transition point p. We compute the objective value of problem (3.5) for all transition points p in the range from k^2 to m and select the one with the highest value. In the case, where the vector $\boldsymbol{b}$ is not sorted, we just simply sort it and note down the sorting permutation vector. The complete algorithm is given in Algorithm 3. The dominant contribution to the running time of Algorithm 3 is the sorting of vector $\boldsymbol{b}$ and therefore can be implemented in $O(m \log m)$ time[1]. Contrast this with the running time of Projection-Hoyer whose worst case is $O(m^2)$ [48, 107].

3.2.2 Sequential Approach —Block Coordinate Descent

Previous approaches for solving SNMF [48, 43] use batch methods to solve for sparsity constraints. That is, the whole matrix is updated at once and projected to satisfy the constraints. We take a different approach of updating a column vector at a time. This gives us the benefit of being able to solve the subproblem (column) efficiently and exactly. Subsequent updates can benefit from the newly updated columns resulting in faster convergence as seen in the experiments.

In particular, consider the optimization problem (3.3) for a column j of the matrix $\mathbf{W}$ while fixing the rest of the elements of matrices $\mathbf{W}, \mathbf{H}$:

$$\min_{\mathbf{W}_j \geq \mathbf{0}} \tilde{f}(\mathbf{W}_j) = \frac{1}{2} g \|\mathbf{W}_j\|_2^2 + \boldsymbol{u}^\top \mathbf{W}_j \text{ s.t. } \|\mathbf{W}_j\|_2 = 1, \|\mathbf{W}_j\|_1 = k$$

[1]This can be further reduced to linear time by noting that we do not need to fully sort the input in order to find $p*$.

Algorithm 3 Sparse-opt($\boldsymbol{b}, k$)

```
1:  Set $\boldsymbol{a} = \text{sort}(\boldsymbol{b})$ and $p^* = m$. Get a mapping $\pi$ such that $a_i = b_{\pi(i)}$ and $a_j \geq a_{j+1}$ for
    all valid $i, j$.
2:  Compute values of $\mu(p), \lambda(p)$ as follows:
3:  for $p = \lceil k^2 \rceil$ to $m$ do
4:      $\lambda(p) = -\sqrt{\frac{p\sum_{i=1}^{p} a_i^2 - \left(\sum_{i=1}^{p} a_i\right)^2}{(p-k^2)}}$
5:      $\mu(p) = -\frac{\sum_{i=1}^{p} a_i}{p} - \frac{k}{p}\lambda(p)$
6:      if $a(p) < -\mu(p)$ then
7:          $p^* = p - 1$
8:          break
9:      end if
10: end for
11: Set $x_i = -\frac{a_i + \mu(p^*)}{\lambda(p^*)}, \forall i \in \{1, \cdots, p^*\}$ and to zero otherwise.
12: Output: Solution vector $\boldsymbol{y}$ where $y_{\pi(i)} = x_i$.
```

where $g = \boldsymbol{H}_j^\top \boldsymbol{H}_j$ and $u = -\boldsymbol{X}\boldsymbol{H}_j^\top + \sum_{i \neq j} \boldsymbol{W}_i (\boldsymbol{H}\boldsymbol{H}^\top)_{ij}$. This reduces to the problem (3.5) for which we have proposed an exact algorithm (Algorithm 3). We update the columns of the matrix factor $\mathbf{W}$ sequentially as shown in Algorithm 4. We call it sequential for we update the columns one at a time. Note that this approach can be seen as an instance of block coordinate descent methods by mapping features to blocks and the Sparse-opt projection operator to a descent step.

3.2.3 SSNMF Algorithm for Sparse NMF

We are now in a position to present our complete Sequential Sparse NMF (SSNMF) algorithm. By combining Algorithms 2, 3 and 4, we obtain SSNMF (Algorithm 5).

Algorithm 4 sequential-pass($\mathbf{X}, \mathbf{W}, \mathbf{H}$)

```
 1: $\mathbf{C} = -\mathbf{X}\mathbf{H}^\top + \mathbf{W}\mathbf{H}\mathbf{H}^\top$
 2: $\mathbf{G} = \mathbf{H}\mathbf{H}^\top$
 3: **repeat**
 4:   **for** j = $1$ to $r$ (randomly) **do**
 5:     $\mathbf{U}_j = \mathbf{C}_j - \mathbf{W}_j G_{jj}$
 6:     $\boldsymbol{t}$ = Sparse-opt($-\mathbf{U}_j, k$).
 7:     $\mathbf{C} = \mathbf{C} + (\boldsymbol{t} - \mathbf{W}_j)\, \boldsymbol{G}_j^\top$
 8:     $\mathbf{W}_j = \boldsymbol{t}$.
 9:   **end for**
10: **until** convergence
11: Output: Matrix $\mathbf{W}$.
```

Algorithm 5 ssnmf($\mathbf{X}, \mathbf{W}, \mathbf{H}$)

```
1: **repeat**
2:   $\mathbf{W}$ = sequential-pass($\mathbf{X}, \mathbf{W}, \mathbf{H}$)
3:   $\mathbf{H}$ = nnls-mult($\mathbf{X}, \mathbf{W}, \mathbf{H}$)
4: **until** convergence
5: Output: Matrices $\mathbf{W}, \mathbf{H}$.
```

3.3 Implementation Issues

For clarity of exposition, we presented the plain vanilla version of our SSNMF Algorithm 5. We now describe some of the actual implementation details.

- Initialization: Generate a positive random vector $\boldsymbol{v}$ of size m and obtain $\boldsymbol{z}$ = Sparse-opt$(\boldsymbol{v}, k)$ where $k = \sqrt{m} - \alpha\sqrt{m-1}$ (from equation (3.4)). Use the solution $\boldsymbol{z}$ and its random permutations to initialize matrix $\mathbf{W}$. Initialize the matrix $\mathbf{H}$ to uniform random entries in $[0, 1]$.

- Incorporating faster solvers: We use multiplicative updates for a fair comparison with NMFSC and TPC. However, we can use other NNLS solvers [75, 60, 23, 49] to solve for matrix $\boldsymbol{H}$. Empirical results (not reported here) show that this further speeds up the SSNMF algorithm.

- Termination: In our experiments, we fix the number of alternate updates or equivalently the number of times we update matrix $\boldsymbol{W}$. Other approaches include specifying total running time, relative change in objective value between iterations or approximate satisfaction of KKT conditions.

- Sparsity constraints: We have primarly considered the sparse NMF model as formulated in [48]. This has been generalized in [43] by relaxing the sparsity constraints to lie in user-defined intervals. Note that, we can handle the relaxed formulation [43] by making a trivial change to Algorithm 4.

3.4 Experiments and Discussion

In this section, we compare the performance of our algorithm with the state-of-the-art NMFSC and TPC algorithms [48, 43]. Running times for the algorithms are presented when applied to one synthetic and three real-world datasets. Experiments report reconstruction error ($||\mathbf{X} - \mathbf{W}\mathbf{H}||_F$) instead of objective value for convenience of display. For all experiments on the datasets, we ensure that our final reconstruction error is always better than that of the other two algorithms. Our algorithm was implemented in MATLAB (`http://www.mathworks.com`) similar to NMFSC and TPC. All of our experiments were run on a 3.2Ghz Intel machine with 24GB of RAM and the number of threads set to one.

3.4.1 Datasets

For comparing the performance of SSNMF with NMFSC and TPC, we consider the following synthetic and three real-world datasets :

- Synthetic: 200 images of size 9×9 as provided in [43] (in their code implementation).

- CBCL: Face dataset of 2429 images of size 19×19 and can be obtained at `http://cbcl.mit.edu/cbcl/software-datasets/FaceData2.html`.
- ORL: Face dataset that consists of 400 images of size 112×92 and can be obtained at `cl.cam.ac.uk/research/dtg/attarchive/facedatabase.html`.
- sMRI: Structural MRI scans of 269 subjects taken at the John Hopkins University were obtained. The scans were taken on a single 1.5T scanner with the imaging parameters set to 35mm TR, 5ms TE, matrix size of 256×256. We segment these images into gray matter, white matter and cerebral spinal fluid images, using the software program SPM5 (`http://www.fil.ion.ucl.ac.uk/spm/software/spm5/`), followed by spatial smoothing with a Gaussian kernel of $10 \times 10 \times 10$ mm. This results in images which are of size $105 \times 127 \times 46$.

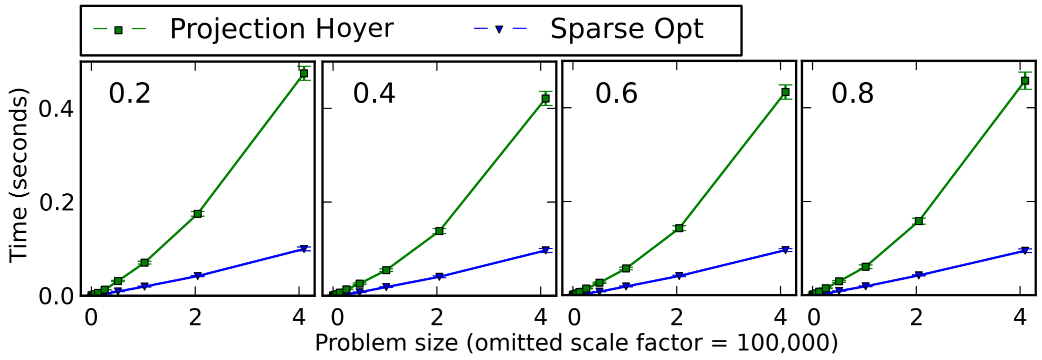


Figure 3.2: Mean running times for Sparse-opt and the Projection-Hoyer are presented for random problems. The x-axis plots the dimension of the problem while the y-axis has the running time in seconds. Each of the subfigures corresponds to a single sparsity value in $\{0.2, 0.4, 0.6, 0.8\}$. Each datapoint corresponds to the mean running time averaged over 40 runs for random problems of the same fixed dimension.

3.4.2 Comparing Performances of Core Updates

We compare our Sparse-opt (Algorithm 3) routine with the competing

Projection-Hoyer [48]. In particular, we generate 40 random problems for each sparsity constraint in $\{0.2, 0.4, 0.6, 0.8\}$ and a fixed problem size. The problems are of size $2^i \times 100$ where i takes integer values from 0 to 12. Input coefficients are generated by drawing samples uniformly at random from $[0, 1]$. The mean values of the running times for Sparse-opt and the Projection-Hoyer for each dimension and corresponding sparsity value are plotted in Figure 3.2.

We compare SSNMF with SSNMF+Proj on the CBCL dataset. The algorithms were run with rank set to 49. The running times are shown in Figure 3.3. We see that in low-

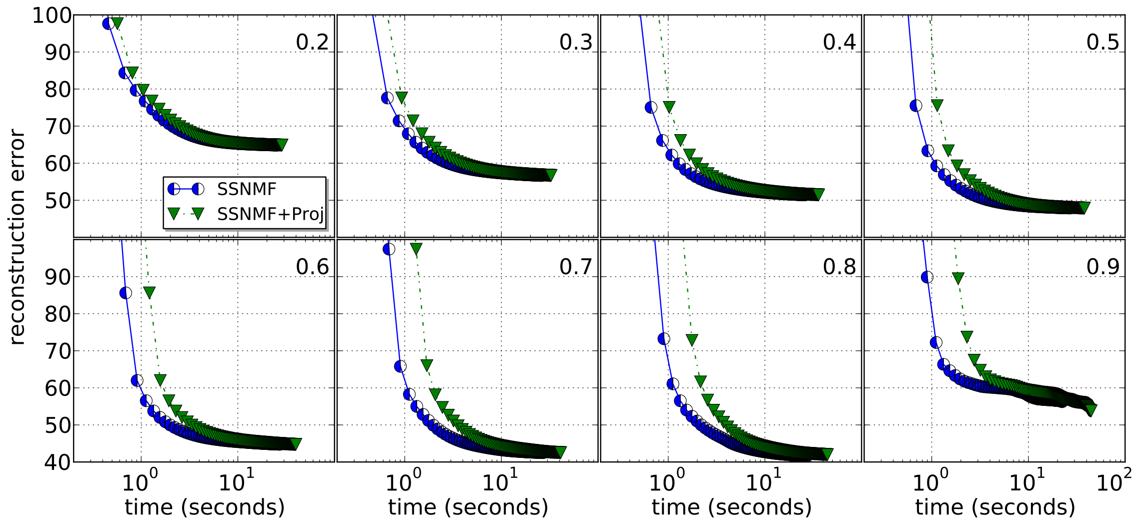


Figure 3.3: Running times for SSNMF and SSNMF+Proj algorithms for the CBCL face dataset with rank set to 49 and sparsity values ranging from 0.2 to 0.9

dimensional datasets, the difference in running times are very small.

3.4.3 Comparing Overall Performances

SSNMF versus NMFSC and TPC: We plot the performance of SSNMF against NMFSC and TPC on the synthetic dataset provided in [43] in Figure 3.4. We used the default settings for both NMFSC and TPC using the software provided by the authors. Our experience with TPC was not encouraging on bigger datasets and hence we show its performance only on the synthetic dataset. It is possible that the performance of TPC can be improved by changing the default settings but we found it non-trivial to do so.

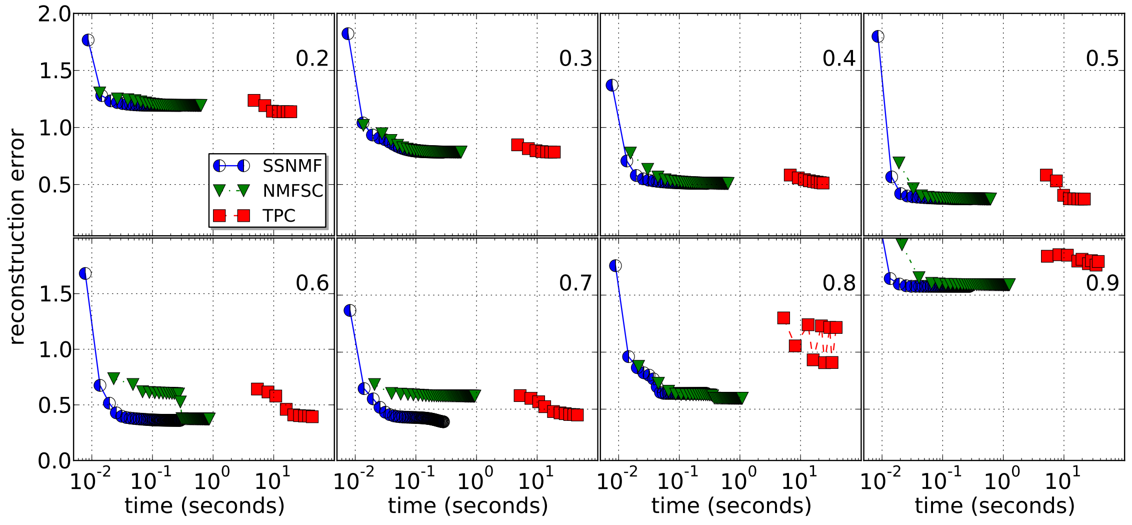


Figure 3.4: Running times for SSNMF and NMFSC and TPC algorithms on the synthetic dataset where the sparsity values range from 0.2 to 0.8 and number of features is 5. Note that SSNMF and NMFSC are over an order of magnitude faster than TPC.

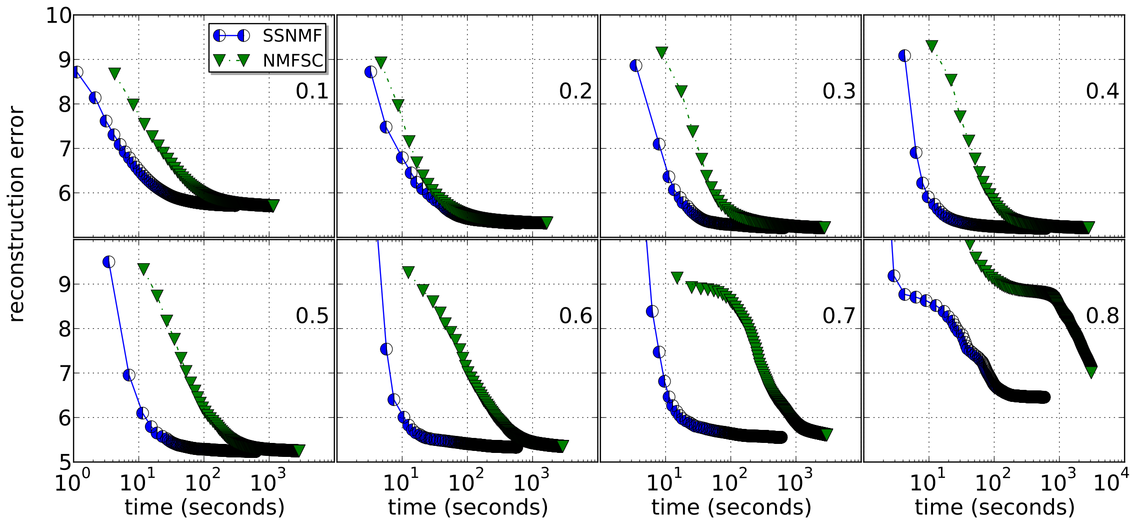


Figure 3.5: Convergence plots for the ORL dataset with sparsity from $[0.1, 0.8]$ for the NMFSC and SSNMF algorithms. Note that we are an order of magnitude faster, especially when the sparsity is higher.

SSNMF versus NMFSC: To ensure fairness, we removed logging information from NMFSC code [48] and only computed the objective for equivalent number of matrix updates as SSNMF. We do not plot the objective values at the first iteration for convenience of display. However, they are the same for both algorithms because of the shared initial-

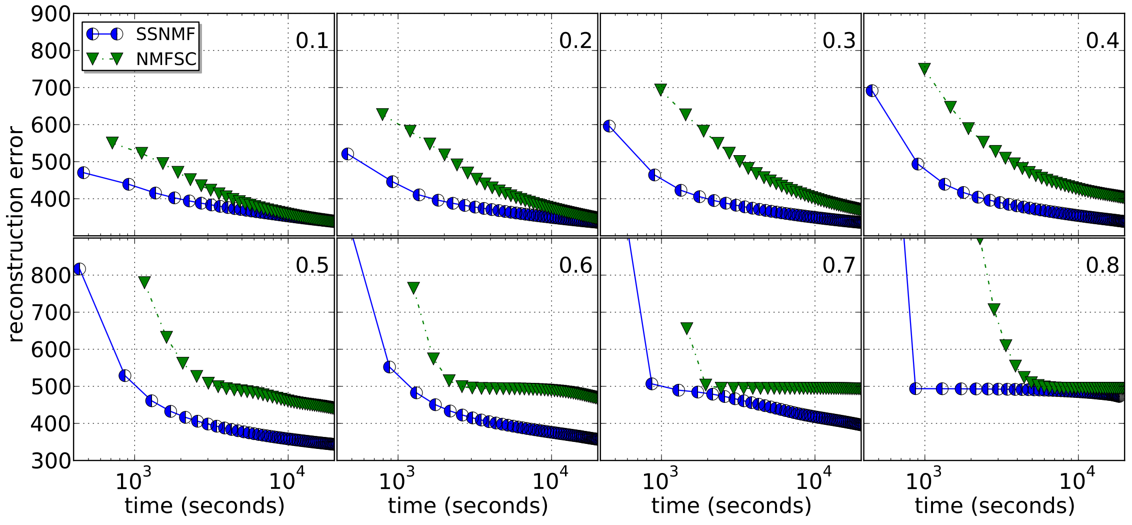


Figure 3.6: Running times for SSNMF and NMFSC algorithms for the sMRI dataset with rank set to 40 and sparsity values of α from 0.1 to 0.8. Note that for higher sparsity values we converged to a lower reconstruction error and are also noticeably faster than the NMFSC algorithm.

ization . We ran the SSNMF and NMFSC on the ORL face dataset. The rank was fixed at 25 in both the algorithms. Also, the plots of running times versus objective values are shown in Figure 3.5 corresponding to sparsity values ranging from 0.1 to 0.7. Additionally, we ran our SSNMF algorithm and NMFSC algorithm on a large-scale dataset consisting of the structural MRI images by setting the rank to 40. The running times are shown in Figure 3.6.

3.4.4 Main Results

We compared the running times of our Sparse-opt routine versus the Projection-Hoyer and found that on the synthetically generated datasets we are faster on average.

Our results on switching the Sparse-opt routine with the Projection-Hoyer did not slow down our SSNMF solver significantly for the datasets we considered. So, we conclude that the speedup is mainly due to the sequential nature of the updates (Algorithm 4).

Also, we converge faster than NMFSC for fewer number of matrix updates. This can be seen by noting that the plotted points in Figures 3.5 and 3.6 are such that the number of matrix updates are the same for both SSNMF and NMFSC. For some datasets, we noted a speedup of an order of magnitude making our approach attractive for computation purposes.

Finally, we note that we recover a parts-based representation as previously shown [48]. An example of the obtained features by NMFSC and ours is shown in Figure 3.7.

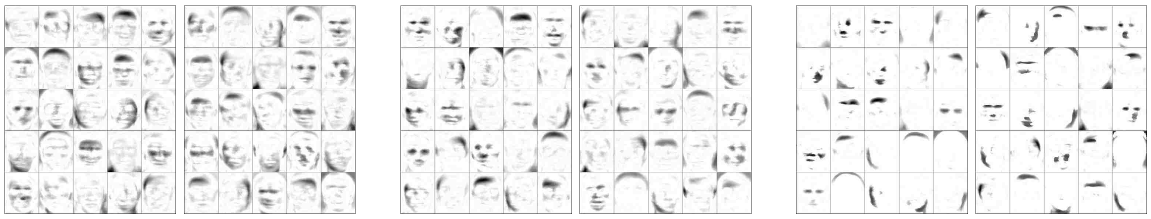


Figure 3.7: Features from (Left) NMFSC algorithm and (Right) SSNMF algorithm (Right) using the ORL face dataset for sparsity values $0.5, 0.6, 0.75$. Note that SSNMF algorithm gives a parts-based representation similar to the one recovered by NMFSC.

3.5 Connections to Related Work

Other SNMF formulations have been considered [47, 85, 58, 92, 93]. SNMF formulations using similar sparsity measures as used in this paper have been considered for applications in speech and audio recordings [111, 110].

We note that our sparsity measure has all the desirable properties, extensively discussed in [51], except for one (“cloning”). Cloning property is satisfied when two vectors of same sparsity when concatenated maintain their sparsity value. Dimensions in our optimization problem are fixed and thus violating the cloning property is not an issue. Compare this with the L_1 norm that satisfies only one of these properties (namely “rising tide”). Rising tide is the property where adding a constant to the elements of a vector decreases the sparsity of the vector. Nevertheless, the measure used in [58] is based on the L_1 norm. The properties satisfied by the measure in [92] are unclear because of the implicit nature of the sparsity formulation.

In [92], it was claimed that the SNMF formulation in [48], as given by problem (3.3) does not capture the variance in the data. However, some transformation of the sparsity values is required to properly match the two formulations [48, 92]. Preliminary results show that the formulation given in [48] is able to capture the variance in the data if the sparsity parameters are set appropriately. In [93], it was proposed to tackle the L_0 norm constrained NMF directly by projecting from intermediate unconstrained solutions to the required L_0 constraint. This leads to the well-known problem of getting stuck in local minima. Indeed, the authors re-initialize their feature matrix with an NNLS solver to recover from the local suboptimum. Our formulation avoids the local minima associated with L_0 norm by using a smooth surrogate.

3.6 Bi-Sparse NMF

In some applications, it is desirable to set the sparsity on both matrix factors. However, this can lead to the situation where the variance in the data is poorly captured [92]. To ameliorate this condition, we formulate it as the following optimization problem and name it Bi-Sparse NMF:

$$\begin{aligned} \min_{\mathbf{W},\mathbf{H},\mathbf{D}} \frac{1}{2}\|\mathbf{X}-\mathbf{WDH}\|_F^2 & \\ \text{s.t.}\mathbf{W} \geq \mathbf{0}, \mathbf{H} \geq \mathbf{0}, \mathbf{D} \geq \mathbf{0} & \\ \|\mathbf{W}_j\|_2 = 1, \text{sp}(\mathbf{W}_j) = \alpha, \forall j \in \{1,\cdots,r\} & \\ \|\mathbf{H}^i\|_2 = 1, \text{sp}(\mathbf{H}^i) = \beta, \forall i \in \{1,\cdots,r\} & \qquad (3.6) \end{aligned}$$

where $\mathbf{D}$ is a $r \times r$ matrix. In the above formulation, we constrain the L_2 norms of the columns of matrix $\mathbf{W}$ to unity. Similarly, we constrain the L_2 norms of rows of matrix $\mathbf{H}$ to be unity. This scaling is absorbed by the matrix $\mathbf{D}$. Note that this formulation with the matrix $\mathbf{D}$ constrained to be diagonal is equivalent to the one proposed in [48] when both the matrix factors have their sparsity specified.

We can solve for the matrix $\mathbf{D}$ with any NNLS solver. A concrete algorithm is the one presented in [32] and is reproduced here for convenience (Algorithm 6). If $\mathbf{D}$ is a diagonal matrix, we only update the diagonal terms and maintain the rest at zero. Algorithms 2 and 6 can be sped up by pre-computing the matrix products which are unchanged during the iterations.

Algorithm 6 Diag-mult($\mathbf{X}, \mathbf{W}, \mathbf{H}, \mathbf{D}$)

```
repeat
    $\mathbf{D} = \mathbf{D} \odot \frac{\mathbf{W}^\top \mathbf{X} \mathbf{H}}{\mathbf{W}^\top \mathbf{W} \mathbf{D} \mathbf{H} \mathbf{H}^\top}$
until convergence
Output: Matrix $\mathbf{D}$.
```

Also, the matrix $\mathbf{D}$ captures the variance of the dataset when we have sparsity set on both the matrices $\mathbf{W}, \mathbf{H}$.

3.7 Conclusions

We have proposed a new efficient algorithm to solve the sparse NMF problem. Experiments demonstrate the effectiveness of our approach on real datasets of practical interest. Our algorithm is faster over a range of sparsity values and generally performs better when the sparsity is higher. The speed up is mainly because of the sequential nature of the updates in contrast to the previously employed batch updates [48]. Also, we presented an exact and efficient algorithm to solve the problem of maximizing a linear objective with a sparsity constraint, which is an improvement over the previous approach [48].

Our approach can be extended to other NMF variants [47]. Another possible application is the sparse version of nonnegative tensor factorization. A different research direction would be to scale our algorithm to handle large datasets by chunking [78] and/or take advantage of distributed/parallel computational settings [9].

Chapter 4

Independence — A closer look

Daubechies et al. [28] claims that ICA for fMRI optimizes for sparsity rather than independence. This is established by first noting that Infomax and FastICA are two algorithms widely used for fMRI analysis and then showing that they separate sparse components better than independent ones on a synthetic dataset. Recreating the synthetic dataset and conducting additional experiments shows that the FastICA and Infomax algorithms indeed do what they are designed to do. Both ICA algorithms can separate sources with either high or low degrees of sparsity, as long as the distributional assumptions of the algorithms are approximately met. To understand the conditions under which these algorithms work requires correct interpretation of what the sources are in an ICA formulation. We examine *exactly what* the sources are in the examples given in Daubechies et al. [28] and show that there is an important mismatch between the concept of source therein and what an ICA source actually is, which is ultimately at the heart of the unsupported conclusions presented in Daubechies et al. [28].

4.1 Review and critique of the presented evidence

We now briefly review the evidence presented in Daubechies et al. [28] to support the claim that Infomax [5] and FastICA [52] select for sparsity and not independence. Daubechies et al. [28] exhibits experimental results in which 1) ICA algorithm performance suffers when the assumptions on the sources are violated, and 2) ICA algorithms can separate sources in certain cases even if the sources are not strictly independent. The two points above, both of which were already widely known in the ICA community at the time, are not sufficient evidence to support the claim that ICA selects for sparsity and not independence. In addition, Daubechies et al. [28] presents a case in which the sources are somewhat dependent but also very sparse, and Infomax and FastICA do well. This result is used to claim that it is sparsity rather than independence that matters. We augment this experiment with new evidence which shows that the same ICA algorithms perform equally well in the case of both minimum and maximum sparsity (using the definition of sparsity in Daubechies et al. [28]), suggesting that the role of sparsity (if any) is minor in the separation performance.

Additional evidence in Daubechies et al. [28] involves a discussion of sparsity in which it is claimed that ICA can separate Gaussian sources (See Legend of Figure 8 in Daubechies et al. [28]) which are also sparse (utilizing a definition of sparsity different from the one initially provided in Daubechies et al. [28]). If true, such a result would support their claim about the role of sparsity in ICA, since it is well established that blind ICA algorithms are not able to separate two or more Gaussian sources. However, as we show, in that example the sources as they are generated are highly non-Gaussian, and the sparsity mentioned in Daubechies et al. [28] does not actually refer to the sources. Rather, it refers to vectors that span parts of both sources. This renders their statement incorrect and hence, does not support the claim being made (see Section 6 for details).

Finally, the paper [28] is focused on showing cases where FastICA and Infomax perform well or poorly, and from these cases the claim is made that this applies to ICA of fMRI in general. There is mention that a more general algorithm [3] does not work for fMRI, but there is no evidence presented to support this claim. As we later discuss in Section 9, other ICA algorithms had indeed been used on fMRI data with success, at the time of the publication [28]. Since then, more flexible ICA algorithms have been applied to fMRI data and noted to demonstrate even better performance than the widely used Infomax and FastICA [71]. Hence, while emphasizing that Infomax and FastICA are not the only two algorithms that have been applied to fMRI analysis, we also note that the prevalence of the use of these two is largely due to the availability of the code for these algorithms and their default use in toolbox implementations for fMRI analyses. Since most of the fMRI community does not specialize in the development of blind source separation algorithms, they have since opted in general for the use of these two implementations. And although they do perform reasonably well on fMRI data, sparsity is not the major driver of this success.

4.2 Experiments on synthetic data: boxes

We now describe the synthetic dataset used in the original paper [28]. Two components C_1 and C_2 are generated as follows:

$$C_i(v) = I_{V_i}(v)x_v^i + [1 - I_{V_i}(v)]y_v^i \ , i \in \{1, 2\} \tag{4.1}$$

where the V_i are different subsets of V, and $I_{V_i}(v)$ denotes the indicator function for $v \in V_i$; the variables x^i, y^i are independent random variables and v is the sample index. In Example 1 [28], the cumulative distribution functions (CDFs) of x^i are identical and given by:

$$\Phi_x(u) = \frac{1}{1 + e^{(2-u)}}, \tag{4.2}$$

i.e., logistic distributions with mean 2 and scale parameter 1 (the standard deviation is $\pi/\sqrt{3}$). In Example 2 [28],

$$\Phi_x(u) = \frac{1}{1 + e^{2(2-u)}} \tag{4.3}$$

(logistic with mean 2, scale parameter 0.5, and standard deviation $\pi/(2\sqrt{3}$). Here, x^i correspond to the activations. Similarly, the cumulative distribution functions (CDF) of y^i are identical and given by $\Phi_y(u) = \frac{1}{1+e^{-1-u}}$, i.e., logistic distributions with mean 1 and scale parameter 1 (the standard deviation is $\pi/\sqrt{3}$), where y^i correspond to the background. The mixtures are given by:

$$X_1(v) = 0.5C_1(v) + 0.5C_2(v) \tag{4.4}$$

$$X_2(v) = 0.3C_1(v) + 0.7C_2(v). \tag{4.5}$$

We have $V = \{1, \cdots, 100\} \times \{1, \cdot, 100\}$, and in the case of "medium boxes":

$$V_1 = \{11, \cdots, 40\} \times \{21, \cdots, 70\} \tag{4.6}$$

$$V_2(\alpha) = \{31 + \alpha, \cdots, 80 + \alpha\} \times \{41 + \alpha, \cdots, 80 + \alpha\} \tag{4.7}$$

where $\alpha \in \{-15, \cdots, 15\}$. Furthermore, for Example 2 [28], in the case of "small boxes", the sample support sets are:

$$V_1 = \{41, \cdots, 60\} \times \{31, \cdots, 50\} \tag{4.8}$$

$$V_2(\alpha) = \{57 + \alpha, \cdots, 81 + \alpha\} \times \{46 + \alpha, \cdots, 65 + \alpha\} \tag{4.9}$$

and in the case of "large boxes":

$$V_1(\alpha) = \{1, \cdots, 48\} \times \{1, \cdots, 100\} \tag{4.10}$$

$$V_2(\alpha) = \{25 + \alpha, \cdots, 74 + \alpha\} \times \{1, \cdots, 100\} \tag{4.11}$$

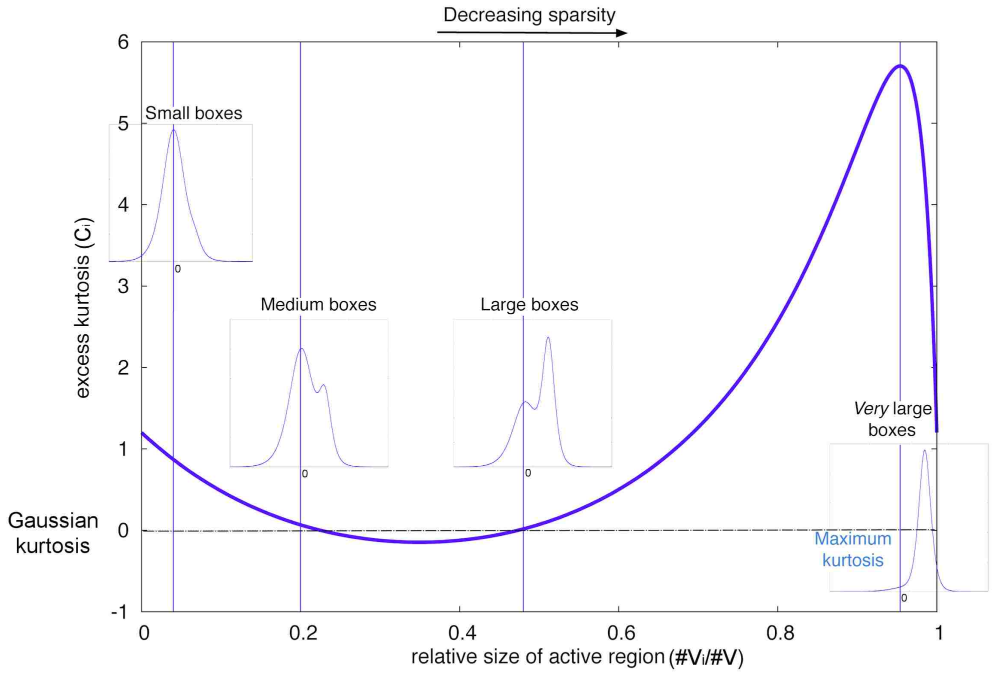
$\alpha = \{-10, \cdots, 20\}$. In all cases, α controls the relative position of the boxes, and $\alpha = 0$ gives statistical independence between C_1 and C_2.

4.3 The statistical properties of synthetic data [28]

Daubechies et al. [28] argues, based largely on results from synthetic datasets using boxes to represent activated regions of a component (see details above), that it is sparsity rather than independence that enables the recovery of the components. However, the case where the algorithms fail is actually due to a mismatch between the algorithms' assumptions and the *statistical properties* of the simulated data. In addition, we demonstrate a case where they perform best, which corresponds to almost the *lowest* sparsity (i.e., not sparse). To facilitate cross-referencing, in the results presented herein, we use the first definition of sparsity ($\frac{V_i}{V}$) provided in Daubechies et al. [28]. Note, however, that the quantification of sparsity may be ambiguous: see Section 8 below, and the two definitions of sparsity in Daubechies et al. [28].

Let us first concentrate on the choice of sources. In Figure 1, we see the excess kurtosis of the simulated sources changes with the relative size of the activation region. For medium and large boxes, the two cases where Infomax and FastICA are noted to fail, the kurtosis values are close to that of a Gaussian (i.e., zero), almost corresponding to the two zero-crossings. Moreover, in these cases the distributions are bimodal, far from the unimodal super-Gaussian assumptions that underpin the nonlinearities of Infomax and FastICA used in Daubechies et al. [28]. The paper [28] showed that Infomax with a non-linearity matched to super-Gaussian sources fails for medium and large boxes, roughly regardless of the relative position of the box; but it was *not* noted that the sources C_1, C_2 were very close to Gaussian (in the sense of kurtosis) and in disagreement with the nonlinearity. Both of these facts create very challenging scenarios for ICA algorithms based on the assumption of unimodal, super-Gaussian sources, as is the case in Infomax and FastICA, and of course sources are not even close to the "ideal" setup for these algorithms, contrary to the claim on p.10418 in Daubechies et al. [28]. In fact, under these scenarios components would *not* be expected to be well separated with either of these algorithms—because of the mismatch of the distribution (for Infomax) and an approximately zero kurtosis (for FastICA).

It is noted in Daubechies et al. [28] that the sources are designed by matching their cumulative distribution function (CDF) to the nonlinearity of the algorithm, resulting in "optimal" detectability for Example 1 [28] , and (intentionally) enforcing a "slight mismatch" for Example 2 [28] these two CDFs are actually the same, except for a scaling factor, which would translate to the so-called scaling ambiguity in ICA. More importantly though, there is a mismatch in vocabulary between what is being identified as the underlying ICA source in Daubechies et al. [28] and *what it actually is* in the experiment. Specifically, the nonlinearity matches solely to the activation part of the components thereby neglecting the background, whereas the ICA source is to be understood as a combination of the two, and thus has a distribution that is a *mixture distribution*, i.e., a weighted sum of both activation and background distributions. Hence the claim (p. 10418, 1st column): "For the first choice, the parameters of our ICA implementations provide optimal 'detectability' in the sense that the nonlinear function defined by the parameter setting of the algorithm coincides with the CDF of the signal source;" is incorrect since the *source* in this linear source separation framework cannot refer to only a part of the underlying distribution. As it turns out, in Example 2 [28] there is actually a *large* mismatch (rather than a “slight mismatch”) with respect to the algorithm’s nonlinearity in that the source distributions are essentially bimodal (see Figure 1, medium box inset).



FastICA (kurt)	0.04	0.15	0.31	0.02
Infomax (super)	0.02	0.74	0.92	0.01
Infomax (sub)		0.11	0.03	
ICA-EBM	0.04	0.05	0.01	0.01

good
poor
very poor

ISI for 4 algorithms

Figure 4.1: The excess kurtosis of a source as a function of the relative size of the active region. A Gaussian has zero excess kurtosis. Here as in Example 2 of the original paper [28]. The four vertical lines at correspond to the relative sizes of the small box, the medium box, the large box, and a very large box corresponding to the maximal kurtosis case. Note that the medium and large box experiments have near zero excess kurtosis, *i.e., kurtosis value matching that of a Gaussian*. In addition, the pdfs of these sources are bimodal (see inset figures), ensuring that ICA algorithms designed for unimodal super-Gaussian distributions such as Infomax and FastICA with standard parameter settings, will likely fail. At the bottom of the figure are the ISI values (see Equation (1)) for the various algorithms at those four points (see Table 1 for full list). Also note the best separation performance of Infomax and FastICA for the maximum kurtosis case, which corresponds to almost the *lowest* level of sparsity.

4.4 Boxes revisited

In the boxes experiment, there are four quantities that are varied: the relative position of the boxes (controlling the amount of overlap), the size of the boxes (small, medium, large), the distribution of the marginal (i.e., the source), and the joint distribution. The shift of the box changes the amount of overlap and, thus, the joint distribution/dependence. The box size controls the sparsity (small box = high sparsity, large box = low sparsity) through the proportion of , and thus changes the marginal distribution of the sources . Clearly, there is dependence between all four quantities, which makes interpretation of the results ambiguous at the least. This is a side effect of the way the sources are sampled in Daubechies et al. [28] which is not independent and identically distributed (i.i.d.) due to the use of the indicator function to define boxes in the spatial map (the sampling distribution is not identical but, instead, conditioned on the location of each sample). With such a design it is very difficult to understand what causes the experimental differences, which is contrary to the claim [28] that it is "easy to change each of these characteristics separately". In addition, in the experiments, a single fixed mixing matrix is used, which is not an ideal way to evaluate performance as results are then biased to a specific (and unjustified) set of mixing matrix parameter choice.

In order to furnish a clear, unbiased interpretation of the effect of the marginal source distributions (closely related to the box-sizes in Daubechies et al. [28]) on the performance of ICA algorithms that exploit non-Gaussianity, we first eliminate the effects of all other parameters by limiting ourselves to the case of two *independent* sources and . Then we generate samples directly from marginal distributions that match those in Daubechies et al. [28] Since the sources defined in Daubechies et al. [28] have distributions that are of mixture type, we can write the CDF of each source C_i as:

$$\Phi_{C_i} = q\Phi_x + (1-q)\Phi_y \tag{4.12}$$

where $0 < q \leq 1$ with $q = \frac{V_i}{V}$, and then draw a set of i.i.d. samples. Under these conditions, the joint distribution of all samples reads

$$p_{C_1[1],C_2[1],\ldots,C_1[V],C_2[V]}(C_1[1],C_2[1],\ldots,C_1[V],C_2[V]) = \prod_{v=1}^{|V|} p_{C_1[v]}(C_1[v])p_{C_2[v]}(C_2[v]) \tag{4.13}$$

, where v is the sample index and $|V|$ is length of V. The first equality follows from independent sampling, the second equality from the independence between components C_1 and C_2, and the third equality from the samples being identically distributed (same distribution regardless of the sample index v). As such, we may generate all samples using independent samples from the inverse CDF transforms $\Phi_{C_i}^{-1}(u_i[v])$, where $u_i[v], i \in \{1,2\}, v \in \{1,2,\ldots,V\}$ are i.i.d. samples from the independent random variables $U_i, i \in \{1,2\}$, uniformly distributed on $[0,1]$, and $\Phi_{C_i}^{-1}$ is the inverse CDF of the mixture distribution

$$\Phi_{C_i}(C_i[v]) = q\Phi_x(C_i[v]) + (1-q)\Phi_y(C_i[v]) \tag{4.14}$$

Here Φ_x is the logistic distribution for activation and Φ_y is the logistic distribution for background as defined in Daubechies et al. [28], and q is the relative area of the activation. To achieve the required visual contrasts—small, medium, large and very large boxes, at any desired position—we reorder the two-dimensional samples, never decoupling the realizations of the sources. The final result, while having a similar visual appearance as the experiments of Daubechies et al. [28] retains the joint pdf. This eliminates possible confusion with respect to the influence of the different box parameters on the results of our experiments. We then compute our results using four algorithms: 1) Infomax with the standard sigmoid nonlinearity that assumes a unimodal super-Gaussian source, called Infomax (super); 2) FastICA with the same nonlinearity used in Daubechies et al. [28], which is ; 3) Infomax with a nonlinearity which assumes a sub-Gaussian source, called Infomax (sub); and 4) ICA-EBM (ICA by entropy bound minimization), a much more flexible ICA algorithm [74] able to deal with both super- and sub-Gaussian sources.

Results are averaged over 100 source realizations (each using a different random full-rank mixing matrix) and 10 ICA runs (see Table 1). We also report two performance metrics, first, using the metric chosen in Daubechies et al. [28], which is *not* invariant to the scaling and permutation ambiguities inherent to ICA. Hence, we also report the results using the inter-symbol interference (ISI), or normalized Moreau-Amari index [82], which is invariant to the scaling and permutation ambiguities:

$$\text{ISI}(P) = \frac{1}{2L(L-1)}[\sum_{i=1}^{L}(\sum_{j=1}^{L}\frac{|p_{ij}|}{\max_k |p_{ik}|} - 1) + \sum_{j=1}^{L}(\sum_{i=1}^{L}\frac{|p_{ij}|}{\max_k |p_{kj}|} - 1)] \tag{4.15}$$

Here, p_{ik} are the elements of the matrix $P = WA$, and L is the number of sources. This performance metric is bounded between 0 and 1 and the lower the ISI value the better the separation performance (the performance metric is zero if and only if the model is identified up to the scaling and permutation ambiguities).

As expected, the most flexible approach, the ICA-EBM algorithm, performs well (ISI$<$ 0.1) in all cases (Table 4.1). Infomax (sub) performs well to moderately-well for the large and medium boxes, both of which are bimodal and have a kurtosis that is close to that of a Gaussian random variable. Infomax (super) and FastICA perform marginally well or poorly in those cases but perform very well for the cases of very large boxes (maximum kurtosis) and for small boxes. This makes intuitive sense, as high-kurtosis data matches the underlying assumptions of both Infomax (super) and FastICA in that the source distributions are unimodal and strongly super-Gaussian. These results directly contradict the claim in Daubechies et al. [28] that Infomax (super) and FastICA select for sparsity, since the maximum kurtosis case also has the lowest sparsity of the four (again using the first definition of sparsity in Daubechies et al. [28].

4.5 Sparsity and sources that are mixture of Gaussians

In the sparsity section in Daubechies et al. [8, p.10421, Figure 8] there are several incorrect statements that are important and require a careful critique. First, Daubechies et al. [28] claims that the sources in the so-called "promotional material for ICA" are Gaussian. We show below that they are in fact highly non-Gaussian. Second, a definition of sparsity different from the one proposed earlier in the paper [28] is used to claim that the sources are sparse. We show that this sparsity does not refer to the sources and in actuality they are not sparse. Finally, we correct several other statements within that section.

Counter proof to claim of Gaussian sources: To identify the distribution of the sources in this example, it is sufficient to look along the mixing directions $\boldsymbol{a}$ and $\boldsymbol{b}$. Observations are defined as:

$$\boldsymbol{r} = \gamma \boldsymbol{r}_1 + (1-\gamma)\boldsymbol{r}_2 \tag{4.16}$$

$$= \gamma[\alpha_1 \boldsymbol{a} + \beta_1 \boldsymbol{b}] + (1-\gamma)[\alpha_2 \boldsymbol{a} + \beta_2 \boldsymbol{b}] \tag{4.17}$$

Reordering the terms gives:

$$\begin{aligned} \boldsymbol{r} &= \boldsymbol{a}[\gamma\alpha_1 + (1-\gamma)\alpha_2] + \boldsymbol{b}[\gamma\beta_1 + (1-\gamma)\beta_2] \\ &= \begin{bmatrix} \boldsymbol{a} & \boldsymbol{b} \end{bmatrix} \boldsymbol{s} \\ &= \boldsymbol{A}\boldsymbol{s} \end{aligned} \tag{4.18}$$

$$s_a = \gamma\alpha_1 + (1-\gamma)\alpha_2 \tag{4.19}$$

$$s_b = \gamma\beta_2 + (1-\gamma)\beta_2 \tag{4.20}$$

Thus, the sources can be identified as mixture distributions. Their distributions are given as:

$$p_{s_a} = \lambda p_{\alpha_1} + (1-\lambda) p_{\alpha_2} p_{s_b} = \lambda p_{\beta_2} + (1-\lambda) p_{\beta_2} \tag{4.21}$$

where λ is the parameter of the Bernoulli distribution of which γ are the realizations. Notice that contrary to what one might expect, a mixture of Gaussian random variables through a Bernoulli random variable as above, in general does not yield a Gaussian random variable, but rather a random variable whose pdf is a weighted mixture of two independent Gaussian pdfs. Finally, since the distributions $p_{\alpha_1}, p_{\alpha_2}, p_{\beta_1}$, and $p_{\beta_2} (p_{\alpha_1} = p_{\beta_2} = p(t))$ are all Gaussian and no two distributions in a mixture have the same variance—e.g., for the choice in Daubechies et al. [8], $p_{\alpha_2}(t) = 10 p_{\alpha_1}(10t)$, which implies $Var(\alpha_2) = 100 Var(\alpha_1) = 100\sigma^2$ — the resulting distribution must be non-Gaussian whenever $\lambda \notin \{0, 1\}$. Hence the statement in Daubechies et al. [8] that "Each component has a Gaussian distribution", is incorrect; the components are in actuality highly non-Gaussian (see Figure 2 (A-B)).

Critique of the claim of sparse components: In the same section there is a claim that the components (*i.e.* sources) in this example are sparse: "Fig. 8 depicts processes with 2 sparse rather than independent components". However, sparsity as defined in this section does not refer to the components at all; rather, it refers to parts of the components together, specifically, the 2D Gaussian vectors $\begin{bmatrix}\alpha_1 & \beta_1\end{bmatrix}$ and $\begin{bmatrix}\alpha_2 & \beta_2\end{bmatrix}$ which are 2D, 1-sparse vector processes. In actuality, however, the components s_a and s_b are *not* sparse for the choice of $\lambda = 50\%$ and $\lambda = 30\%$ used in M_1 and M_2 respectively. This is because $Var(s_a) = \sigma^2 \frac{99\lambda+1}{100}$ and $Var(s_b) = \sigma^2 \frac{100-99\lambda}{100}$, which are typically much greater than zero. Therefore, it cannot be sparsity that is driving these algorithms towards the solution.

A few additional clarifications: There are two other sentences in the section on sparsity in Daubechies et al. [8] which require some clarification. First, in the sentence "However, in the example given here, is Gaussian; because ICA methods cannot separate mixtures of independent Gaussian processes, the successful separation of components by Infomax and FastICA underscores again their ability to identify sparse components" $p(t)$ is *not* the distribution of the components . In addition, the statement instills belief that this example has only a single mixing process, when in fact it has two: 1) the mixing of the (Gaussian) α_k's and β_k's through λ , which gives the (non-Gaussian) sources s_i, and 2) the mixing of sources s_i through the mixing matrix $\boldsymbol{A} = \begin{bmatrix}\boldsymbol{a} & \boldsymbol{b}\end{bmatrix}$. The statement suggests Infomax and FastICA can unmix the Gaussian random variables α_k, β_k which constitute the mixture distribution of a source (i.e. the two parts of a single source s_i) which is clearly incorrect (they unmix the sources s_i, not their subparts). Lastly, the sentence "Infomax or FastICA identify the 2 special directions **a** and **b** correctly as the components" incorrectly labels $\boldsymbol{a}$ and $\boldsymbol{b}$ as components, when they actually are the *mixing coefficients* that make up the $\boldsymbol{A}$ matrix.

4.6 ICA of sources with mixture of Gaussians distribution

The discussion related to the example in Figure 8 of the original paper [8] initially notes that mixtures of independent Gaussian random variables cannot be recovered by ICA, which is true if each source comes from a single Gaussian distribution, and the algorithms are only based on higher-order statistics, as in the case of Infomax and FastICA (i.e., the algorithms do not exploit sample correlation). However, these algorithms (and many oth-

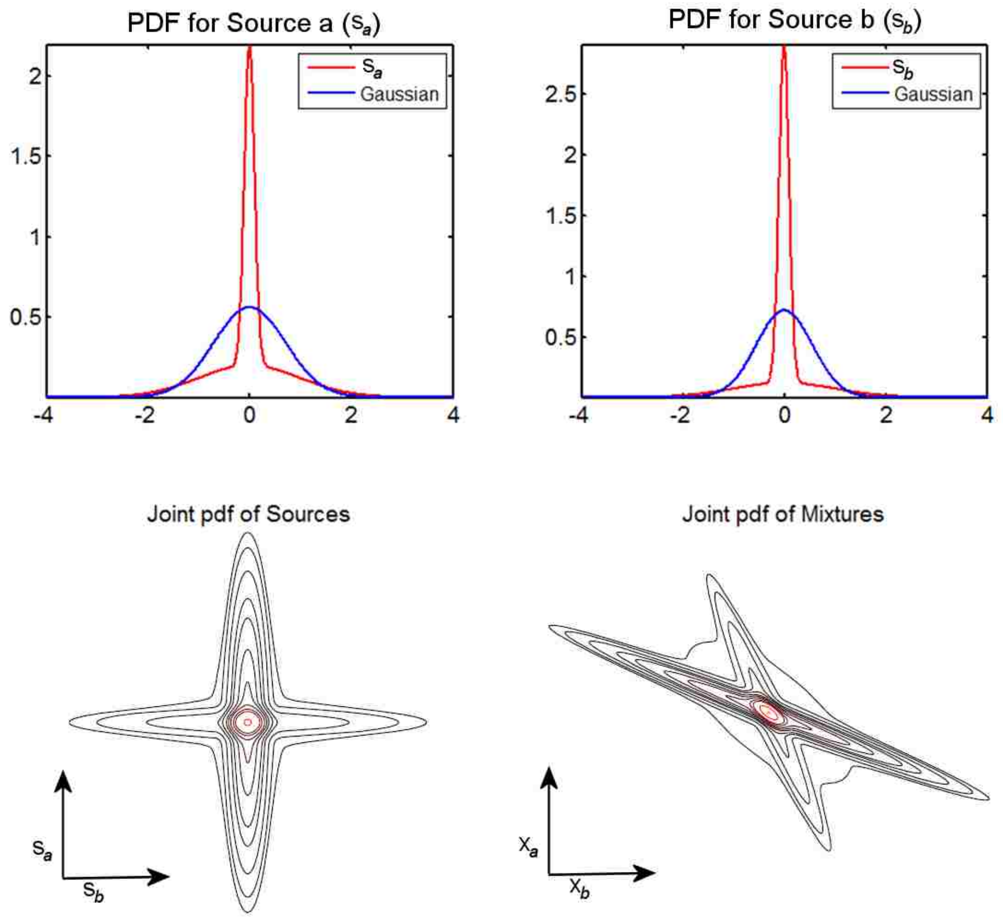


Figure 4.2: The distribution of sources and mixtures for $\lambda = 30\%(M_2)$. We plot (A-C) the distribution of sources, and (D) the contour plot of mixtures for the case of $\lambda = 30\%(M_2)$. Contrary to the claim made in Daubechies et al., the sources have in fact very peaky and heavy-tailed distributions and are not at all close to a Gaussian distribution. For comparison purposes we also present Gaussian distribution curves (blue, A-B).

ers that have been developed and also applied to fMRI data [14]) *can* separate sources whose probability density can be represented via a Gaussian mixture model, as long as the resulting distribution itself is not a Gaussian. The latter is the case in the example presented in Figure 8 of Daubechies et al. [8], which was incorrectly seen as evidence that sparsity was the driving force helping ICA to recover Gaussian sources. We showed that the sparsity mentioned in Daubechies et al. [8] is not related to the sources. Also, this example utilizes a mixture of Gaussians as the sources. With the parameters described in Daubechies et al. [8], the sources are in fact super-Gaussian (i.e. they have positive excess kurtosis, as shown in Table 4.2). Infomax and FastICA with nonlinearities selected to match a super-Gaussian distribution are expected to successfully separate such sources, as also is the more flexible ICA-EBM algorithm [12]. Conversely, Infomax with a nonlinearity selected to be sensitive to sub-Gaussian sources is expected to exhibit suboptimal performance (see performance Table 2). This can also be visualized in Figure 2 where we show the sources and the mixtures for the case of $\lambda = 30\%$ as described in Daubechies et al. [8]. This example again points to the confusion discussed in Section 4 with respect to the definition of the underlying ICA sources, i.e., what is actually being simulated and what is assumed in Daubechies et al. [8].

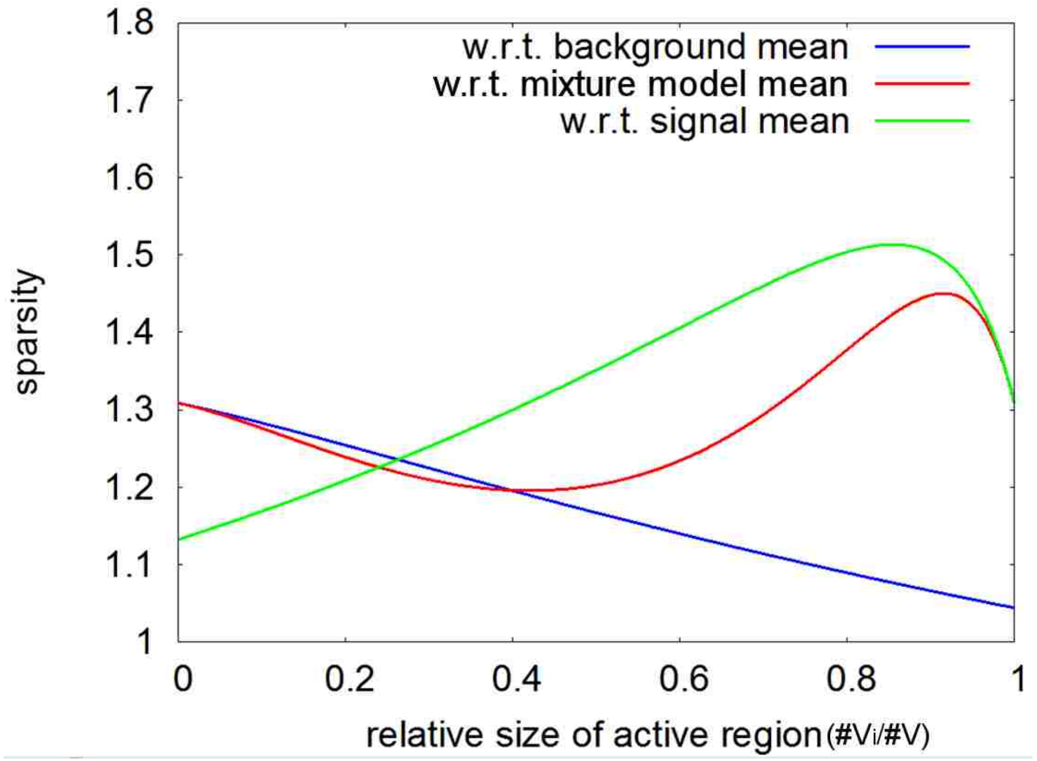


Figure 4.3: Sparsity measures for three different coordinate system origins z_0. Sparsity as measured with respect to different coordinate system origins z_0, as a function of the relative size of the active region. Remark that for a relative size of zero, only background samples are present and, thus, the mean of the mixture model coincides with the mean of the background (and the two sparsity measures correspond at this point). An analogous observation can be made for a relative size of one, now with respect to the activity (signal samples).

4.7 On the definition of sparsity

In coding theory, whether in transmission or in storage of a signal, a trade-off often is necessary between attainable compression rates and signal restoration error. In this context, sparsity is a signal property that allows for high compression rates, while compromising only little in the restoration error. A sparse signal generally consists of $N = |V|$ coefficients of which $n << N$ coefficients concentrate all information within the signal. Indeed, under the hypothesis that coding a string of zeroes has little cost in resources with respect to coding whatever floating/integer number, all other $N - n$ coefficients could be set to zero without significant loss of information but with a substantial gain in compression rate.

A legitimate question now is what about a signal of which all but 1 coefficient differ from a number, say, μ. Let that one coefficient equal zero. Is that signal sparse? Under the above definition, the signal would not be considered as sparse, since only a single coefficient could be coded as a zero without introducing a reconstruction error. However, if we would allow for coding a shift by $-\mu$, then coding $N - 1$ coefficients as zero would result in a reconstruction error ϵ upper bounded by $\|\mu\|$. It is clear from this very simple example that it is important to appropriately choose the origin for the coordinate system (z_0) in which one foresees to evaluate the sparseness of the signal. For the model considered in Daubechies et al. [8], we plot the sparsity measure $\sqrt{E_z\{(z - z_0)^2\}}/E_z\{|z - z_0|\}$ for three different choices of z_0. Here, the ordinary sparsity measure (as understood in Daubechies et al. [8]) is taken with respect to $z_0 = -1$, i.e., the mean of the "background distribution", with sparsity decreasing as the active region size increases (see Figure 3). Note that for fMRI we typically use zero-mean samples when using ICA, thus measuring our sparsity with respect to the mean of the mixture model.

4.8 On the application of ICA to fMRI

We also note that, contrary to the claims in Daubechies et al. [8], Infomax and FastICA, though the most widely used at the time—due in large part to their availability in fMRI-friendly software packages—were not the only ICA algorithms that had been applied to fMRI analysis with success at the time [26, 46]. This trend has continued and in recent years even more flexible algorithms such as those based on entropy bound minimization (EBM) or full blind source separation (FBSS) have been used increasingly to analyze fMRI data, outperforming both Infomax and FastICA [15, 72, 35]. In general, we would recommend that these and other more recent algorithms preferentially be applied to fMRI, as they are generally more robust to non-super-Gaussian and/or multimodal distributed sources which can occur in real fMRI data, observed in the context of certain artifacts. These algorithms and many others are implemented in the group ICA of fMRI toolbox (GIFT; http://mialab.mrn.org/software/gift). An interesting historical note is that before extended Infomax [69] was introduced, there was confusion as to how ICA of fMRI really worked when it was applied as temporal ICA and early results indeed were not convincing—since time courses are more likely to be sub-Gaussian than super-Gaussian [80], whereas in the spatial ICA case super-Gaussian sources are more common. Another important point regarding the real fMRI experiment mentioned in Daubechies et al. [8] is that each voxel is identified as belonging to only one underlying source (page 10416, left col, third paragraph). Such an approach is perhaps a reflection of the way one might approach an fMRI experiment with a sparsity focus, but in reality, and more in line with the complexity and connectivity of the human brain, each voxel typically has a contribution from multiple components (sources), making this an ideal case for ICA.

4.9 Conclusions

We reviewed the main claim made in Daubechies et al. [8] and its supporting evidence. We revisit the initial experiments and present new evidence showing conclusively that the arguments fall short of supporting the claim that Infomax and FastICA select for sparsity and not for independence. While pointing out that the use of other metrics for fMRI analysis such as sparsity—besides independence, which is widely used—is a reasonable goal, the claims that are used to justify this desire are misleading at best and in some cases are simply incorrect. In summary, we show that ICA algorithms, including FastICA and Infomax, are indeed doing what they were designed to do, maximize independence.

Boxes Size	Observed Features (good is ISI < 0.1)		
Small Unimodal, super-Gaussian sources	Source C_1 (excess) Kurtosis [< 0.1 is Gauss-like] :		0.8829
	Source C_2 (excess) Kurtosis [< 0.1 is Gauss-like] :		0.8107
	Mutual information Between Sources C_1 and C_2 :		0.0920
	Algorithm	Daubechies	Amari
FastICA Infomax (super), and ICA-EBM perform well	FastICA	0.0547 ± 0.0150	0.0383
	Infomax (super)	0.0331 ± 0.0002	0.0228
	Infomax (sub)	1.0493 ± 0.0015	0.9499
	ICA-EBM	0.0554 ± 0.0066	0.0388
Medium Bimodal, close-to Gaussian sources	Source C_1 (excess) Kurtosis [< 0.1 is Gauss-like] :		0.2564
	Source C_2 (excess) Kurtosis [< 0.1 is Gauss-like] :		0.0879
	Mutual information Between Sources C_1 and C_2 :		0.0929
	Algorithm	Daubechies	Amari
ICA-EBM performs good, Infomax (sub), performs fair	FastICA	0.2068 ± 0.0662	0.1464
	Infomax (super)	0.8722 ± 0.0651	0.7434
	Infomax (sub)	0.1597 ± 0.0058	0.1144
	ICA-EBM	0.0693 ± 0.0105	0.0488
Large Bimodal, close-to Gaussian sources	Source C_1 (excess) Kurtosis [< 0.1 is Gauss-like] :		0.0010
	Source C_2 (excess) Kurtosis [< 0.1 is Gauss-like] :		0.0762
	Mutual information Between Sources C_1 and C_2 :		0.0892
	Algorithm	Daubechies	Amari
ICA-EBM and Infomax (sub) perform well	FastICA	0.4081 ± 0.1003	0.3102
	Infomax (super)	1.0297 ± 0.0009	0.9236
	Infomax (sub)	0.0401 ± 0.0004	0.0260
	ICA-EBM	0.0145 ± 0.0008	0.0094
Very Large (max kurtosis) Unimodal, super-	Source C_1 (excess) Kurtosis [< 0.1 is Gauss-like] :		5.6432
	Source C_2 (excess) Kurtosis [< 0.1 is Gauss-like] :		5.6394
	Mutual information Between Sources C_1 and C_2 :		0.0686
Gaussian sources.	Algorithm	Daubechies	Amari
FastICA, Infomax (super) and ICA-EBM perform well	FastICA	0.0263 ± 0.0078	0.0180
	Infomax (super)	0.0131 ± 0.0003	0.0086
	Infomax (sub)	1.0711 ± 0.0014	0.9762
	ICA-EBM	0.0218 ± 0.0019	0.0148

Table 4.1: Source estimates for the four cases indicated in Figure 1 as in Example 2 of the original paper [8]. The algorithms behave as one would expect if they are selecting for independence. For the bimodal or Gaussian-like cases, ICA-EBM and Infomax (sub) do well, and for the unimodal or maximum kurtosis or low sparsity case Infomax-super, FastICA and ICA-EBM all do extremely well. Numbers in boldface indicate when separation was good.

Observed Features (good is < 0.1)		
Property	Source a (s_a)	Source b (s_b)
Negentropy	0.2753	0.3708
(excess) Kurtosis	3.0630	3.5225
Algorithm	Daubechies	Amari
FastICA	0.0154	0.0108
Infomax (super)	0.0076	0.0052
Infomax (sub)	1.0758	0.9899
ICA-EBM	0.0059	0.0039

Table 4.2: Tabulated results for the so-called [28] ICA “promotional material”. Both Infomax (super) and FastICA do successfully separate (zero indicates perfect separation) the super-Gaussian sources s_a and s_b . Note the excess kurtosis is more than 3 for both sources. Numbers in boldface indicate when separation was good.

Chapter 5

Conclusions and Future Work

Connections: We explored the structural connections between nonnegative least squares (NNLS), nonnegative matrix factorization (NMF) and support vector machines (SVM). In particular, we showed a reduction from totally nonnnegative least squares to support vector machines. This gave us insight into the connection between nonnegativity and sparsity and further enabled us to propose an efficient algorithm to solve the TNNLS problem. Also, we show that nonnegativity corresponds to sparsity depending on how many elements lie on the maximum-margin hyperplane. This explains the connection between nonnegativity and sparsity posed in the work on compressive sensing [89]. In particular, it enabled enabled us to reduce the planning time for a cancer treatment planning system by an order of magnitude compared to the state-of-the-art solvers. Also, we exploited the structural connection between NMF and SVM to propose novel algorithms for both the SVM and NMF problems. Recently, an equivalence between L2-SVM and LASSO has been shown [53]. That is given an L2-SVM problem, we can construct a LASSO problem with an equivalent solution and vice-versa. Can we expect to find a similar equivalence/reduction from NNLS to SVM? This will enable one to use existing fast SVM solvers to general NNLS problems. Also, it would be interesting to extend the algorithms to the distributed setting where the data is split across the computational nodes. Other issues such as privacy [19] may need to be addressed in this setting. One such scenario is when maintaining confidentiality of medical records. We derived simple multiplicative update rules for solving the maximum-margin classifier problem in SVMs. Coordinate descent is another popular approach to solve SVM problems [50]. Recently, coordinate-descent methods have been accelerated to reach a quadratic rate of convergence in the parallel setting on general convex problems [98, 86]. It would be interesting to compare the performance of coordinate-descent methods versus multiplicative updates in GPU settings [10]. Also, multiplicative updates only have a linear rate of convergence. Can they also be accelerated to achieve a quadratic rate of convergence?

Sparsity: We developed a block coordinate descent for solving sparse NMF problem and showed it to be an order of magnitude faster than the competing algorithms on real-world datasets consisting of faces, structural and functional MRI images. Our algorithm is faster over a range of sparsity values and generally performs better when the sparsity is higher. The speed up is mainly because of the sequential nature of the updates in contrast to the previously employed batch updates [48]. Also, we presented an exact and efficient algorithm to solve the problem of maximizing a linear objective with a sparsity constraint, which is an improvement over the previous approach [48]. Faster algorithms for the projection problem have also been presented [116, 108] but at a price that they do not completely characterize the solution over all sparsity values [95]. Our approach can be extended to other NMF variants [47]. Another possible application is the sparse version of nonnegative tensor factorization. A different research direction would be to scale our algorithm to handle large datasets by chunking [78] and/or take advantage of distributed/parallel computational settings [9]. Other extensions include handling orthogonality [32, 22], convolutive [105] or Lp-norm constraints.

ICA: Finally, we investigated a recent claim in a PNAS article [28] that sparsity and not independence is the cause for the success of independent component analysis (ICA) algorithms in fMRI analysis. Sparsity has been shown to be successful for a wide range of domains and it can play an important role in fMRI analysis. Recent work [64] has exploited sparsity combined with orthogonality to learn overcomplete representations. So, it might fruitful to further explore the connection between sparsity, nonnegativity and independence. Nonnegativity combined with orthogonality requirements has been exploited to achieve independence [113]. Conditions under which the original matrix factors, given just the data matrix, can be recovered are being studied for dictionary learning [106]. It would be interesting to extend these result for nonnegative matrix factorization. Initial results for recoverability in the case of ICA have begun to appear [2].

References

[1] Sanjeev Arora, Rong Ge, Ravindran Kannan, and Ankur Moitra. Computing a nonnegative matrix factorization – provably. In *Proceedings of the 44th symposium on Theory of Computing*, STOC '12, pages 145–162, New York, NY, USA, 2012. ACM.

[2] Sanjeev Arora, Rong Ge, Ankur Moitra, and Sushant Sachdeva. Provable ica with unknown gaussian noise, and implications for gaussian mixtures and autoencoders. *arXiv preprint arXiv:1206.5349*, 2012.

[3] Hagai Attias. Independent factor analysis. *Neural computation*, 11(4):803–851, 1999.

[4] H. Avron, P. Maymounkov, and S. Toledo. Blendenpik: Supercharging lapack's least-squares solver. *SIAM Journal on Scientific Computing*, 32(3):1217–1236, 2010.

[5] A. J. Bell and T. J. Sejnowski. An information-maximization approach to blind separation and blind deconvolution. *Neural Computation*, 7(6):1129–1159, November 1995.

[6] M. H. Van Benthem and M. R. Keenan. Fast algorithm for the solution of large-scale non-negativity-constrained least squares problems. *Journal of chemometrics*, 18(10):441–450, 2004.

[7] Michael W Berry, Murray Browne, Amy N Langville, V Paul Pauca, and Robert J Plemmons. Algorithms and applications for approximate nonnegative matrix factorization. *Computational Statistics & Data Analysis*, 52(1):155–173, 2007.

[8] C. Boutsidis and P. Drineas. Random projections for the nonnegative least-squares problem. *Linear Algebra and its Applications*, 431(5-7):760–771, 2009.

[9] Joseph K. Bradley, Aapo Kyrola, Danny Bickson, and Carlos Guestrin. Parallel coordinate descent for L1-regularized loss minimization. In *ICML*, pages 321–328, 2011.

[10] Matthew Brand and Donghui Chen. Parallel quadratic programming for image processing. In *ICIP*, pages 2261–2264, 2011.

[11] R. Bro and S. De Jong. A fast non-negativity-constrained least squares algorithm. *Journal of Chemometrics*, 11(5):393–401, 1997.

[12] A. M. Bruckstein, M. Elad, and M. Zibulevsky. A non-negative and sparse enough solution of an underdetermined linear system of equations is unique. *Submitted to IEEE Transactions on Information Theory*, 2008.

[13] G. Buchsbaum and O. Bloch. Color categories revealed by non-negative matrix factorization of munsell color spectra. *Vision research*, 42(5):559–563, 2002.

[14] V. D. Calhoun, T. Adali, G. D. Pearlson, and J. J. Pekar. A method for making group inferences from functional MRI data using independent component analysis. *Human Brain Mapping*, 14(3):140–151, 2001.

[15] Vince D Calhoun and T Adali. Multisubject independent component analysis of fmri: a decade of intrinsic networks, default mode, and neurodiagnostic discovery. *Biomedical Engineering, IEEE Reviews in*, 5:60–73, 2012.

[16] Cardoso. High-order contrasts for independent component analysis. *Neural Computation*, 11(1):157, 1999.

[17] Ali Taylan Cemgil. Bayesian inference for nonnegative matrix factorisation models. *Computational Intelligence and Neuroscience*, 2009, 2009.

[18] C. I. Chang and D. C. Heinz. Constrained subpixel target detection for remotely sensed imagery. *Geoscience and Remote Sensing, IEEE Transactions on*, 38(3):1144–1159, 2000.

[19] Kamalika Chaudhuri, Claire Monteleoni, and Anand D Sarwate. Differentially private empirical risk minimization. *The Journal of Machine Learning Research*, 12:1069–1109, 2011.

[20] Yunmei Chen and Xiaojing Ye. Projection onto a simplex. *arXiv preprint arXiv:1101.6081*, 2011.

[21] L. Chin and W. Regine. *Principles and practice of stereotactic radiosurgery*. Springer Verlag, 2008.

[22] Seungjin Choi. Algorithms for orthogonal nonnegative matrix factorization. In *Neural Networks, 2008. IJCNN 2008.(IEEE World Congress on Computational Intelligence). IEEE International Joint Conference on*, pages 1828–1832. IEEE, 2008.

[23] A. Cichocki and A. H. Phan. Fast local algorithms for large scale nonnegative matrix and tensor factorizations. *IEICE Transactions on Fundamentals of Electronics*, 92:708–721, 2009.

[24] Andrzej Cichocki and Rafal Zdunek. Regularized alternating least squares algorithms for non-negative matrix/tensor factorization. In *ISNN '07: Proceedings of the 4th international symposium on Neural Networks*, pages 793–802, Berlin, Heidelberg, 2007. Springer-Verlag.

[25] J. E. Cohen and U. G. Rothblum. Nonnegative ranks, decompositions, and factorizations of nonnegative matrices. *Linear Algebra and its Applications*, 190:149–168, 1993.

[26] Nicolle Correa, Tülay Adalı, and Vince D Calhoun. Performance of blind source separation algorithms for fmri analysis using a group ica method. *Magnetic resonance imaging*, 25(5):684–694, 2007.

[27] Thilo-Thomas Frießand Nello Cristianini and Colin Campbell. The Kernel-Adatron algorithm: a fast and simple learning procedure for support vector machines. In *Proc. 15th International Conf. on Machine Learning*, pages 188–196. Morgan Kaufmann, San Francisco, CA, 1998.

[28] I. Daubechies, E. Roussos, S. Takerkart, M. Benharrosh, C. Golden, K. D'Ardenne, W. Richter, JD Cohen, and J. Haxby. Independent component analysis for brain fMRI does not select for independence. *Proceedings of the National Academy of Sciences*, 106(26):10415–10422, 2009.

[29] Inderjit Dhillon and Suvrit Sra. Generalized nonnegative matrix approximations with Bregman divergences. In Y. Weiss, B. Schölkopf, and J. Platt, editors, *Advances in Neural Information Processing Systems 18*, pages 283–290. MIT Press, Cambridge, MA, 2006.

[30] Inderjit S Dhillon and Suvrit Sra. Generalized nonnegative matrix approximations with bregman divergences. In *NIPS*, volume 18, 2005.

[31] Chris Ding, Tao Li, and Michael I. Jordan. Convex and semi-nonnegative matrix factorizations. *LBNL Tech Report 60428*, 2006.

[32] Chris Ding, Tao Li, Wei Peng, and Haesun Park. Orthogonal nonnegative matrix t-factorizations for clustering. In *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD '06, pages 126–135, New York, NY, USA, 2006. ACM.

[33] D. L. Donoho and J. Tanner. Thresholds for the recovery of sparse solutions via l1 minimization. In *Information Sciences and Systems, 2006 40th Annual Conference on*, pages 202–206. IEEE, 2006.

[34] David Donoho and Victoria Stodden. When does non-negative matrix factorization give a correct decomposition into parts? In Sebastian Thrun, Lawrence Saul, and Bernhard Schölkopf, editors, *Advances in Neural Information Processing Systems 16*. MIT Press, Cambridge, MA, 2004.

[35] Wei Du, Hualiang Li, Xi-Lin Li, Vince D Calhoun, and Tülay Adali. Ica of fmri data: performance of three ica algorithms and the importance of taking correlation information into account. In *Biomedical Imaging: From Nano to Macro, 2011 IEEE International Symposium on*, pages 1573–1576. IEEE, 2011.

[36] John Duchi, Shai Shalev-Shwartz, Yoram Singer, and Tushar Chandra. Efficient projections onto the l1-ball for learning in high dimensions. In *Proceedings of the 25th international conference on Machine learning*, pages 272–279, 2008.

[37] J. Eggert and E. Körner. Sparse coding and NMF. *Neural Networks, 2004. Proceedings. 2004 IEEE International Joint Conference on*, 4:2529–2533, 25-29 July 2004.

[38] Cédric Févotte, Nancy Bertin, and Jean-Louis Durrieu. Nonnegative matrix factorization with the itakura-saito divergence: With application to music analysis. *Neural computation*, 21(3):793–830, 2009.

[39] V. Franc, V. Hlavac, and M. Navara. Sequential coordinate-wise algorithm for the non-negative least squares problem. In *Computer Analysis of Images and Patterns*, page 407, 2005.

[40] V. Franc and S. Sonnenburg. Optimized cutting plane algorithm for support vector machines. In *Proceedings of the 25th international conference on Machine learning*, pages 320–327. ACM, 2008.

[41] Nicolas Gillis and François Glineur. Using underapproximations for sparse nonnegative matrix factorization. *Pattern recognition*, 43(4):1676–1687, 2010.

[42] Gene H. Golub and Charles F. Van Loan. *Matrix Computations (3rd Ed.)*. Johns Hopkins University Press, Baltimore, MD, USA, 1996.

[43] Matthias Heiler and Christoph Schnörr. Learning sparse representations by nonnegative matrix factorization and sequential cone programming. *The Journal of Machine Learning Research*, 7:2006, 2006.

[44] S. Hochreiter and M. C. Mozer. Monaural separation and classification of mixed signals: A support-vector regression perspective. In *3rd International Conference on Independent Component Analysis and Blind Signal Separation, San Diego, CA*. Citeseer, 2001.

[45] T. Hofmann. Unsupervised learning by probabilistic latent semantic analysis. *Machine Learning*, 42(1):177–196, 2001.

[46] Baoming Hong, Godfrey D Pearlson, and Vince D Calhoun. Source density-driven independent component analysis approach for fmri data. *Human brain mapping*, 25(3):297–307, 2005.

[47] P. O. Hoyer. Non-negative sparse coding. In *Neural Networks for Signal Processing, 2002. Proceedings of the 2002 12th IEEE Workshop on*, pages 557–565, 2002.

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

[49] C. J. Hsieh and I. Dhillon. Fast coordinate descent methods with variable selection for non-negative matrix factorization. *ACM SIGKDD Internation Conference on Knowledge Discovery and Data Mining*, pages 1064–1072, 2011.

[50] Cho-Jui Hsieh, Kai-Wei Chang, Chih-Jen Lin, S. Sathiya Keerthi, and S. Sundararajan. A dual coordinate descent method for large-scale linear SVM. In *Proceedings of the 25th international conference on Machine learning*, ICML '08, pages 408–415, New York, NY, USA, 2008. ACM.

[51] Niall Hurley and Scott Rickard. Comparing measures of sparsity. *IEEE Trans. Inf. Theor.*, 55:4723–4741, October 2009.

[52] Aapo Hyvärinen and Erkki Oja. A fast fixed-point algorithm for independent component analysis. *Neural computation*, 9(7):1483–1492, 1997.

[53] Martin Jaggi. An equivalence between the lasso and support vector machines. *arXiv preprint arXiv:1303.1152*, 2013.

[54] Thorsten Joachims. Making large-scale SVM learning practical. LS8-Report 24, Universität Dortmund, LS VIII-Report, 1998.

[55] Vojislav Kecman, Michael Vogt, and Te Ming Huang. On the equality of kernel adatron and sequential minimal optimization in classification and regression tasks and alike algorithms for kernel machines. In *ESANN*, pages 215–222, 2003.

[56] S. S. Keerthi, S. K. Shevade, C. Bhattacharyya, and K. R. K. Murthy. A fast iterative nearest point algorithm for support vector machine classifier design. *Neural Networks, IEEE Transactions on*, 11(1):124–136, January 2000.

[57] D. Kim, S. Sra, and I. S. Dhillon. *A new projected quasi-newton approach for the nonnegative least squares problem.* Citeseer, 2006.

[58] Hyunsoo Kim and Haesun Park. Sparse non-negative matrix factorizations via alternating non-negativity-constrained least squares for microarray data analysis. *Bioinformatics*, 23(12):1495–1502, 2007.

[59] J. Kim and H. Park. Fast active-set-type algorithms for L1-regularized linear regression. *Proc. AISTAT*, pages 397–404, 2010.

[60] Jingu Kim and Haesun Park. Toward faster nonnegative matrix factorization: A new algorithm and comparisons. *Data Mining, IEEE International Conference on*, 0:353–362, 2008.

[61] T. F. De Laney and H. M. Kooy. *Proton and charged particle radiotherapy.* Lippincott Williams & Wilkins, 2007.

[62] CL Lawson and RJ Hanson. *Solving least squares problems, 340 pp.* Prentice-Hall, Upper Saddle River, NJ, 1974.

[63] W. H. Lawton and E. A. Sylvestre. Self modeling curve resolution. *Technometrics*, pages 617–633, 1971.

[64] Quoc V Le, Alexandre Karpenko, Jiquan Ngiam, and Andrew Y Ng. Ica with reconstruction cost for efficient overcomplete feature learning. In *NIPS*, pages 1017–1025, 2011.

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

[66] Daniel D. Lee and Sebastian H. Seung. Algorithms for non-negative matrix factorization. In *NIPS* [67], pages 556–562.

[67] Daniel D. Lee and Sebastian H. Seung. Algorithms for non-negative matrix factorization. In *NIPS*, pages 556–562, 2000.

[68] Honglak Lee, Alexis Battle, Rajat Raina, and Andrew Y. Ng. Efficient sparse coding algorithms. In Bernhard Schölkopf, John C. Platt, and Thomas Hoffman, editors, *Advances in Neural Information Processing Systems 19, Proceedings of the Twentieth Annual Conference on Neural Information Processing Systems, Vancouver, British Columbia, Canada, December 4-7, 2006*. MIT Press, 2007.

[69] Te-Won Lee, Mark Girolami, and Terrence J Sejnowski. Independent component analysis using an extended infomax algorithm for mixed subgaussian and supergaussian sources. *Neural computation*, 11(2):417–441, 1999.

[70] F. Li, Y. Yang, and E. Xing. From lasso regression to feature vector machine. *Advances in Neural Information Processing Systems*, 18:779, 2006.

[71] Hualiang Li, Tülay Adali, Nicolle Correa, Pedro A Rodriguez, and Vince D Calhoun. Flexible complex ica of fmri data. In *Acoustics Speech and Signal Processing (ICASSP), 2010 IEEE International Conference on*, pages 2050–2053. IEEE, 2010.

[72] Hualiang Li, Nicolle M Correa, Pedro A Rodriguez, Vince D Calhoun, and Tülay Adali. Application of independent component analysis with adaptive density model to complex-valued fmri data. *Biomedical Engineering, IEEE Transactions on*, 58(10):2794–2803, 2011.

[73] Liangda Li, Guy Lebanon, and Haesun Park. Fast bregman divergence nmf using taylor expansion and coordinate descent. In *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 307–315. ACM, 2012.

[74] Xi-Lin Li and Tülay Adali. Complex independent component analysis by entropy bound minimization. *Circuits and Systems I: Regular Papers, IEEE Transactions on*, 57(7):1417–1430, 2010.

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

[76] S. Luan, N. Swanson, Z. Chen, and L. Ma. Dynamic gamma knife radiosurgery. *Physics in Medicine and Biology*, 54:1579, 2009.

[77] J. Mairal, F. Bach, J. Ponce, and G. Sapiro. Online learning for matrix factorization and sparse coding. *The Journal of Machine Learning Research*, 11:19–60, 2010.

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

[79] M. J. McKeown, S. Makeig, G. G. Brown, T. P. Jung, S. S. Kindermann, A. J. Bell, and T. J. Sejnowski. Analysis of fMRI data by blind separation into independent spatial components. *Human Brain Mapping*, 6(3):160–188, 1998.

[80] Martin J McKeown, Lars Kai Hansen, and Terrence J Sejnowsk. Independent component analysis of functional mri: what is signal and what is noise? *Current opinion in neurobiology*, 13(5):620–629, 2003.

[81] M. Merritt and Y. Zhang. Interior-point gradient method for large-scale totally nonnegative least squares problems. *Journal of optimization theory and applications*, 126(1):191–202, 2005.

[82] Eric Moreau and Odile Macchi. High-order contrasts for self-adaptive source separation. *International Journal of Adaptive Control and Signal Processing*, 10(1):19–46, 1996.

[83] M. Morup and L. H. Clemmensen. Multiplicative updates for the LASSO. In *Machine Learning for Signal Processing, 2007 IEEE Workshop on*, pages 33–38. IEEE, 2007.

[84] M. Mørup and L. H. Clemmensen. Multiplicative updates for the LASSO,. *Machine Learning for Signal Processing, 2007 IEEE Workshop on*, pages 33–38, 2007.

[85] Morten Mørup, Kristoffer Hougaard Madsen, and Lars Kai Hansen. Approximate L0 constrained non-negative matrix and tensor factorization. In *ISCAS*, pages 1328–1331, 2008.

[86] Indraneel Mukherjee, Kevin Canini, Rafael Frongillo, and Yoram Singer. Parallel boosting with momentum. In *Machine Learning and Knowledge Discovery in Databases*, pages 17–32. Springer, 2013.

[87] Y. Nesterov. Efficiency of coordinate descent methods on huge-scale optimization problems. *CORE Discussion Papers*, 2010.

[88] C. L. Blake D. J. Newman and C. J. Merz. UCI repository of machine learning databases, 1998.

[89] P. D. O'Grady and S. T. Rickard. Compressive sampling of non-negative signals. In *Machine Learning for Signal Processing, 2008. MLSP 2008. IEEE Workshop on*, pages 133–138. IEEE, 2008.

[90] Paul D. O'Grady and Barak A. Pearlmutter. Convolutive non-negative matrix factorisation with a sparseness constraint. In *Proceedings of the IEEE International Workshop on Machine Learning for Signal Processing (MLSP 2006)*, pages 427–432, Maynooth, Ireland, September 2006.

[91] E. Osuna, R. Freund, and F. Girosi. An improved training algorithm for support vector machines. *Neural Networks for Signal Processing [1997] VII. Proceedings of the 1997 IEEE Workshop*, pages 276–285, September 1997.

[92] A. Pascual-Montano, J. M. Carazo, K. Kochi, D. Lehmann, and R. D. Pascual-Marqui. Nonsmooth nonnegative matrix factorization (nsNMF). *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 28(3):403–415, March 2006.

[93] R. Peharz and F. Pernkopf. Sparse nonnegative matrix factorization with l^0-constraints. *Neurocomputing*, 2011.

[94] J. Platt. Sequential minimal optimization: A fast algorithm for training support vector machines, 1998.

[95] Vamsi K Potluru, Jonathan Le Roux, Barak A Pearlmutter, John R Hershey, and Matthew E Brand. Coordinate descent for mixed-norm nmf. *NIPS Workshop: Greedy algorithms, Frank-wolfe and Friends - A modern perspective*, 2013.

[96] Vamsi K. Potluru, Sergey M. Plis, Shuang Luan, Vince D. Calhoun, and Thomas P. Hayes. Sparseness and a reduction from totally nonnegative least squares to SVM. In *Neural Networks (IJCNN), The 2011 International Joint Conference on*, pages 1922–1929, 31 2011-August 5 2011.

[97] Vamsi K. Potluru, Sergey M. Plis, Morten Morup, Vincent D. Calhoun, and Terran Lane. Efficient multiplicative updates for support vector machines. In *Proceedings of the 2009 SIAM Conference on Data Mining (SDM)*, 2009.

[98] Peter Richtárik and Martin Takáč. Distributed coordinate descent method for learning with big data. *arXiv preprint arXiv:1310.2059*, 2013.

[99] Ruslan Salakhutdinov and Sam Roweis. Adaptive overrelaxed bound optimization methods. In *Proceedings of the International Conference on Machine Learning*, volume 20, pages 664–671, 2003.

[100] M. N. Schmidt and R. K. Olsson. Single-channel speech separation using sparse non-negative matrix factorization. In *International Conference on Spoken Language Processing (INTERSPEECH)*, volume 2, page 1. Citeseer, 2006.

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

[102] B. Schölkopf and A. J. Smola. *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond (Adaptive Computation and Machine Learning)*. The MIT Press, 2001.

[103] Fei Sha, Lawrence K. Saul, and Daniel D. Lee. Multiplicative updates for large margin classifiers. In *Proceedings of the Sixteenth Annual Conference on Computational Learning Theory (COLT)*, Washington D.C., USA, 2003.

[104] Fei Sha, Lawrence K. Saul, and Daniel D. Lee. Multiplicative updates for non-negative quadratic programming in support vector machines. In Sebastian Thrun Suzanna Becker and Klaus Obermayer, editors, *Advances in Neural Information Processing Systems 15*, Cambridge, MA, 2003. MIT Press.

[105] Paris Smaragdis. Non-negative matrix factor deconvolution; extraction of multiple sound sources from monophonic inputs. *International Congress on Independent Component Analysis and Blind Signal Separation*, September 2004.

[106] Daniel A Spielman, Huan Wang, and John Wright. Exact recovery of sparsely-used dictionaries. In *Proceedings of the Twenty-Third international joint conference on Artificial Intelligence*, pages 3087–3090. AAAI Press, 2013.

[107] Fabian J Theis, Kurt Stadlthanner, and Toshihisa Tanaka. First results on uniqueness of sparse non-negative matrix factorization. In *Proceedings of the 13th European Signal Processing Conference (EUSIPCO05)*, 2005.

[108] Markus Thom and Günther Palm. Efficient sparseness-enforcing projections. *arXiv preprint arXiv:1303.5259*, 2013.

[109] R. Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society. Series B (Methodological)*, pages 267–288, 1996.

[110] Tuomas Virtanen. Monaural sound source separation by nonnegative matrix factorization with temporal continuity and sparseness criteria. *Audio, Speech, and Language Processing, IEEE Transactions on*, 15(3):1066–1074, 2007.

[111] Felix Weninger, Jordi Feliu, and Bjorn Schuller. Supervised and semi-supervised suppression of background music in monaural speech recordings. In *Acoustics, Speech and Signal Processing (ICASSP), 2012 IEEE International Conference on*, pages 61–64. IEEE, 2012.

[112] Tom White. *Hadoop: The Definitive Guide*. O'Reilly Media, original edition, June 2009.

[113] Kevin W Wilson and Bhiksha Raj. Spectrogram dimensionality reduction with independence constraints. In *Acoustics Speech and Signal Processing (ICASSP), 2010 IEEE International Conference on*, pages 1938–1941. IEEE, 2010.

[114] W. Xu, X. Liu, and Y. Gong. Document clustering based on non-negative matrix factorization. In *Proceedings of the 26th annual international ACM SIGIR confer-*

ence on Research and development in informaion retrieval, pages 267–273. ACM, 2003.

[115] Y Kenan Yılmaz, A Taylan Cemgil, and Umut Simsekli. Generalised coupled tensor factorisation. In *Advances in Neural Information Processing Systems*, pages 2151–2159, 2011.

[116] Adams Wei Yu, Hao Su, and Li Fei-Fei. Efficient euclidean projections onto the intersection of norm balls. *arXiv preprint arXiv:1206.4638*, 2012.

[117] Rafal Zdunek and Andrzej Cichocki. Fast nonnegative matrix factorization algorithms using projected gradient approaches for large-scale problems. *Computational Intelligence and Neuroscience*, page 13, 2008.

[118] E. Zeng and M. Ogihara. Nonnegative least square–A new look into SAGE data. In *Proceedings of CSB*, volume 9, 2009.

[119] Michael Zibulevsky and Barak A. Pearlmutter. Blind source separation by sparse decomposition in a signal dictionary. *Neural Computation*, 13(4):863–882, 2001.