Convex non-negative matrix factorization for massive datasets

Article *in* Knowledge and Information Systems · November 2011

DOI: 10.1007/s10115-010-0352-6

4 authors, including:

Christian Thurau
Game Analytics
65 PUBLICATIONS **1,267** CITATIONS
SEE PROFILE

Kristian Kersting
University of Bonn
304 PUBLICATIONS **3,323** CITATIONS
SEE PROFILE

Christian Bauckhage
University of Bonn
277 PUBLICATIONS **4,149** CITATIONS
SEE PROFILE

Under consideration for publication in Knowledge and Information Systems

Convex Non-negative Matrix Factorization for Massive Datasets

Christian Thurau, Kristian Kersting, Mirwaes Wahabzada, Christian Bauckhage

Fraunhofer IAIS
Schloss Birlinghoven, Sankt Augustin, Germany
Email: {christian.thurau, kristian.kersting, mirwaes.wahabzada, christian.bauckhage}@iais.fraunhofer.de

Abstract. Non-negative matrix factorization (NMF) has become a standard tool in data mining, information retrieval, and signal processing. It is used to factorize a non-negative data matrix into two non-negative matrix factors that contain basis elements and linear coefficients, respectively. Often, the columns of the first resulting factor are interpreted as "cluster centroids" of the input data and the columns of the second factor are understood to contain cluster membership indicators. When analyzing data such as collections of gene expressions, documents, or images, it is often beneficial to ensure that the resulting cluster centroids are meaningful, for instance, by restricting them to be convex combinations of data points. However, known approaches to convex NMF suffer from high computational costs and therefore hardly apply to large scale data analysis problems. This paper presents a new framework for convex NMF that allows for an efficient factorization of data matrices of millions of data points. Triggered by the simple observation that each data point can be expressed as a convex combination of vertices of the data convex hull, we require the basic factors to be vertices of the data convex hull. The benefits of *convex-hull NMF* are twofold. First, for a growing number of data points the expected size of the convex hull, i.e. the number of its vertices, grows much slower than the dataset. Second, distance preserving low-dimensional embeddings allow us to efficiently sample the convex hull and hence to quickly determine candidate vertices. Our extensive experimental evaluation on large datasets shows that convex-hull NMF compares favorably to convex NMF both in terms of speed and reconstruction quality. We demonstrate that our method can easily be applied to large-scale, real-world datasets, in our case consisting of 750,000 DBLP entries, 4,000,000 digital images, and 150,000,000 votes on World of Warcraft ® guilds, respectively.

1. Introduction

Matrix factorization is a fundamental step in many approaches to data mining, machine learning, and statistical pattern analysis. It can be found in application areas such as computational biology, computer vision, information retrieval, or social network analysis. Recent work in machine learning has focused on matrix factorizations which obey particular constraints that are inherent to certain data and therefore should be accounted for in any analysis. In particular, non-negative matrix factorization (NMF)

focuses on the analysis of data matrices whose elements are non-negative, a common occurrence in representations of text or image data among others (Chen, Rege, Dong & Hua 2008, Li 2008). Given a non-negative input matrix $\boldsymbol{V}$, NMF aims at determining two non-negative matrix factors $\boldsymbol{W}$ and $\boldsymbol{H}$ such that

$$\boldsymbol{V} \approx \boldsymbol{W}\boldsymbol{H}.$$

Convex non-negative matrix factorization (C-NMF) approaches additionally restrict the columns of $\boldsymbol{W}$ to be convex combinations of the data points gathered in $\boldsymbol{V}$. This enforces $\boldsymbol{W}$ to represent meaningful "cluster centroids" and proves to be beneficial in applications such as text or genome mining, as well as image or social network analysis.

Although several (convex) NMF approaches have been proposed, very little work exists on how to apply NMF in the wild, i.e. in situations where one has to deal with millions of data points. This is exactly the problem we address in this paper.

Our main contribution is *convex-hull* non-negative matrix factorization (CH-NMF) which we demonstrate to be a very fast and scalable convex NMF technique. Triggered by the simple observation that each data point is a convex combination of vertices of the data convex hull, the key idea is to even further restrict $\boldsymbol{W}$ and require it to only contain vertices of the convex hull. The benefits are twofold:

- First, the expected size of the convex hull typically grows considerably slower than that of the dataset. Consider for instance n random Gaussian points in the plane. Here the expected number of vertices of the convex hull is $\Omega(\sqrt{\log n})$.
- Second, distance preserving, low-dimensional projections of the data allow one to efficiently sample the data convex hull. Candidate vertices can therefore be computed efficiently.

This article extends an earlier contribution (Thurau, Kersting & Bauckhage 2009). In addition to related NMF algorithms, here we also compare the proposed method to other recent matrix factorization techniques such as the CUR or CMD decomposition introduced by Drineas, Kannan & Mahoney (2006) and Sun, Xie, Zhang & Faloutsos (2007). Moreover, we discuss the merits of incorporating the FastMap algorithm for convex hull sampling. Our extensive experimental evaluation on synthetic and large-scale real world data shows that convex-hull NMF compares favorably to existing matrix factorization methods, both in terms of speed and reconstruction quality. Moreover, we show that convex-hull NMF can be easily applied to large scale datasets. Specifically, we apply it to two large-scale, real world datasets, consisting of more than 750,000 DBLP data records, of 4 million digital images, and of 150 million votes on World of Warcraft ®guilds.

We proceed as follows: We begin by introducing definitions and notational conventions used in this article. Next, we will review NMF and related variants and then, in Section 4, we will develop convex-hull NMF. Section 5 will present our experimental evaluation on several synthetic and real-world datasets. Before concluding, we will briefly touch upon efficient methods for hull sampling.

2. Notation and Definitions

Throughout this article, vectors are written as bold lower case letters ($\boldsymbol{v}$) and their entries are denoted using subscripted lower case italics (v_k). $\mathbf{0}$ is the vector of all zeros and $\mathbf{1}$ is the vector of all ones. We write $\boldsymbol{v} \succeq \mathbf{0}$ to indicate that $v_k \geq 0$ for all k. The inner product of two vectors $\boldsymbol{u}$ and $\boldsymbol{v}$ is written as $\boldsymbol{u}^T\boldsymbol{v}$. Consequently, the expression $\mathbf{1}^T\boldsymbol{v}$ is a shorthand for $\sum_k v_k$.

Matrices are written using bold upper case letters ($\boldsymbol{M}$) and subscripted upper case italics (M_{ij}) denote individual entries. In order to indicate that $\boldsymbol{M}$ is a real-valued $m \times n$ matrix, i.e. $\boldsymbol{M} \in \mathbb{R}^{m \times n}$, we may use the shorthand $\boldsymbol{M}^{m \times n}$. If the columns of a matrix are known, we also write $\boldsymbol{M} = [\boldsymbol{m}_1, \boldsymbol{m}_2, \ldots, \boldsymbol{m}_n]$ where $\boldsymbol{m}_j \in \mathbb{R}^m$ is the jth column vector of $\boldsymbol{M}$.

In a slight abuse of notation, we may identify a finite set of vectors $\{\boldsymbol{v}_1, \ldots, \boldsymbol{v}_n\}$ with the matrix $\boldsymbol{V} = [\boldsymbol{v}_1, \ldots, \boldsymbol{v}_n]$ whose columns are given by the elements of the set. Moreover, we will simply use $\|\boldsymbol{M}\|$ to denote the *Frobenius norm* of $\boldsymbol{M}$ so that $\|\boldsymbol{M}\|^2 = \sum_{i,j} M_{ij}^2$.

A set $\mathcal{S} \subset \mathbb{R}^m$ is *convex*, if every point on the line segment between any two points in $\mathcal{S}$ is also in $\mathcal{S}$, i.e. if $\forall \boldsymbol{u}, \boldsymbol{v} \in \mathcal{S},\ \forall \lambda \in [0,1] : \lambda \boldsymbol{u} + (1-\lambda)\boldsymbol{v} \in \mathcal{S}$. A vector $\boldsymbol{v} \in \mathbb{R}^m$ is a *convex combination* of vectors $\boldsymbol{v}_1, \boldsymbol{v}_2, \ldots, \boldsymbol{v}_l \in \mathbb{R}^m$, if $\boldsymbol{v} = \sum_i \lambda_i \boldsymbol{v}_i$ where $\lambda_i \geq 0$ and $\sum_i \lambda_i = 1$. Using matrix notation, we write convex combinations as $\boldsymbol{v} = \boldsymbol{V}\boldsymbol{\lambda}$ where $\boldsymbol{V} = [\boldsymbol{v}_1, \boldsymbol{v}_2, \ldots, \boldsymbol{v}_l]$ and $\boldsymbol{\lambda} \in \mathbb{R}^l$ such that $\mathbf{1}^T\boldsymbol{\lambda} = 1$ and $\boldsymbol{\lambda} \succeq \mathbf{0}$.

An *extreme point* of a convex set $\mathcal{S}$ is any point $\boldsymbol{v} \in \mathcal{S}$ that is not a convex combination of other points in $\mathcal{S}$. That is, if $\boldsymbol{v}$ is an extreme point and $\boldsymbol{v} = \lambda \boldsymbol{u} + (1-\lambda)\boldsymbol{w}$ for $\boldsymbol{u}, \boldsymbol{w} \in \mathcal{S}$ and $\lambda \in [0,1]$, then $\boldsymbol{v} = \boldsymbol{u} = \boldsymbol{w}$.

The *convex hull* $\mathcal{C}$ of a set $\mathcal{S} \subset \mathbb{R}^m$ is the set of all convex combinations of points in $\mathcal{S}$, that is

$$\mathcal{C}(\mathcal{S}) = \left\{ \sum_{\boldsymbol{v}_i \in \mathcal{R}} \lambda_i \boldsymbol{v}_i \;\middle|\; \mathcal{R} \subseteq \mathcal{S}, |\mathcal{R}| < \infty, \sum_i \lambda_i = 1, \lambda_i \geq 0\ \forall i \right\}.$$

A *polytope* is the convex hull of finitely many points, i.e. it is the set $\mathcal{C}(\mathcal{S})$ for $|\mathcal{S}| < \infty$. The extreme points of a polytope are called *vertices*. We use $\mathcal{V}(\mathcal{S})$ to denote the set of all vertices of a polytope. Note that every point inside a polytope can be expressed as a convex combination of the points in $\mathcal{V}(\mathcal{S})$.

3. Non-Negative Matrix Factorization

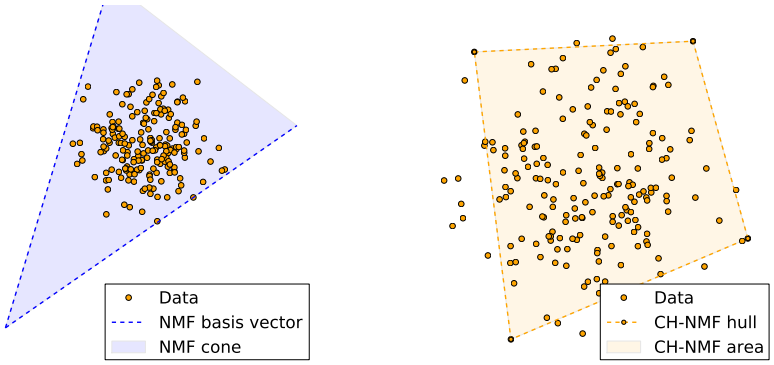
Assume an $m \times n$ input data matrix $\boldsymbol{V} = [\boldsymbol{v_1}, \ldots, \boldsymbol{v_n}] \in \mathbb{R}^{m \times n}$ that consists of n real-valued column vectors of dimensionality m. We consider factorizations of the form

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

where the matrix $\boldsymbol{W} \in \mathbb{R}^{m \times k}$ contains a set of $k \ll n$ basis vectors which are linearly combined using the coefficients in $\boldsymbol{H} \in \mathbb{R}^{k \times n}$ to represent the data.

Common approaches to achieve such a factorization include principal component analysis (Jolliffe 1986), singular value decomposition (Golub & van Loan 1996), vector quantization or non-negative Matrix Factorization (Paatero & Tapper 1994, Lee & Seung 1999). Note that these methods imposes different constraints and thus yield different matrix factors: principal component analysis constrains $\boldsymbol{W}$ to be composed of orthonormal vectors and typically produces a dense $\boldsymbol{H}$, vector quantization constrains $\boldsymbol{H}$ to unary vectors, and non-negative matrix factorization (NMF) assumes $\boldsymbol{V}$, $\boldsymbol{W}$, and $\boldsymbol{H}$ to be non-negative matrices and often leads to sparse representations of the data. Therefore, in the case of NMF, the factors in the matrix $\boldsymbol{W}$ are often easy to interpret and $\boldsymbol{H}$ tends to be sparse. From this point of view, NMF marks a middle ground between densely distributed and unary representations.

In addition to the the data-compression aspects of NMF, the intuitive interpretability of the resulting factors makes it especially interesting for data-mining. Figure 1(a) shows a geometric interpretation of NMF. Donoho & Stodden (2004) pointed out that



(a) (b)

Fig. 1. Geometric interpretations of different variants of NMF. (1(a)) The non-negativity constraints of standard NMF enforce the identification of the convex cone in which the data reside. (1(b)) CH-NMF as proposed in this article factorizes the data such that the resulting basis vectors approximate the data convex hull.

non-negative data reside in a convex cone and that a suitable set of non-negative basis vectors would span this cone. In fact, Langville, Meyer & Albright (2006) investigated several NMF algorithms and found them to yield comparable solutions (up to scale) that typically represent points on the boundary of the cone. Hoyer (2004) demonstrated that incorporating spareness constraints leads to results that more closely resemble existing data points. We also note that Vasiloglou, Gray & Anderson (2009) recently proved that for any choice of the number k of basis vectors, there exists a unique optimal solution, however, an algorithm that is guaranteed to find this solution in reasonable time is not yet available.

NMF has become an active area of research and various variants and improvements have been proposed. For example, Cai, He, Wu & Han (2008) present a matrix factorization that attempts to capture the geometric structure of the data. Kim & Park (2008) accelerate NMF using projected gradient and alternating nonnegative least squares algorithms. Another interesting variation has recently been proposed by Suvrit (2008) who bases the optimization steps in NMF on a block-iterative acceleration technique.

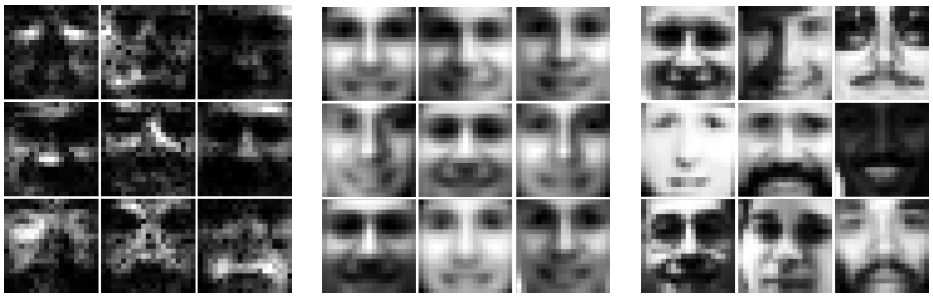
In this work, we focus on the problem of analyzing massive datasets using NMF. We build our approach on the idea of *Convex-NMF (C-NMF)* which was recently introduced by Ding, Li & Jordan (2009). Convex-NMF aims at approximating the data matrix $\boldsymbol{V}$ by means of convex combinations of data points, i.e.

$$\boldsymbol{V} = \boldsymbol{V}\boldsymbol{G}\boldsymbol{H}^T$$

where each column i of $\boldsymbol{G}$ is a stochastic vector that obeys

$$\mathbf{1}^T\boldsymbol{g}_i = 1,\ \boldsymbol{g}_i \succeq \mathbf{0}.$$

This is akin to *Archetypal Analysis* according to Cutler & Breiman (1994) where both matrices $\boldsymbol{G}$ and $\boldsymbol{H}^T$ are to be stochastic. Convex-NMF yields interesting interpretations of the data because each data point can now be expressed as a weighted sum of given data points. Consider the examples in Fig. 2 where we applied several NMF variants



(a) Standard NMF (b) Convex NMF (c) Convex hull NMF

Fig. 2. Basis vectors resulting from different NMF variants applied to the CBCL Face Database. (2(a)) Standard NMF results in sparse representations. Data points cannot be expressed as convex combinations of these basis elements. (2(b)) C-NMF yields basis elements that allow for convex combinations. Moreover, the basis vectors are "meaningful" since they closely resemble given data points. They are, however, not indicative of characteristic variations among individual samples. (2(c)) CH-NMF as proposed in this paper results in non-negative factors that represent such variations in the data (e.g. pale faces, faces with glasses, faces with beards, etc.).

to analyze the CBCL Face Database 1 which consists of 2,429 19×19 gray-scale face images[1]. As one can see, standard NMF results in part-based, sparse representations. Data points, however, do not correspond to convex combinations of these elementary parts. C-NMF, on the other hand, yields basis elements that allow for expressing data points as convex combinations of given data points. Accordingly, the "meaning" of these basis elements is intuitively understandable.

In the remainder of this section, we will briefly review C-NMF. In particular, we shall point out its runtime complexity and relevancy for data analysis. C-NMF as introduced by Ding et al. (2009) minimizes

$$J = \left\| \boldsymbol{V} - \boldsymbol{V}\boldsymbol{G}\boldsymbol{H}^T \right\|^2$$

where $\boldsymbol{V} \in \mathbb{R}^{m \times n}$, $\boldsymbol{G} \in \mathbb{R}^{n \times k}$, and $\boldsymbol{H} \in \mathbb{R}^{n \times k}$. The matrices $\boldsymbol{G}$ and $\boldsymbol{H}$ are updated iteratively until convergence using the following update rules:

$$G_{ik} = G_{ik} \sqrt{\frac{(\boldsymbol{Y}^+\boldsymbol{H})_{ik} + (\boldsymbol{Y}^-\boldsymbol{G}\boldsymbol{H}^T\boldsymbol{H})_{ik}}{(\boldsymbol{Y}^-\boldsymbol{H})_{ik} + (\boldsymbol{Y}^+\boldsymbol{G}\boldsymbol{H}^T\boldsymbol{H})_{ik}}} \tag{1}$$

$$H_{ik} = H_{ik} \sqrt{\frac{(\boldsymbol{Y}^+\boldsymbol{G})_{ik} + (\boldsymbol{H}\boldsymbol{G}^T\boldsymbol{Y}^-\boldsymbol{G})_{ik}}{(\boldsymbol{Y}^-\boldsymbol{G})_{ik} + (\boldsymbol{H}\boldsymbol{G}^T\boldsymbol{Y}^+\boldsymbol{G})_{ik}}} \tag{2}$$

where $\boldsymbol{Y} = \boldsymbol{V}^T\boldsymbol{V}$, and the matrices $\boldsymbol{Y}^+$ and $\boldsymbol{Y}^-$ are given by

$$Y_{ik}^+ = \frac{1}{2}|Y_{ik}| + Y_{ik}$$

and

$$Y_{ik}^- = \frac{1}{2}|Y_{ik}| - Y_{ik},$$

[1] http://cbcl.mit.edu/projects/cbcl/software-datasets/

respectively.

Two methods are proposed for the initialization of $\boldsymbol{G}$ and $\boldsymbol{H}$. The first one initializes both matrices to (almost) unary representations based on a k-means clustering of $\boldsymbol{V}$. The second one assumes a given NMF or Semi-NMF solution which is then refined using the above iterative algorithm. Note that in our experimental evaluation in Section 5 of this paper we only present results using the first initialization scheme since we did not find the two alternatives to yield significantly different reconstruction errors.

Convex-NMF is closely related to k-means clustering and results in similar basis vectors $\boldsymbol{W}$. However, it usually outperforms the k-means algorithm w.r.t. cluster accuracy.

Looking at the the C-NMF update rules in Eq. (1) and Eq. (2), it becomes clear that they have a time complexity of $O(n^2)$ where n is the number of data point contained in $\boldsymbol{V}$. Therefore, although there are only simple matrix multiplications at the core of the above algorithm, the size of the involved matrices quickly becomes a limiting factor, since $\boldsymbol{V}^T\boldsymbol{V}$ results in an $n \times n$ matrix. Switching to an online update rule would avoid memory issues but it would at the same time introduce additional computational overhead. Overall, we can say that C-NMF does not scale to large datasets. In the following, we will present *Convex-Hull NMF (CH-NMF)* which is a novel C-NMF method that is better suited for large-scale data analysis.

4. Convex-Hull NMF

Convex-Hull NMF aims at a factorization that incorporates data points that reside on the data convex hull. The corresponding data reconstruction has two interesting properties:

- First, the basis vectors correspond to given data points and mark, unlike in most other clustering or factorization techniques, the most extreme and not the most average data points.
- Second, any data point can be expressed as a convex and meaningful combination of these basis elements.

Both these characteristics offer interesting new opportunities for data interpretation. We exemplify this in Fig. 2 and will provide further evidence in Section 5.

Following Ding et al. (2009), we consider a non-negative factorization of the form

$$\boldsymbol{V} = \boldsymbol{V}\boldsymbol{G}\boldsymbol{H}^T \ ,$$

where $\boldsymbol{V} \in \mathbb{R}^{m \times n}, \boldsymbol{G} \in \mathbb{R}^{n \times k}, \boldsymbol{H} \in \mathbb{R}^{n \times k}$. Extending the approach by Ding et al. (2009), we further restrict the columns of $\boldsymbol{G}$ and $\boldsymbol{H}$ to convexity, i.e.,

$$\mathbf{1}^T\boldsymbol{g}_i = 1, \ \boldsymbol{g}_i \succeq \mathbf{0}$$
$$\mathbf{1}^T\boldsymbol{h}_j = 1, \boldsymbol{h}_j \succeq \mathbf{0}.$$

We note again that Ding et al. (2009) also consider convex combinations but not for the matrix $\boldsymbol{H}$. In other words, in contrast to C-NMF, our idea of CH-NMF aims at factorizing the data such that each data point can be expressed as a convex combination of convex combinations of specific data points.

The task now is to solve the following constrained quadratic problem

$$\begin{aligned} &\text{minimize} \;\; J = \left\| \boldsymbol{V} - \boldsymbol{V}\boldsymbol{G}\boldsymbol{H}^T \right\|^2 \\ &\text{subject to} \;\; \mathbf{1}^T\boldsymbol{g}_i = 1,\; \boldsymbol{g}_i \succeq \mathbf{0} \\ &\qquad\qquad \mathbf{1}^T\boldsymbol{h}_j = 1,\; \boldsymbol{h}_j \succeq \mathbf{0}. \end{aligned} \tag{3}$$

In the following, we set

$$\boldsymbol{X} = \boldsymbol{V}\boldsymbol{G}$$

so that $\boldsymbol{X} \in \mathbb{R}^{m \times k}$. The intuition behind our algorithm is as follows: due to the above properties of the coefficient matrix $\boldsymbol{G}$, the column vectors in $\boldsymbol{X}$ are convex combinations of columns in $\boldsymbol{V}$. Accordingly, by definition of the convex hull, the convex hull $\mathcal{C}(\boldsymbol{V})$ of $\boldsymbol{V}$ must contain $\boldsymbol{X}$. Obviously, we could achieve a perfect factorization of the data matrix, giving $J = 0$ in Eq. (3), by choosing the columns of $\boldsymbol{G}$ such that they would contain exactly one entry equal to 1 for each data point that is a vertex of the convex hull while all other entries were set to zero. Or more informal: *following the definition of the convex hull we can perfectly reconstruct any data point using a convex combination of data points that are vertices of the data convex hull*.

Therefore, our goal becomes to solve Eq. (3) by finding k appropriate data points on the convex hull of $\boldsymbol{V}$. In other words, we aim at solving

$$\begin{aligned} &\text{minimize} \;\; J = \left\| \boldsymbol{V} - \boldsymbol{X}\boldsymbol{H}^T \right\|^2 \\ &\text{subject to} \;\; \boldsymbol{x}_i \in \mathcal{V}(\boldsymbol{V}),\; i = 1, \ldots, k. \end{aligned} \tag{4}$$

Finding a solution to Eq. (4) is not necessarily straight forward. Rather, it is known that the worst case complexity for computing the convex hull of n data points in m dimensions is $\Theta(n^{\frac{m}{2}})$. Consequently, computing the vertices of the convex hull of large datasets quickly becomes impractical. In this paper, we therefore propose an approximate solution that subsamples the convex hull but still offers convenient data reconstruction.

Our approach exploits the fact that any data point on the convex hull of a linear lower dimensional projection of the data also resides on the convex hull in the original data dimension. Formally, since $\boldsymbol{V}$ contains finitely many points and therefore forms a polytope in $\mathbb{R}^m$, we can resort to

Theorem 1 (Main Theorem of Polytope Theory). Every image of a polytope P under an affine map $\pi : \boldsymbol{x} \rightarrow \boldsymbol{M}\boldsymbol{x} + \boldsymbol{t}$ is a polytope.

In particular, every vertex of an affine image of P, i.e., every vertex of the convex hull of the image of P, corresponds to a vertex of P. A proof of this important result can be found in the standard text by Ziegler (1995).

Therefore, computing the vertices of the convex hulls of several 2D affine projections of the data offers a way of subsampling $\mathcal{V}(\boldsymbol{V})$. This is in fact an efficient way of doing so since computing the extreme points of the convex hull of a set of 2D points can be done in $O(n \log n)$ time (de Berg, van Kreveld, Overmars & Schwarzkopf 2000).

This subsampling strategy is the main idea underlying Convex-Hull NMF, and its soundness directly follows from Theorem 1. Moreover, various methods can be used for linearly projecting the data to a 2D space:

(a) complete projections: for lower dimensional data it is possible to perform any pairwise projection of any data dimension.

(b) random projections: randomly select the dimensions to be projected.

Algorithm 1 Convex-hull NMF using eigenvector projection as a mechanism for subsampling the convex hull.

1: Compute d eigenvectors $\boldsymbol{e}_1, \ldots \boldsymbol{e}_d$ of the covariance matrix derived from the data matrix $\boldsymbol{V}^{m \times n}$

2: Project $\boldsymbol{V}$ onto the 2D-subspaces

$$\boldsymbol{E}_{o,q}^{2 \times n} = \boldsymbol{V}^T [\boldsymbol{e}_o, \boldsymbol{e}_q],\ o = 1 \ldots d,\ q = 1 \ldots d,\ o \neq q$$

3: Compute and mark convex hull vertices $\mathcal{C}(\boldsymbol{E}_{o,q})$ for each 2D projection

4: Collect marked convex hull vertices (using the original data dimensionality m)

$$\boldsymbol{S}^{m \times p} = \mathcal{C}(\boldsymbol{E}_{1,2}) \cup \ldots \cup \mathcal{C}(\boldsymbol{E}_{k-1,k})$$

5: Solve

$$\begin{aligned} \text{min } & J_S = \left\| \boldsymbol{S} - \boldsymbol{S}\boldsymbol{I}^{p \times k} \boldsymbol{J}^{k \times p} \right\|^2 \\ \text{s.t. } & \mathbf{1}^T \boldsymbol{i}_i = 1,\ \boldsymbol{i}_i \succeq \mathbf{0} \\ & \mathbf{1}^T \boldsymbol{j}_i = 1,\ \boldsymbol{j}_i \succeq \mathbf{0} \end{aligned}$$

6: Compute

$$\boldsymbol{X}^{m \times k} = \boldsymbol{S}\boldsymbol{I}$$

7: Solve

$$\begin{aligned} \text{min } & J = \left\| \boldsymbol{v}_i - \boldsymbol{X}\boldsymbol{h}_i^T \right\|^2 \\ \text{s.t. } & \mathbf{1}^T \boldsymbol{h}_i = 1,\ \boldsymbol{h}_i \succeq \mathbf{0} \end{aligned}$$

for $i = 1, \ldots, n$

(c) eigenvector projections: project the data using pairwise combinations of the first d eigenvectors of the covariance matrix of $\boldsymbol{V}$.

(d) FastMap projections: use fastmap projections as introduced by Faloutsos & Lin (1995).

Choosing PCA for linearly projecting the data to a 2D space, i.e. following option **(c)**, leads to the Convex-Hull NMF approach that is summarized in Alg. 1. The mean and covariance matrix of $\boldsymbol{V}$ can be computed iteratively and the resulting matrices of size $m \times m$ and can be efficiently stored. Projecting the data onto the $j = \frac{d(d-1)}{2}$ 2D subspaces resulting from pairwise combinations of the first d eigenvectors of the covariance matrix of $\boldsymbol{V}$ copes with major variations in the data; in our experiments, we select d such that the first d corresponding eigenvalues account for 95% of the energy of the eigenvalue spectrum. Estimates of the resulting number of sampled points can be obtained from a result by Hueter (1999), who showed that the expected size of the convex hull of n Gaussian data points in the plane is $\Omega(\sqrt{\log n})$. For Gaussian data, we thus expect to sample $p = j\sqrt{\log n}$ points; for data that can be approximated using a mixture of q Gaussian, we expect to sample $p = jq\sqrt{\log(n/q)}$ data points. In both cases, the candidate set grows much slower than n.

Give a candidate set $\boldsymbol{S} \subset \mathcal{V}(\boldsymbol{V})$ containing p vertices of the data convex hull, we now select those $k < p$ vertices that yield the best reconstruction of the remaining points in $\boldsymbol{S}$. Since $\boldsymbol{S}$ is a $m \times p$ matrix, this, too, can be formulated as a NMF optimization

problem with convexity constraints

$$\begin{aligned} \text{minimize} \quad & J_S = \left\| \boldsymbol{S} - \boldsymbol{SIJ} \right\|^2 \\ \text{subject to} \quad & \mathbf{1}^T \boldsymbol{i}_i = 1, \ \boldsymbol{i}_i \succeq \mathbf{0} \\ & \mathbf{1}^T \boldsymbol{j}_i = 1, \ \boldsymbol{j}_i \succeq \mathbf{0} \end{aligned} \tag{5}$$

where $\boldsymbol{I} \in \mathbb{R}^{p \times k}$ and $\boldsymbol{J} \in \mathbb{R}^{k \times p}$. Since $p \ll n$, solving (5) can be done efficiently using common quadratic programming routines. Note that the data dimensionality is m. The convex hull projection only served to determine a candidate set; all further computations are carried out in the original data space.

Once a suitable $\boldsymbol{I} \in \mathbb{R}^{p \times k}$ has been determined, the matrix $\boldsymbol{X}$ in Eq. (4) can be written as $\boldsymbol{X} = \boldsymbol{SI}$ which guarantees that the minimization problem in Eq. (4) is solely concerned with k data points on the convex hull of $\boldsymbol{V}$. We found that $\boldsymbol{I}$ usually results in unary representations. If this is not the case, we simply map $\boldsymbol{SI}$ to their nearest neighboring data point in $\boldsymbol{S}$.

Given $\boldsymbol{X}$, the computation of the coefficients $\boldsymbol{H}$ in Eq.4is straight forward. For smaller datasets, it is possible to use the iterative update rule from Eq. (2). However, since we do not further modify the basis vectors $\boldsymbol{X}$, we can also find an optimal solution for each data point $\boldsymbol{v}_i$ individually by minimizing $J_i = \|\boldsymbol{v}_i - \boldsymbol{X}\boldsymbol{h}_i\|^2, \mathbf{1}^T \boldsymbol{h}_i = 1, \boldsymbol{h}_i \succeq \mathbf{0}$ using common solvers. Obviously, this can be easily parallelized.

5. Experimental Evaluation

In this section, we present experimental evaluations of our CH-NMF algorithm on different datasets. Our first experimental setup compares CH-NMF to related methods. We evaluate the run-time behavior and reconstruction accuracy of CH-NMF and C-NMF, and we compare the proposed projection sampling to common random sampling techniques. For measuring the reconstruction accuracy we usually compute the Frobenius norm or the relative Frobenius norm $R = \|\boldsymbol{V} - \boldsymbol{V}'\|^2 / \|\boldsymbol{V}\|^2$, where $\boldsymbol{V}'$ denotes the reconstruction of the data matrix $\boldsymbol{V}$ using the estimated basis vectors and coefficients (the accuracy is then defined as: accuracy $= 1 - R$). We consider synthetic data and several small real-world datasets. Evaluation of C-NMF and other factorization methods is rather infeasible for very large datasets because these methods scale with $O(n^2)$, where n is the number of data. Thus, we limit the maximum number of randomly generated data to 5000 points, and use real-world datasets of even smaller size.

Since our main interest is in the analysis of very large real-world datasets, we also present experiments that evaluate CH-NMF on three challenging datasets.

- The first real-world dataset is a snapshot of the DBLP citation database[2] and we consider $757,368$ data vectors in $\mathbb{R}^{68}$.
- Our second large dataset contains 70-dimensional feature vectors and comprises $150,000,000$ player activity scores for the Massively Multiplayer Online Game World of Warcraft ®.
- The third massive dataset consists of 4,000,000 images of the Tiny Image dataset collected by Torralba, Fergus & Freeman (2008). The images are RGB color pictures of size 32×32 which are represented by means of 384-dimensional GIST feature vectors (Olivia & Torralba 2001).

[2] `http://www.informatik.uni-trier.de/ ley/db/`

(a) CBCL Face Database

# basis vectors	5	10	20	30
accuracy CH-NMF	**0.783**	**0.802**	**0.82**	**0.829**
time in sec. CH-NMF	**27.788**	**43.011**	**69.473**	**95.125**
accuracy C-NMF	0.771	0.78	0.786	0.788
time in sec. C-NMF	48.121	59.906	90.131	120.756

(b) 2000 random images from the Tiny Image dataset

# basis vectors	5	10	20	30
accuracy CH-NMF	**0.564**	**0.595**	**0.616**	**0.626**
time in sec. CH-NMF	**19.532**	**22.034**	**38.477**	**44.73**
accuracy C-NMF	0.563	0.578	0.586	0.593
time in sec. C-NMF	35.996	43.832	64.084	84.102

Table 1. Comparison of C-NMF and CH-NMF for two smaller real-word datasets. As C-NMF can not be run on very large datasets, we compare the average accuracy and runtime on the CBCL Face dataset and a selection from the Tiny Image dataset. It can be seen that both approaches yield rather similar accuracy for varying numbers of basis vectors. As expected for small datasets, CH-NMF is only slightly faster.

Our experimental evaluation is based on a basic Python implementation[3] that is running on a standard Intel-Quadcore 2.GHz computer. For the quadratic optimization steps in our implementation we apply the `cvxopt` library provided by Dahl and Vandenberghe [4]. Although CH-NMF can be easily parallelized, we used a strictly serial implementation.

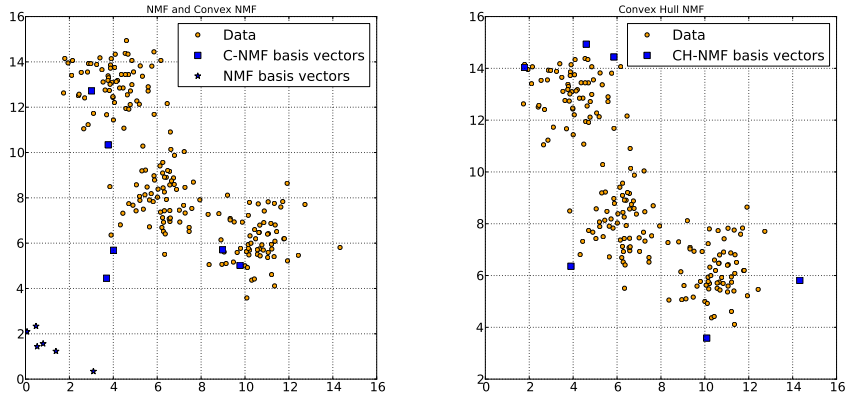
5.1. Method Comparison

Similar to the experimental procedure by Ding et al. (2009), we evaluated run-time performance using varying numbers of data points sampled from three randomly positioned 2D Gaussians (see Fig. 3). In addition to measuring computation times, we also computed the mean reconstruction accuracy.

First of all, we compared CH-NMF against C-NMF. As an additional baseline, and since it is closely connected to C-NMF, we also evaluated k-means clustering. We varied the number of data points ranging from 250 to 5500 in steps of 250. In order to account for effects of random initializations we repeated each experiment 5 times. The number of basis vectors (or clusters for k-means) was set to 6. The resulting average reconstruction accuracies can be seen in Fig. 4(a) and the observed runtime performance is shown in Fig. 4(b). Following Ding et al. (2009), we carried out 100 iterations for each approach.

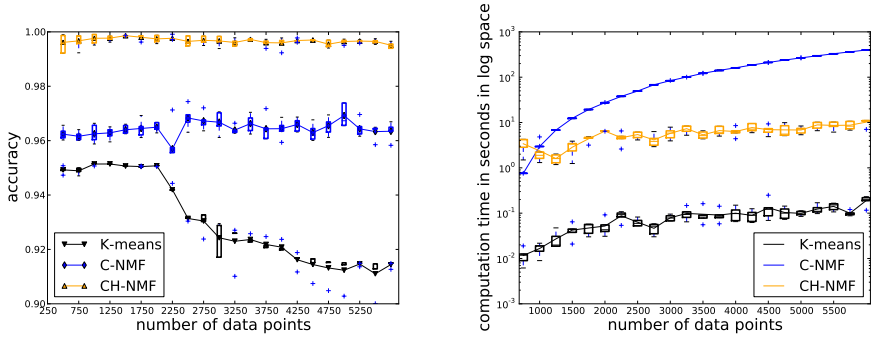
Overall, CH-NMF provides a very accurate data reconstruction. This confirms the intuition that forms the basis of our work: using vertices of the convex hull in order to reconstruct data points must lead to a near perfect reconstruction. Since we exploit

[3] `http://www-kd.iai.uni-bonn.de/index.php?page=software`
[4] `http://abel.ee.ucla.edu/cvxopt/`



(a) (b)

Fig. 3. Basis vectors produced by convex-NMF, convex-hull-NMF, and NMF for data sampled from randomly placed Gaussian distributions in 2D. It can be seen that CH-NMF basis vectors reside on the convex hull of the data whereas C-NMF usually converges against cluster centroids. NMF basis vectors, in contrast, do not correspond to actual data points, however, they could be re-scaled to reside closer to actual data points. (Best viewed in color.)



(a) (b)

Fig. 4. 4(a) Boxplots of reconstruction accuracy of k-means, C-NMF, and CH-NMF for varying numbers of synthetically generated data. As one can see, CH-NMF outperforms both other methods. 4(b) Boxplots of computation time (in log-space) for k-means, C-NMF, and CH-NMF for varying numbers of synthetically generated data. It can be seen that, for smaller datasets of up to 1000 samples, CH-NMF is actually slower than C-NMF. However, for sample size larger than 1000 CH-NMF shows significant speed-ups. The computation time remains almost constant as it mainly grows with the number of vertices of the convex hull and not with the size of the data. (Best viewed in color.)

this property and subsample the convex hull, we obtain very accurate data reconstruction on average. In comparison, C-NMF often converges to basis vectors that correspond to local modes of the data. While this is by itself a desirable property, it yields a slightly worse reconstruction. We attribute this to the that fact that C-NMF does impose a convexity constraint on the coefficients in $\boldsymbol{H}^T$. This results in a conic reconstruction

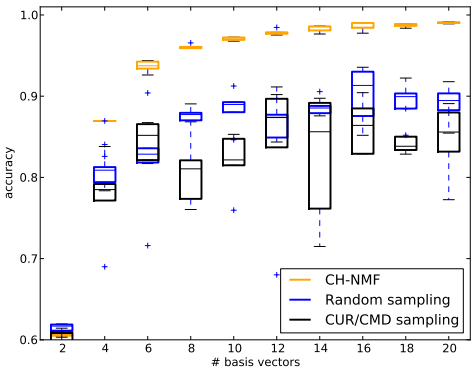


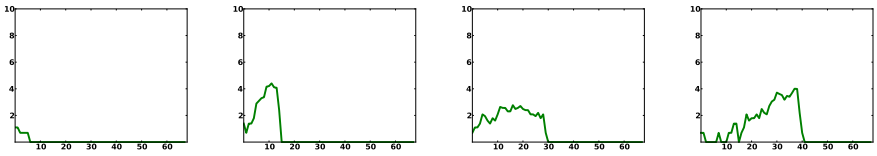
Fig. 5. Comparison of random sampling strategies to CH-NMF for the convexity constrained matrix factorization problem (averaged over 5 runs). While random sampling and CUR/CMD sometimes yield convenient results, CH-NMF is more stable and provides a higher accuracy.

that limits perfect reconstruction to data-points within the conic hull of basis vectors (Klingenberg, Curry & Dougherty 2008).

Regarding runtime performance, it can be seen that while CH-NMF initially takes longer for fewer data samples (up to 1000), its computation time increases only moderately for a growing number of data. This is due to the moderate growth of the number of vertices of the convex hull w.r.t. the total number of data points (Hueter 1999). For n data points that are distributed according to a mixture of q Gaussians, we usually have far less than $\Omega(q\sqrt{\log n})$ data point on the convex hull. Accordingly, it only takes about 54 seconds to compute basis vectors from 50,000 data points. It should be noted that about 90% of this time is spent on the constrained quadratic minimization problems $J = \left\| \boldsymbol{v}_i - \boldsymbol{X}\boldsymbol{h}_i^T \right\|^2$ which could be solved in parallel. Memory requirements for CH-NMF are rather low as it only requires to store k convex hull data points at maximum.

For comparing C-NMF and CH-NMF on real-world datasets, we use the CBCL Face Database 1 (2,429 19×19 gray-scale face images), and a random selection of 2000 images from the already mentioned Tiny Image dataset (Torralba et al. 2008). The results are summarized in Table 1. It an be seen that overall CH-NMF and C-NMF yield a very similar performance for smaller datasets. Usually, CH-NMF is faster and has a better reconstruction accuracy.

Apart from directly sampling the convex hull, one could try to reduce the computational complexity of very large datasets by employing random sampling strategies. The most simple approach would be a uniform random sampling, i.e. each data sample is picked randomly assuming a uniform distribution. While this leads to very fast sampling, the resulting factorizations will be suboptimal as it effectively ignores large parts of the data. CUR matrix decomposition, as introduced by Drineas et al. (2006) provides a smarter subsampling. In CUR decompositions the columns (or rows) of a matrix are sampled from a weighted distribution. The sampling probability for picking a column (or row) depends on its vector norm and columns with a higher norm (a high statistical leverage) have a higher sampling probability. *Compact Matrix Decomposition (CMD)* as proposed by Sun et al. (2007) further improves on CUR by avoiding double selections and is therefore faster to compute. As CMD and CUR usually yield the same accuracy (Sun et al. 2007), we use them synonymously in the following.



(a) (b) (c) (d)

Fig. 6. Basis vectors resulting from the application of CH-NMF to publication histograms extracted from DBLP. Going from (a) to (d) the histograms reflect publication frequencies at different stages of an academic career. For example, (a) could be describe a Ph.D. candidate that just started with a very few publications, and (d) describes a group leader who continuously increased his/her publication output.

For comparing the proposed convex-hull sampling to random sampling strategies, we changed the sampling to a uniform random sampling and CMD/CUR, and then applied C-NMF to the reduced dataset. The resulting basis vectors are then used for computing coefficients for the complete dataset and measuring the reconstruction quality. We sampled 500 three dimensional data points from 5 randomly placed Gaussian distributions. We first run CH-NMF using the eigenvector based sampling. As CH-NMF automatically finds a number of suitable data points to sample, we could use this for specifying the number of random samples. For the relatively small sample sizes considered here, runtime performance for the three approaches is very similar. It should be noted that for very large datasets uniform random sampling or CMD/CUR are faster than the CH-NMF sampling strategies.

Figure 5 shows the results for different numbers of basis vectors. Overall, it can be seen that CH-NMF provides the best reconstruction accuracy and is also more stable than the random sampling strategies. Interestingly, CMD/CUR show a worse performance than random sampling. The statistical leverage does not seem to be a suitable criterion for convexity constrained matrix factorization as it ignores possibly good candidates that reside on the convex hull but have a low statistical leverage.

Next, we will present experiments on large, real world datasets which are impractical for established NMF methods but can be processed in reasonable time using the proposed CH-NMF technique.

5.2. Bibliographic Analysis: DBLP

Bibliographic databases such as DBLP[5] are a rich source of information. In the experiments reported here, we are interested in the question whether there are common patterns in the development of academic careers. To this aim, we extracted from DBLP the cumulative publication histograms of $757,368$ authors (see Fig. 6). A publication histogram consists of the number of publications listed in DBLP in her first year, second year, and so on. We have cumulated the publications numbers of the years. The longest histogram we found spanned a period of 68 years. To get curves of equal length, we filled missing years with zeros. Following Aitchison (1982), we use logarithmic histogram values in our analysis. The idea is that the publication histogram of an author provides a good characterization of her activity and also, to some, extend of her success (but of course does not suggest impact or quality).

[5] `http://www.informatik.uni-trier.de/ ley/db/`

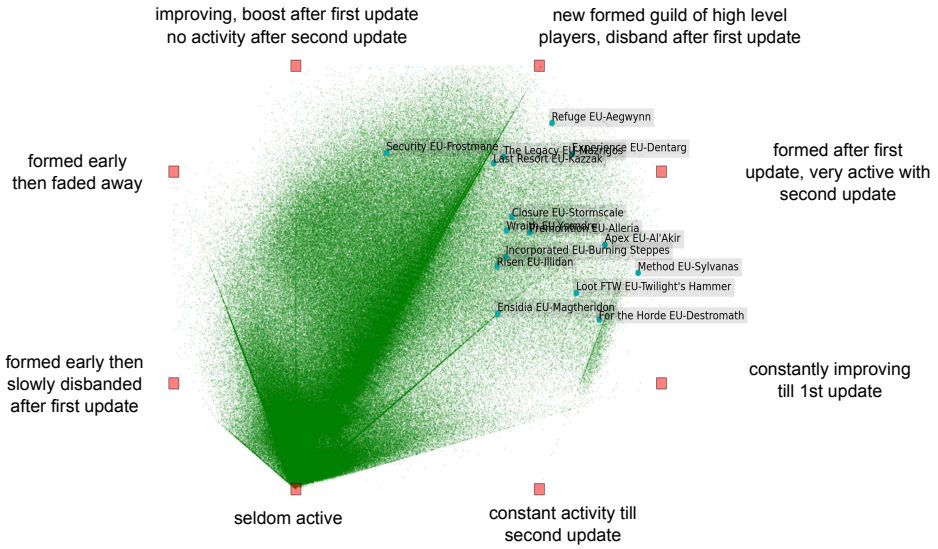


Fig. 7. The world of World of Warcraft ®. The figure shows a 2D projection of 150 million votes for guilds (green points) into the space spanned by the convex-hull NMF bases (red boxes). Intuitively understandable interpretations of the basis elements are shown right next to them. The names of 13 of the 20 most successful guilds (according to `http://www.wowprogress.com/`) are shown inside the plot. Computing the factorization and the embedding took less than 1.5 hours on a standard desktop machine.

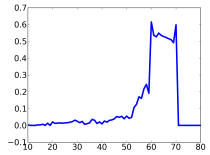
Application of CH-NMF to publication histograms yields interesting basis histograms, which can be seen in Figure 6. Again, it is important to note that the resulting basis vectors correspond to real data or, in this particular case, to real people. Other publication records can now be easily expressed as a convex combination of the most extreme academic career profiles.

5.3. Activity Mining: World of Warcraft ®

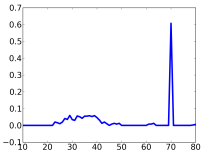
This dataset consists of recordings of the online appearance of characters in the computer game World of Warcraft ®. To the best of pour knowledge, this is the first time that vast recordings of in-game data of Massively Multiplayer Online games are considered as a source for data mining, we will briefly describe the data in more detail.

At the time of this writing, World of Warcraft ®has more than 12 million p(l)aying customers. The game takes place in a virtual medieval fantasy environment. However, instead of one persistent world, players are distributed among different so called realms. These realms exist in parallel and can have slightly varying rule-sets, i.e. each realm is its own world. In the US and Europe there exist about 500 unique realms. World of Warcraft ®is often considered one large social platform which is used for chatting, team-play, and gathering. Compared to well known virtual worlds that mainly serve as chat platforms such as, for example, Second-life ®, World of Warcraft ®is probably the *real* second life as it has a larger and more active (paying) user base. Moreover, a whole industry is developing around World of Warcraft ®. It is estimated that 400,000 people world-wide are employed as gold-farmers who are collecting virtual goods for online games and sell them over the Internet.

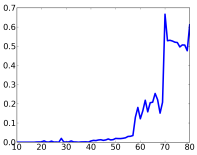
Players organize in groups, which are called guilds. Unlike groups known from



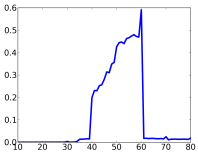
(a) improving, boost with 1st update no activity with 2nd update



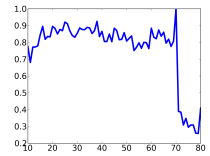
(b) new formed guild of mostly high level players, disband after first update



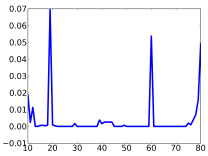
(c) formed after first update, boost with second update



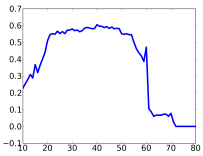
(d) constant improve till 1st update



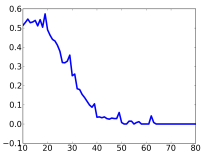
(e) high activity till 2nd update



(f) seldom active



(g) formed early then slowly disbanded after first update



(h) formed early then faded away

Fig. 8. Basis vectors resulting from applying CH-NMF to the World of Warcraft ® data. Each subplot shows a level-guild histogram. The x-axis denotes histogram bins corresponding to player levels. Player levels describe a character's experience and may increase over time. Each bin contains accumulated player votes in log-space. By definition of CH-NMF, the basis vectors show the most extreme data-points. Accordingly, one can easily provide intuitive explanations of the determined basis vectors. For example, 8(c) shows a guild that mainly consists of high level players whose experience ranges from level 70 to 80. Therefore, we can assume that the guild was formed after the 2nd update, when the level cap was raised to 80, and then stayed together. 8(h), as a second example, shows high activity of players of lower levels (up to level 30). Here, we can assume that the guild was formed early after the release of the game and then disbanded.

	Basis vector coefficients							
	Fig. 8(a)	Fig. 8(b)	Fig. 8(c)	Fig. 8(d)	Fig. 8(e)	Fig. 8(f)	Fig. 8(g)	Fig. 8(h)
The Legacy	0.313	0.185	0.33	0.119	0.013	0.0	0.04	0.0
Closure	0.0	0.4	0.325	0.0	0.0	0.275	0.0	0.0
Refuge	0.139	0.373	0.455	0.0	0.004	0.014	0.0	0.016
Risen	0.042	0.144	0.454	0.001	0.0	0.359	0.0	0.0
Wraith	0.306	0.0	0.31	0.23	0.089	0.0	0.065	0.0
Security	0.625	0.021	0.146	0.08	0.051	0.0	0.077	0.0
Ensidia	0.0	0.0	0.552	0.0	0.0	0.448	0.0	0.0
Apex	0.084	0.0	0.634	0.072	0.21	0.0	0.0	0.0
Premonition	0.16	0.013	0.545	0.0	0.181	0.0	0.102	0.0
For the Horde	0.053	0.0	0.396	0.2	0.35	0.0	0.0	0.0
Loot FTW	0.091	0.0	0.428	0.169	0.275	0.0	0.038	0.0
Method	0.044	0.0	0.485	0.412	0.059	0.0	0.0	0.0
Experience	0.036	0.318	0.555	0.0	0.0	0.058	0.007	0.026
Last Resortk	0.119	0.499	0.202	0.007	0.0	0.174	0.0	0.0
Incorporated	0.0	0.237	0.417	0.0	0.0	0.346	0.0	0.0

Table 2. Basis vector coefficients for a number of selected World of Warcraft ® guilds (the corresponding basis vectors are explained in detail in Fig. 8). These guilds are a selection from the top 20 guilds worldwide. Their coefficients in a basis representation resulting from CH-NMF tend to be similar; they all show a strong tendency towards the basis vector in Fig. 8(c) "*formed after first update, very active with second update*" just as it is indicated in the projection in Fig. 7.

other social platforms or services such as Facebook or Flickr, membership in a guild is exclusive. Obviously, their choice of a guild influences with whom players frequently interact. It also influences how successful players are in terms of in-game achievements, for instance, how likely they are to obtain better equipment or rare items.

The data we consider here was crawled from a publicly accessible web site[6]. In the following, we interpret each online appearance of a character as a vote. Character observations span a period of 4 years. Every time a character is seen online, (s)he votes for the guild (s)he is a member of; votes are cast according to a player's level. We accumulate the votes into a level-guild histogram, ranging from level 10 (levels 1–9 are excluded) to level 80 (the highest possible level). Players advance in level by engaging in in-game activity, i.e. by completing quests or other heroic deeds.

We assume that the level distribution among a guild is a good descriptor for its success. For example, a guild of very experienced level 80 characters has a higher chance of great achievements than a guild of level 10 players. Also, a level histogram provides an indicator of player activity over time. If players are continuously staying with a particular guild, we expect an equally distributed level histogram, as the characters increase their levels over time. Again following Aitchison (1982), we use logarithmic histogram values in our analysis. In total, we collected 150 million votes of 18 million characters belonging to 1.4 million guilds.

Application of CH-NMF to the World of Warcraft ®data reveals interesting structures in the data. Fig. 7 shows a projection of all level-guild histograms into the space spanned by the CH-NMF basis vectors. We decided for 8 basis vectors since we assumed that one basis vector would function as a unity vector and encode a range of 10 levels. However, the actual outcome differs from what we expected. Figure 8 shows intuitive explanations of the the basis vectors we obtained. They provide an interpretation of the distribution of all World of Warcraft ®guilds; in particular, it is interesting to note that the majority of guilds is very close to the basis vector in Fig. 8(f). Apparently, the vast majority of guilds or players seem to be casual gamers. Contrarily, none of the most successful guilds appear to belong to that group. Also, notice that we can spot particular events in the history of the game just from looking at the basis level-guild histograms. Considerable updates of the game content (a regular procedure that makes novel content available and also allows a further advancement in character level) appear to entail a restructuring of social groups.

Regarding runtime performance, the application of CH-NMF took 57 minutes (this includes the complete algorithm described in Fig. 1). The computation of 8 basis vectors only took 7.5 minutes. As expected, most time was spent on computing the 1.4 million basis vector coefficients, and on the computation of level-guild histograms (about 20 minutes).

5.4. Image Retrieval: Large-scale Image Collection

The third massive dataset that we considered in our experiments consists of 4 million images downloaded from the Internet (Torralba et al. 2008). The images are re-scaled to 32×32 pixels in 3 color channels and are represented by means of 384-dimensional GIST feature vectors. A selection of 25 basis vectors obtained from CH-NMF can be seen in Fig. 9. Interestingly, some of the basis vectors found among the tiny images bear a geometric similarity to 2D Fourier basis functions or Gabor filters. In fact, this stood

[6] http://www.warcraftrealms.com

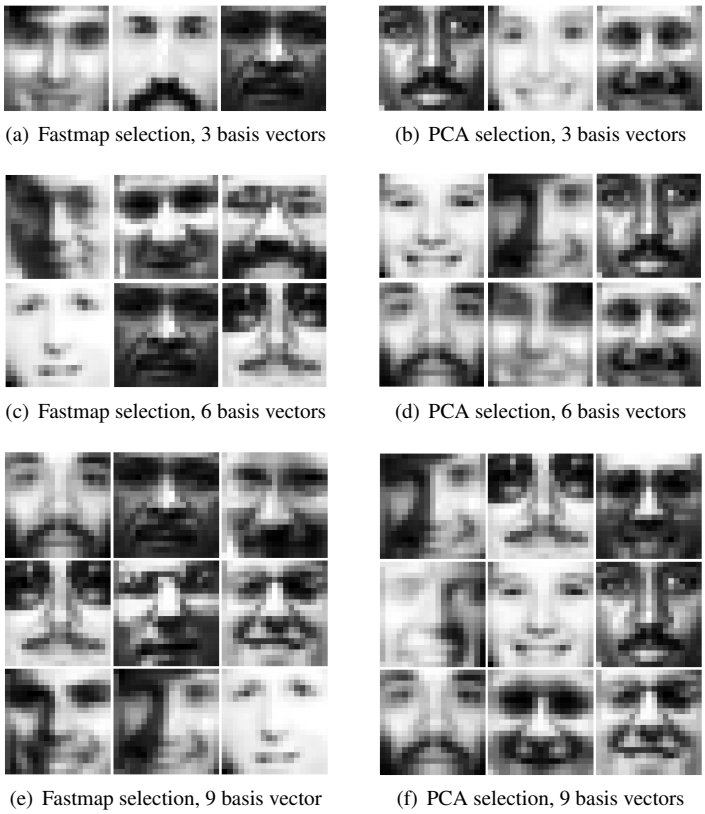
Fig. 9. 25 basis vectors (shown as images) that result from applying CH-NMF to a dataset of 4.000.000 tiny images. Interestingly, the structure of these basis images resembles elementary Fourier basis functions or Gabor filters. (Best viewed in color.)

to be expected, because the GIST features introduced by Olivia & Torralba (2001) can be seen as a frequency domain representation of the input images. Accordingly, those images whose geometric structure is most similar to superpositions of elementary 2D sine or cosine functions form the extreme points in this space. Using larger numbers of basis vectors added more and more structured images to the set of basis images.

Due to the higher dimension of the data considered in this experiment, the application of CH-NMF took longer than in the case of the World of Warcraft ®data. Using the idea of FastMap projections for an efficient pre-selection of candidate vectors for CH-NMF, it took about 44 hours on a single computer to determine the 25 basis elements shown in Fig. 9. This is, in fact, a remarkable feat as the data was stored on an external USB hard drive and runtime could be considerably reduced using an internal hard disk.

6. Sampling the Convex Hull

For massive, high-dimensional data, computing the covariance matrix can take a lot of time. In these cases, we use the FastMap projections to sample vertices from the data convex hull. We briefly summarize the algorithm in the following. FastMap computes a u-dimensional Euclidean embedding and proceeds as follows. Given pairwise distances among the columns of the data matrix $\boldsymbol{V}$, we select a pair of distant vectors called *pivot objects*. The line segment between the pivot objects serves as the first coordinate axis. For each data point $\boldsymbol{v}$, we determine its coordinate value along this axis by projecting $\boldsymbol{v}$ onto this line. Next, the pairwise distances of all objects are updated to reflect this projection, i.e. we compute the pairwise distances among the data points in the subspace orthogonal to the line. This process is repeated until, after u iterations, we obtain the u coordinates as well as the u-dimensional representation of all data points. Ostrouchov & Samatova (2005) have shown that the pivots are taken from the faces, usually vertices, of



(a) Fastmap selection, 3 basis vectors (b) PCA selection, 3 basis vectors

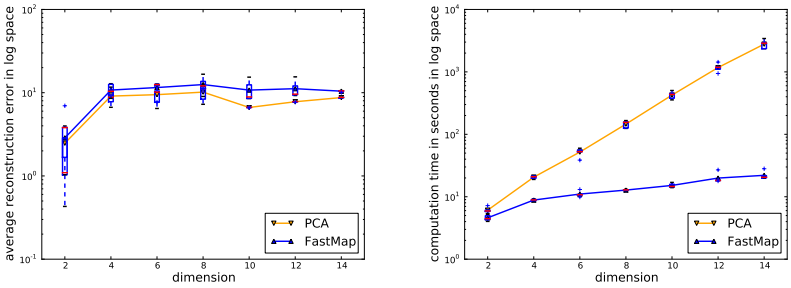
(c) Fastmap selection, 6 basis vectors (d) PCA selection, 6 basis vectors

(e) Fastmap selection, 9 basis vector (f) PCA selection, 9 basis vectors

Fig. 10. Comparison of basis vectors resulting from using Fastmap and PCA for sampling convex hull vertices. While the resulting basis images differ between the to methods, for each image in the left column of the figure we can identify a very similar counterpart in the right column.

the convex hull of the data points in the original implicit Euclidean space. This justifies the idea to employ FastMap in step (3) of the CH-NMF algorithm discussed in Section 4. An experimental comparison of the PCA an FastMap based subsampling strategies is shown in Fig. 10 and Fig. 11.

First, we compared CH-NMF with PCA against CH-NMF with FastMap on synthetic data. We evaluated the mean reconstruction error and run-time performance (see Fig. 11) using ten randomly positioned Gaussians in 2D with varying number of dimensions (ranging between 2 and 14) and a total number of 4000 data points. We repeated each experiment 5 times to account for effects of random initialization. The number of basis vectors was set to the number of dimensions +1. As can be seen, using Fastmap for candidate selection is not only several orders of magnitude faster than sampling based on PCA, it also manages to achieve comparable reconstruction errors. As a second test, we provide a qualitative evaluation of the resulting basis vectors. Using the CBCL face database, we considered candidate sets obtained from Fastmap and PCA for computing CH-NMF. Figure 10 shows the resulting bases. It can be seen that, while the resulting bases are not identical, they show a high degree of similarity.



(a) (b)

Fig. 11. Results of using different sampling strategies on synthetic data. While the average reconstruction error is almost identical for Fastmap and PCA (Fig. 11(a)), Fastmap subselection is generally much faster to compute (Fig. 11(b))

7. Conclusions

Machine learning and data mining techniques typically consists of two parts: the model and the data. Most effort in recent years has gone into the modeling part. Large-scale datasets, however, allow one to move into the opposite direction (Torralba et al. 2008, Halevy, Norvig & Pereira 2009, Aggarwal 2009) and ask the question of how much can the data itself help us to solve the problem. This direction is particularly appealing given that the Internet nowadays offers a plentiful supply of large-scale datasets for many challenging tasks.

Motivated by this, we have presented a data-driven NMF approach, called *convex-hull NMF*, that is fast and scales well. The key idea is to restrict the "clusters" to be combinations of vertices of the convex hull of the dataset and thus directly explore the data itself in order to solve the convex NMF problem. Our experimental results reveal that convex-hull NMF can indeed effectively extract meaningful "clusters" from datasets containing millions of images and rating.

For future work it is interesting to apply convex-hull NMF to other challenging datasets, such as Wikipedia, Netflix, Facebook, or the blogosphere, and to use it for applications such as collaborative filtering. To improve performance, one could explore mixtures of convex hulls to deal with general distributions. Also, an analysis as to how outliers effect the performance on small and medium size datasets should be done, as our results indicate that this is not an issue for very large datasets. Finally, a comparison to max-margin matrix factorization (Srebro, Rennie & Jaakola 2005, Rennie & Srebro 2005), which compute a low-norm instead of a low-rank factorization, would be interesting. Overall our experimental results are an encouraging sign that applying NMF techniques in the wild, i.e. on millions of data points may not be insurmountable.

Acknowledgements

The authors would like to thank A. Torralba, R. Fergus, and W. T. Freeman for making the tiny images freely available. Kristian Kersting and Mirwaes Wahabzada were supported by the Fraunhofer ATTRACT Fellowship STREAM.

References

Aggarwal, C. (2009), 'On Classification and Segmentation of Massive Audio Data Streams', *Knowledge and Information Systems* **20**(2), 137–156.

Aitchison, J. (1982), 'The Statistical Analysis of Compositional Data', *J. of the Royal Statistical Society B* **44**(2), 139–177.

Cai, D., He, X., Wu, X. & Han, J. (2008), Non-negative Matrix Factorization on Manifold, *in* 'Proc. IEEE Int. Conf. on Data Mining'.

Chen, Y., Rege, M., Dong, M. & Hua, J. (2008), 'Non-negative Matrix Factorization for Semi-supervised Data Clustering', *Knowledge and Information Systems* **17**(3), 355–379.

Cutler, A. & Breiman, L. (1994), 'Archetypal Analysis', *Technometrics* **36**(4), 338–347.

de Berg, M., van Kreveld, M., Overmars, M. & Schwarzkopf, O. (2000), *Computational Geometry*, Springer.

Ding, C., Li, T. & Jordan, M. (2009), 'Convex and Semi-Nonnegative Matrix Factorizations', *IEEE Trans. on Pattern Analysis and Machine Intelligence* **32**(1), 45–55.

Donoho, D. & Stodden, V. (2004), When Does Non-negative Matrix Factorization Give a Correct Decomposition into Parts?, *in* 'Advances in Neural Information Processing Systems 16', MIT Press.

Drineas, P., Kannan, R. & Mahoney, M. (2006), 'Fast Monte Carlo Algorithms III: Computing a Compressed Approixmate Matrix Decomposition', *SIAM J. Computing* **36**(1), 184–206.

Faloutsos, C. & Lin, K.-I. (1995), FastMap: A Fast Algorithm for Indexing, Data-mining and Visualization of Traditional and Multimedia Datasets, *in* 'Proc. ACM SIGMOD Conf.'.

Golub, G. & van Loan, J. (1996), *Matrix Computations*, 3rd edn, Johns Hopkins University Press.

Halevy, A., Norvig, P. & Pereira, F. (2009), 'The Unreasonable Effectiveness of Data', *IEEE Intelligent Systems* **24**(2), 8–12.

Hoyer, P. (2004), 'Non-Negative Matrix Factorization with Sparseness Constraints', *J. of Machine Learning* **5**(Dec.), 1457–1469.

Hueter, I. (1999), 'Limit Theorems for the Convex Hull of Random Points in Higher Dimensions', *Trans. of the American Mathematical Society* **351**(11), 4337–4363.

Jolliffe, I. (1986), *Principal Component Analysis*, Springer.

Kim, J. & Park, H. (2008), Toward Faster Nonnegative Matrix Factorization: A New Algorithm and Comparisons, *in* 'Proc. IEEE Int. Conf. on Data Mining'.

Klingenberg, B., Curry, J. & Dougherty, A. (2008), 'Non-negative Matrix Factorization: Ill-posedness and a Geometric Algorithm', *Pattern Recognition* **42**(5), 918 – 928.

Langville, A., Meyer, C. & Albright, R. (2006), Initializations for the Nonnegative Matrix Factorization, *in* 'Proc. ACM Int. Conf. on Knowledge Discovery and Data Mining'.

Lee, D. D. & Seung, H. S. (1999), 'Learning the Parts of Objects by Non-negative Matrix Factorization', *Nature* **401**(6755), 788–799.

Li, T. (2008), 'Clustering Based on Matrix Approximation: a Unifying View', *Knowledge and Information Systems* **17**(1), 1–15.

Olivia, A. & Torralba, A. (2001), 'Modeling the Shape of the Scene: A Holistic Representation of the Spatial Envelope', *Int. J. of Computer Vision* **42**(3), 145–175.

Ostrouchov, G. & Samatova, N. (2005), 'On FastMap and the Convex Hull of Multivariate Data: Toward Fast and Robust Dimension Reduction', *IEEE Trans. on Pattern Analysis and Machine Intelligence* **27**(8), 1340–1434.

Paatero, P. & Tapper, U. (1994), 'Positive Matrix Factorization: A Non-negative Factor Model with Optimal Utilization of Error Estimates of Data Values', *Environmetrics* **5**(2), 111–126.

Rennie, J. & Srebro, N. (2005), Fast Maximum Margin Matrix Factorization for Collaborative Prediction, *in* 'Proc. Int. Conf. on Machine Learning'.

Srebro, N., Rennie, J. M. & Jaakola, T. (2005), Maximum-Margin Matrix Factorization, *in* 'Advances in Neural Information Processing Systems 17', MIT Press.

Sun, J., Xie, Y., Zhang, H. & Faloutsos, C. (2007), Less is More: Compact Matrix Decomposition for Large Sparse Graphs, *in* 'Proc. SIAM Int. Conf. on Data Mining'.

Suvrit, S. (2008), Block-Iterative Algorithms for Non-negative Matrix Approximation, *in* 'Proc. IEEE Int. Conf. on Data Mining'.

Thurau, C., Kersting, K. & Bauckhage, C. (2009), Convex Non-Negative matrix Factorization in the Wild, *in* 'Proc. IEEE Int. Conf. on Data Mining'.

Torralba, A., Fergus, R. & Freeman, W. T. (2008), '80 Million Tiny Images: A Large Data Set for Non-parametric Object and Scene Recognition', *IEEE Trans. on Pattern Analysis and Machine Intelligence* **30**(11), 1958–1970.

Vasiloglou, N., Gray, A. & Anderson, D. (2009), Non-Negative Matrix Factorization, Convexity and Isometry, *in* 'Proc. SIAM Int. Conf. on Data Mining'.

Ziegler, G. (1995), *Lectures on Polytopes*, Springer.