LOW DIMENSIONAL POLYTOPE APPROXIMATION AND ITS APPLICATIONS TO NONNEGATIVE MATRIX FACTORIZATION

DRAFT AS OF January 15, 2007

MOODY T. CHU* AND MATTHEW M. LIN†

Abstract. In this study, nonnegative matrix factorization is recast as the problem of approximating a polytope on the probability simplex by another polytope with fewer facets. Working on the probability simplex has the advantage that data are limited to a compact set with known boundary, making it easier to trace the approximation procedure. In particular, the supporting hyperplane that separates a point from a disjoint polytope, a fact asserted by the Hahn-Banach theorem, can be calculated in finitely many steps. This approach leads to a convenient way of computing the proximity map which, in contrast to most existing algorithm where only an approximate map is used, finds the unique and global minimum per iteration. This paper sets up a theoretical framework, outlines a numerical algorithm and suggests an effective implementation. Testing results strongly evidence that this approach obtains better low rank nonnegative matrix approximation in fewer steps than conventional methods.

AMS subject classifications. 41A36, 65D18, 46A22, 90C25, 65F30

Key words. nonnegative matrix factorization, polytope approximation, probability simplex, supporting hyperplane, Hahn-Banach theorem

1. Introduction.

The problem of nonnegative matrix factorization (NMF) arises in a large variety of disciplines in sciences and engineering. Its wide range of important applications such as text mining, chemoinformatics, factor retrieval, image articulation, dimension reduction and so on has attracted considerable research efforts. Many different kinds of NMF techniques have been proposed in the literature, notably the popular Lee and Seung iterative update algorithm [16, 17]. Successful applications to various models with nonnegative data values are abounding. We can hardly be exhaustive by suggesting [6, 11, 12, 13, 18, 19, 20, 21, 23] and the references contained therein as a partial list of interesting work. The review article [26] contains many other more recent applications and references.

The basic idea behind NMF is the typical linear model,

$$Y = AF, \tag{1.1}$$

where $Y = [y_{ij}] \in \mathbb{R}^{m\times n}$ denotes the matrix of "observed" data with y_{ij} representing, in a broad sense, the *score* obtained by entity j on variable i, $A = [a_{ik}] \in \mathbb{R}^{m\times p}$ is a matrix with a_{ik} representing the *loading* of variable i from factor k or, equivalently, the *influence* of factor k on variable i, and $F = [f_{kj}] \in \mathbb{R}^{p\times n}$ with f_{kj} denoting the *score* of factor k by entity j or the *response* of entity j to factor k. The particular emphasis in NMF is that all entries of the matrices are required to be nonnegative. To provide readers with a little bit motives on the study of NMF, we briefly outline two applications below. A good survey of other interesting NMF applications can be found in [26, Section 6.2].

The *receptor model* is an observational technique commonly employed by the air pollution research community which makes use of the ambient data and source profile data to apportion sources or source categories [10, 11, 14, 25]. The fundamental principle in this model is the conservation of masses. Assume that there are p sources which contribute m chemical species to n samples. The *mass balance*

*Department of Mathematics, North Carolina State University, Raleigh, NC 27695-8205. (chu@math.ncsu.edu) This research was supported in part by the National Science Foundation under grants CCR-0204157 and DMS-0505880, and NIH Roadmap for Medical Research grant 1 P20 HG003900-01.

†Department of Mathematics, North Carolina State University, Raleigh, NC 27695-8205. (mlin@ncsu.edu) This research was supported in part by the National Science Foundation under grant DMS-0505880.

equation within this system can be expressed via the relationship,

$$y_{ij} = \sum_{k=1}^{p} a_{ik} f_{kj}, \tag{1.2}$$

where y_{ij} is the elemental concentration of the ith chemical measured in the jth sample, a_{ik} is the gravimetric concentration of the ith chemical in the kth source, and f_{kj} is the airborne mass concentration that the kth source has contributed to the jth sample. In a typical scenario, only values of y_{ij} are observable whereas neither the sources are known nor the compositions of the local particulate emissions are measured. Thus, a critical question is to estimate the number p, the compositions a_{ik}, and the contributions f_{kj} of the sources. For physical feasibility, the source compositions a_{ik} and the source contributions f_{kj} must all be nonnegative. The identification and apportionment, therefore, becomes a nonnegative matrix factorization problem of Y.

In another application, NMF has been suggested as a way to identify and classify intrinsic "parts" that make up the object being imaged by multiple observations. More specifically, each column $\mathbf{y}_j$ of a nonnegative matrix Y now represents m pixel values of one image. The columns $\mathbf{a}_k$ of A are basis elements in $\mathbb{R}^m$. The columns of F, belonging to $\mathbb{R}^p$, can be thought of as coefficient sequences representing the n images in the basis elements. In other words, the relationship,

$$\mathbf{y}_j = \sum_{k=1}^{p} \mathbf{a}_k f_{kj}, \tag{1.3}$$

can be thought of as that there are *standard parts* $\mathbf{a}_k$ in a variety of positions and that each image $\mathbf{y}_j$ is made by superposing these parts together in some ways. Those parts, being images themselves, are necessarily nonnegative. The superposition coefficients, each part being present or absent, are also necessarily nonnegative.

In either case above and in many other contexts of applications, we see that the p factors, interpreted as either the sources or the basis elements, play a vital role. In practice, there is a need to determine as fewer factors as possible and, hence, a low rank nonnegative matrix approximation of the data matrix Y arises. The mathematical problem can be stated as follows:

(NMF) *Given a nonnegative matrix $Y \in \mathbb{R}^{m\times n}$ and a positive integer $p < \min\{m, n\}$, find nonnegative matrices $U \in \mathbb{R}^{m\times p}$ and $V \in \mathbb{R}^{p\times n}$ so as to minimize the functional*

$$f(U, V) := \frac{1}{2}\|Y - UV\|_F^2. \tag{1.4}$$

It has been argued that NMF can be interpreted geometrically as the problem of finding a simplicial cone which contains a cloud of data points located in the first orthant [6]. Further extending that thought, this paper recasts NMF as the problem of approximating a polytope on the probability simplex by another polytope which has fewer facets. Our basic idea follows from computational geometry where a complex surface is to be approximated by a simpler one. In our particular setting, the complex surface refers to the boundary of a convex polytope of n points in $\mathbb{R}^m$ whereas the simpler surface refers to that of p points in the same space. A unique feature in our approach is that the approximation is to take place on the probability simplex. We shall exploit the fact that the probability simplex, being a compact set with well distinguishable boundary, makes the optimization procedure easier to manage.

This paper actually deals with two related but independent problems. The first problem considers the polytope approximation only on the probability simplex. The underlying geometry is easy to understand, but the problem is of interest in itself. More importantly, with slight modifications the idea lends it geometric characteristics naturally to the more difficult NMF problem.

We organize our presentation as follows. Beginning in section 2, we briefly describe how the original NMF can be formulated as a low dimensional polytope approximation through scaling. When limited to the probability simplex, we introduce in section 3 two basic mechanisms for polytope approximation — a recursive algorithm which projects a given point onto a prescribed polytope and a descent method which slides points along the boundary of the probability simplex to improve the objective value. We emphasize that the recursive algorithm determines the unique and global proximity map in finitely many steps and thus enables our NMF method, which will be discussed in section 4, to obtain much better, generally of several order, improvement over the well know Lee-Seung updating algorithm [17] per iterative step.

2. Pull-back to the Probability Simplex. For convenience, denote a nonnegative vector $\mathbf{y}$ by $\mathbf{y} \succeq 0$. Given a nonnegative matrix $Y = [\mathbf{y}_1, \ldots, \mathbf{y}_n] \in \mathbb{R}^{m\times n}$ with nonzero columns $\mathbf{y}_k \in \mathbb{R}^m$, define the *scaling factor* $\sigma(Y)$ by

$$\sigma(Y) := \operatorname{diag}\left\{\|\mathbf{y}_1\|_1, \ldots, \|\mathbf{y}_n\|_1\right\}, \tag{2.1}$$

where $\|\cdot\|_1$ stands for the 1-norm of a vector, and the *pull-back map* $\vartheta(Y)$ by

$$\vartheta(Y) := Y\sigma(Y)^{-1}. \tag{2.2}$$

Each column of $\vartheta(Y)$ can be regarded as a point on the $(m-1)$-dimensional *probability simplex* $\mathcal{D}_m$ defined by

$$\mathcal{D}_m := \left\{\mathbf{y} \in \mathbb{R}^m | \mathbf{y} \succeq 0, \mathbf{1}_m^\top \mathbf{y} = 1\right\}, \tag{2.3}$$

where $\mathbf{1}_m = [1, \ldots, 1]^\top$ stands for the vector of all 1's in $\mathbb{R}^m$.

For each given nonnegative matrix $Y \in \mathbb{R}^{m\times n}$, there is a smallest convex polytope $\mathcal{C}(Y)$ on $\mathcal{D}_m$ containing all columns of $\vartheta(Y)$. Indeed, the vertices of $\mathcal{C}(Y)$ consist of a subset of columns of $\vartheta(Y)$. If we denote this fact by

$$\mathcal{C}(Y) := \operatorname{conv}(\vartheta(Y)) = \operatorname{conv}(\vartheta(\widetilde{Y}))$$

where $\widetilde{Y} = [\mathbf{y}_{i_1}, \ldots, \mathbf{y}_{i_p}] \in \mathbb{R}^{m\times p}$ is a submatrix of Y, then every column of $\vartheta(Y)$ is a convex combination of columns of $\vartheta(\widetilde{Y})$ and we can write

$$\vartheta(Y) = \vartheta(\widetilde{Y})Q, \tag{2.4}$$

where $Q \in \mathbb{R}^{p\times n}$ itself represents p points in the simplex $\mathcal{D}_p$. Together, we have obtained

$$Y = \vartheta(Y)\sigma(Y) = \vartheta(\widetilde{Y})(Q\sigma(Y)) \tag{2.5}$$

which is an exact nonnegative matrix factorization of Y. It is important to note in this setting that the integer p is the number of vertices of the minimal convex hull $\operatorname{conv}(\vartheta(Y))$. We only know that $p \le n$, but it might be that $p \ge m$. See Figure 2.1. In practice, we prefer to see that $\vartheta(Y)$ is contained in convex hull with fewer than $\min\{m, n\}$ vertices, but clearly that is not always possible.

Conversely, suppose a given Y can be factorized as the product of two nonnegative matrices, $Y = UV$, with $U \in \mathbb{R}^{m\times p}$ and $V \in \mathbb{R}^{p\times m}$. We can write

$$Y = \vartheta(Y)\sigma(Y) = UV = \vartheta(U)\vartheta(\sigma(U)V)\sigma(\sigma(U)V). \tag{2.6}$$

Note that the product $\vartheta(U)\vartheta(\sigma(U)V)$ itself is on the simplex $\mathcal{D}_m$. It follows that

$$\vartheta(Y) = \vartheta(U)\vartheta(\sigma(U)V), \tag{2.7}$$

$$\sigma(Y) = \sigma(\sigma(U)V). \tag{2.8}$$

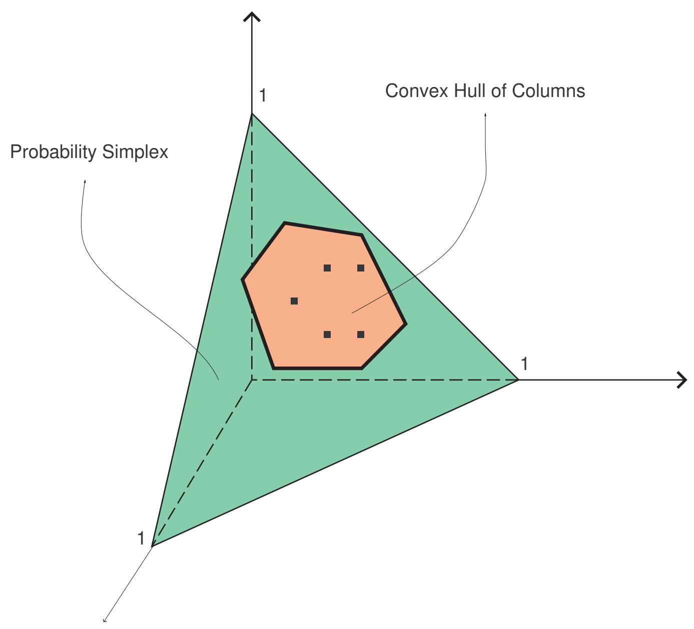


FIG. 2.1. *Convex hull of $\vartheta(Y) \in \mathbb{R}^{3\times n}$ with $n = 11$.*

Because $UV = (UD)(D^{-1}V)$ for any invertible matrix $D \in \mathbb{R}^{p\times p}$, we may assume without loss of generality that U is already a pull-back so that $\sigma(U) = I_n$. It follows that $\vartheta(Y) = \vartheta(U)\vartheta(V)$ and $\sigma(Y) = \sigma(V)$.

The discussion thus far assumes the equality in the factorization of Y. To hold this equality would require either that the integer p is large enough to accommodate all vertices for the convex hull of $\vartheta(Y)$ or that the matrix Y itself is of low rank p. Though these requirements cannot be met in general, the above analysis does suggest that the pull-back of Y to the probability simplex might shed some interesting geometric meaning of NMF which we shall study in the subsequent sections.

Assuming henceforth that the matrix U is already normalized to $\mathcal{D}_m$, that is, assuming $U = \vartheta(U)$ and $\sigma(U) = I_p$, rewrite the objective functional in NMF as follows:

$$f(U,V) = \frac{1}{2}\|Y - UV\|_F^2 = \frac{1}{2}\|\left(\vartheta(Y) - UV\sigma(Y)^{-1}\right)\sigma(Y)\|_F^2. \tag{2.9}$$

For convenience, denote

$$W = V\sigma(Y)^{-1}. \tag{2.10}$$

Then columns in the product $UV\sigma(Y)^{-1} = UW$, each of which is a nonnegative combination of columns of U, can be interpreted as points in the simplicial cone of U. The problem of NMF is equivalent to finding points $\{\mathbf{u}_1, \ldots, \mathbf{u}_p\}$ on the simplex $\mathcal{D}_m$ so that the total distance from $\vartheta(Y)$ to the simplicial cone spanned by $\{\mathbf{u}_1, \ldots, \mathbf{u}_p\}$, measured with respect to a weighted norm, is minimized. Corresponding to each given Y, the scaling factor $\sigma(Y)$ is determined. The distance between Y and UV measured in the Frobenius norm therefore can be considered as the distance between $\vartheta(Y)$ and UW measured in the induced norm from a weighted inner product. The general theory for Hilbert space, in particular the separation of disjoint convex sets by hyperplanes based on the Hahn-Banach theorem, remains applicable. The main thrust of this paper is to explain that by working on the probability simplex $\mathcal{D}_m$ the supporting hyperplane can be calculated effectively.

3. Polytope Approximation on the Probability Simplex. Before we discuss how (2.9) can be minimized in the next section, it might be worthwhile to consider in this section a related set estimation problem — fit $\mathrm{conv}(\vartheta(Y))$ by $\mathrm{conv}(U)$ on the simplex $\mathcal{D}_m$ in the sense that

$$\frac{1}{2}\|\vartheta(Y) - UW\|_F^2 \tag{3.1}$$

is minimized, subject to $U \in \mathcal{D}_m$ and $W \in \mathcal{D}_p$.

It is not difficult to see that this problem is similar in spirit to the well known sphere packing problem [5] as well as the classical problem of approximating convex bodies by polytopes [3, 4, 7, 8]. The underlying principal is also similar to that of the k-plane method [2] or the support vector machines method [1], except that the geometry object used in our approximation, which is either a subspace or a polytope, is of high co-dimension and hence is harder to characterize. For applications, we mention that set estimation is important in pattern analysis [9], robotic vision and tomography, except that most of the applications in pattern recognition are limited to only 3-D objects whereas we are interested in higher dimensional entities.

The optimization problem proposed in (3.1) seems to possess a profound geometric meaning in itself. Since $\mathrm{conv}(U)$ has fewer vertices than $\mathrm{conv}(\vartheta(Y))$, minimizing (3.1) means in a sense retrieving and regulating the "shape" of $\vartheta(Y)$ by fewer facets. If the data $\vartheta(Y)$ have an elongated distribution over $\mathcal{D}_m$ to begin with, the convex hull of at the optimal solution U should display a simpler shape but with similar orientation. Since columns of W represents convex combination coefficients, the diameter of $\mathrm{conv}(U)$ should be at least as large as that of $\mathrm{conv}(\vartheta(Y))$. With that in mind, we notice that the solution U need not be unique because we can always"expand" $\mathrm{conv}(U)$ outward a little bit toward the boundary of $\mathcal{D}_m$ and still obtain the same convex combination UW. The drawing depicted in Figure 3.1 illustrates the idea for the case $m = 3$ and $p = 2$. Note that the segment representing

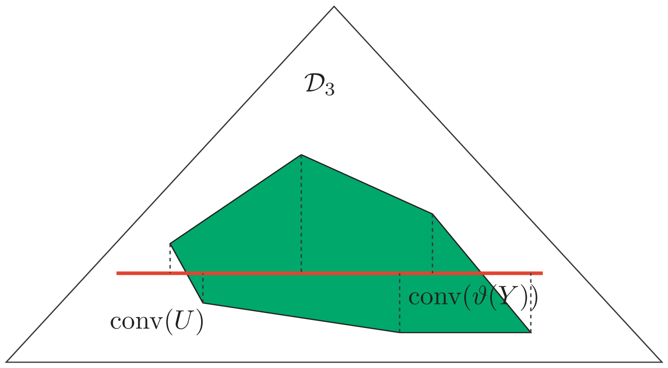


Fig. 3.1. *Convex hull of $\vartheta(Y)$ and U in $\mathcal{D}_3$.*

$\mathrm{conv}(U)$ can be extended to the two sides of the equilateral triangle while still resulting in the same nearest points to $\vartheta(Y)$. We may therefore assume that the columns of U reside on the boundary of the simplex $\mathcal{D}_m$ to begin with. A point on the boundary of $\mathcal{D}_m$ is characterized by the location of zero(s) in its coordinates. For example, each facet has one zero. The ith column in the matrix

$$\frac{1}{m-1}\begin{bmatrix} 0 & 1 & 1 & \dots & & 1 \\ 1 & 0 & 1 & \dots & & 1 \\ 1 & 1 & 0 & & & 1 \\ \vdots & & & \ddots & & \\ 1 & & & & & 1 \\ 1 & 1 & & & 1 & 0 \end{bmatrix}$$

is the "midpoint" of the ith facet of the simplex $\mathcal{D}_m$. When there is more than one zero in the entries, we consider this point to be on a "ridge" and so on.

Putting the about thoughts together, the polytope fitting problem can be cast as a constrained optimization problem,

$$\begin{aligned} \text{minimize} \quad & g(U,W) = \frac{1}{2}\|\vartheta(Y) - UW\|_F^2, \\ \text{subject to} \quad & U \in \partial\mathcal{D}_m, \quad W \succeq 0, \quad \mathbf{1}_p^\top W = \mathbf{1}_n^\top, \end{aligned} \tag{3.2}$$

where $\partial\mathcal{D}_m$ stands for the boundary of $\mathcal{D}_m$. During the computation, it matters to track which boundary is being involved in U.

3.1. Nearest Point in the Convex Hull of U. Let $U \in \mathbb{R}^{m\times p}$ be a fixed matrix with its columns in $\partial\mathcal{D}_m$. The Hahn-Banach theorem asserts that, corresponding to each given column of $\vartheta(Y)$, there is a unique nearest point in $\operatorname{conv}(U)$. Being in $\operatorname{conv}(U)$, the nearest point is a convex combination of columns of U. That is, $W \succeq 0$ automatically. The question is how to find the proximity map for each column of $\vartheta(Y)$.

It is not difficult to derive from linear regression theory that the expression,

$$W := (U^\top U)^{-1}(U^\top \vartheta(Y) - \mathbf{1}_p \boldsymbol{\mu}^\top), \tag{3.3}$$

with

$$\boldsymbol{\mu}^\top = \frac{\mathbf{1}_p^\top (U^\top U)^{-1} U^\top \vartheta(Y) - \mathbf{1}_n^\top}{\mathbf{1}_p^\top (U^\top U)^{-1} \mathbf{1}_p}, \tag{3.4}$$

is a global minimizer of $g(U,W)$ and satisfies $\mathbf{1}_p^\top W = \mathbf{1}_n^\top$. The trouble is that the constraint $W \succeq 0$ often is not satisfied. Instead, since usually p is not too large, we suggest using a recursive algorithm proposed in [24] to find the proximity map.

ALGORITHM 3.1. *Given $Z = \vartheta(Y) \in \mathbb{R}^{m\times n}$ and $U \in \partial\mathcal{D}_m$, the following loop computes the convex combination coefficients W for the proximity map of Z onto* $\operatorname{conv}(U)$.

```
S = [];
C = [];
for i = 1:n
    y = Z(:,i);
    U0 = U - y*ones(1,p);
    active = 1:p;
    [s,c] = sekitani(U0,active);
    S = [S,s+y];
    C = [C,c];
end
W = C;
```

whereas the routine SEKITANI *is evaluated recursively as follow:*

```
function [y,c] = sekitani(U,active);
%
% The code SEKITANI computes the nearest point on the polytope generated by
% the columns of U to the origin
%
```

```
% Reference:
%
%   Sekitani and Yamamoto Algorithm, Math. Programming, 61(1993), 233-249
%
% Input:
%
%   U = vertices of the given polytope
%   active = set of active column indices (of U) for recursive purpose.
%
% Output:
%
%   y = the point on conv(U) with minimal norm
%   c = convex coordinates of y with respect to U, that is, y = Uc
%

[m,p0] = size(U);
p = length(active);
eps = 1.e-13;                                  % relaxed machine zero
looping = 'y';

temp0 = norm(U(:,active(1)));
S = 1;
for j = 2:p
    temp1 = norm(U(:,active(j)));
    if temp1 < temp0, temp0 = temp1; S = j; end
end
x = U(:,active(S));                            % starting point
xc = zeros(p0,1);
xc(active(S)) = 1;                             % initial coordinates of x

while looping == 'y';
    temp0 = x'*U(:,active);
    alpha = min(temp0);
    K0 = find(temp0<=alpha+eps);               % find supporting hyperplanes
    if norm(x)^2 <= alpha+eps
        y = x;
        c = xc;
        looping = 'n';
        return
    else
        Pk = active(K0);
        if length(K0) == 1,
            y = U(:,Pk);
            c = zeros(p0,1);
            c(Pk) = 1;
        else
            [y,c] = sekitani(U,Pk);            % recursion
        end
    end
    I_K0 = setdiff(1:p,K0);
    temp1 = y'*U(:,active(I_K0));
    beta = min(temp1);
    if norm(y)^2 <= beta+eps
        looping = 'n';
        return
    else
        temp3 = U(:,active(I_K0))-y*ones(1,length(I_K0));
        temp5 = find(((y-x)'*temp3)<0);
        temp4 = (x'*temp3(:,temp5))./((y-x)'*temp3(:,temp5));
        lambda = -max(temp4);                   % maximal rotation parameter
        delta = lambda*(y-x);
        if norm(delta) <= max([m,p])*eps
            looping = 'n';
            return
        else
            x = x + lambda*(y-x);
            xc = xc + lambda*(c-xc);            % updated coordinates of x
            looping = 'y';
        end
    end
end
```

Some remarkably nice features of the routine SEKITANI include that it is not based on simplicial decomposition as in the classical method [27] and hence does not require to solve systems of linear equations, that it can start with an arbitrary point in $\mathrm{conv}(U)$, that it does not need an initial separating hyperplane, and that it terminates in finite steps. Details of its mathematical theory can be found in [24]. The method originally proposed in [24] only computes the nearest point $\mathbf{y}$ on the polytope $\mathrm{conv}(U)$ to the origin, but does not provide the corresponding convex combination coefficients of $\mathbf{y}$ with respect to U. As indicated in the above algorithm, the coefficients $\mathbf{c}$ such that $\mathbf{y} = U\mathbf{c}$ with $\sum_{i=1}^{p} c_i = 1$ and $c_i \geq 0$ are attainable with just a few vector operations.

3.2. Improving U. Once the optimal W is found, the next step is to move columns of U along the boundary $\partial\mathcal{D}_m$ to improve the objective value. This amounts to redefining the convex hull $\mathrm{conv}(U)$ to better fit $\mathrm{cov}(\vartheta(Y))$. We suggest to carry out this task by the projected gradient with line search. In this section, we show how the projected gradient and the line search can be calculated efficiently.

We first characterize the projected gradient. It is easy to see that the gradient of g with respect to U at a feasible point (U, W) is given by the $m \times p$ matrix

$$\nabla_U g(U, W) := \frac{\partial g}{\partial U} = -(\vartheta(Y) - UW)W^\top. \tag{3.5}$$

For $i = 1, \dots p$, the ith column of $\nabla_U g(U, W)$ represents precisely the gradient $\nabla_{\mathbf{u}_i} g(U, W)$. In general, this gradient points away from the simplex $\mathcal{D}_m$. We want to project the gradient back to $\partial\mathcal{D}_m$ so that a movement of $\mathbf{u}_i$ in the negative direction of its projected gradient remains in $\partial\mathcal{D}_m$ and decreases the objective value of g. The probability simplex $\mathcal{D}_m$ is made of m facets, identified by the trait that one of the coordinates is zero. It is important to know onto which facets the gradient $\nabla_{\mathbf{u}_i} g(U, W)$ is projected. Typically, we expect that an improvement of $\mathbf{u}_i$ should take place within the same facet where $\mathbf{u}_i$ resides. However, if $\mathbf{u}_i$ happens to be at the intersection of two facets, then a decision should be made on over which facet a better improvement is achieved. Crossing facets is possible. This more complicated task will be addressed later. The projection has to be calculated column by column.

We mentioned earlier that for computation it is important to track which facets of $\mathcal{D}_m$ are involved. Here is the reason. Suppose that $\mathbf{u}_i$ resides on the jth facet of $\mathcal{D}_m$. The jth facet of $\mathcal{D}_m$ can be thought of as the intersection of two hyperplanes $y_1 + \ldots + y_m = 1$ and $y_j = 0$ in $\mathbb{R}^m$. If we define the $m \times 2$ matrix

$$A_j := [\mathbf{1}_m, \mathbf{e}_j], \tag{3.6}$$

where $\mathbf{e}_j$ is the jth standard unit vector in $\mathbb{R}^m$, then the projection of $\nabla_{\mathbf{u}_i} g(U, W)$ onto the jth facet is given by

$$\nabla^j_{\mathbf{u}_i} g(U, W) := (I_m - A_j(A_j^\top A_j)^{-1} A_j^\top)\nabla_{\mathbf{u}_i} g(U, W). \tag{3.7}$$

In fact, the projection matrix $A_j(A_j^\top A_j)^{-1}A_j^\top$ need not be calculated since it is of the form

$$A_j(A_j^\top A_j)^{-1}A_j^\top = \frac{1}{m-1}\begin{bmatrix} 1 & \ldots & 1 & 0 & 1 & \ldots & 1 \\ \vdots & \ddots & & \vdots & & & \\ 1 & & 1 & 0 & 1 & & \\ 0 & & 0 & m-1 & 0 & \ldots & 0 \\ 1 & & 1 & 0 & 1 & & 1 \\ \vdots & & & & & & \vdots \\ 1 & \ldots & 1 & 0 & 1 & \ldots & 1 \end{bmatrix}, \tag{3.8}$$

where the value $m - 1$ occurs at the (j, j) position of the matrix. This formula offers a direct way to compute the projected gradient.

Once we obtain the projection $\nabla^j_{\mathbf{u}_i} g(U, W)$, $i = 1, \ldots, p$, we may use line search techniques to adjust $\mathbf{u}_i$ to reduce the object value. When doing the line search, it becomes critical to adjust the step size so as not to overshoot beyond the facet. That is, we do not allow any entry of the newly computed matrix U to become negative. The optimal step size, which is easy to compute, must be scale back whenever an entry is crossing zero. Zero crossing is an indication that the search path is now reaching a "ridge" and a change of facet is necessary. A preliminary numerical algorithm for updating U is as follows:

ALGORITHM 3.2. *Given $Z = \vartheta(Y) \in \mathbb{R}^{m\times n}$, $U \in \partial\mathcal{D}_m$ and $W \in \mathbb{R}^{p\times n}$, the following steps update U along $\partial\mathcal{D}_m$ while minimizing the objective $g(U, W)$.*

```
function [U,facet] = update(Z,U,W,facet);
%
% Input:
%
%   Z = points to be aprroximated
%   U = vertices of the current polytope
%   W = convex combination coefficients
%   facet = indices of facets where U is residing
%
% Output:
%
%   U = vertices of new polytope
%   facet = indices of new facets where U is residing
%

[m,p] = size(U);
```

```
eps = 1.e-10;                                    % termination tolerance

residue = Z-U*W;
gradient_U = residue*W';                         % negative gradient
if norm(residue,'fro') < eps, return, end

tempA = eye(m)-ones(m)/(m-1);
for i = 1:p
    Proj_matrix = tempA;
    Proj_matrix(facet(i),:) = 0; Proj_matrix(:,facet(i)) = 0;
    temp0 = Proj_matrix*gradient_U(:,i);
    if norm(temp0) > eps*10
        temp0 = temp0/norm(temp0);
    else
        temp0 = temp0*0;
    end
    proj_grad_U(:,i) = temp0;                    % normalized projected gradient
end

temp1 = trace(proj_grad_U*gradient_U');
temp2 = norm(proj_grad_U*W,'fro')^2;
alpha = temp1/temp2;                             % step size for steepest descent

Unew = U + alpha * proj_grad_U;
Unew = Unew.*(abs(Unew)>eps*10);

[row,column] = find(Unew<0);                     % detecting zero crossing
if isempty(column) == 0,
    alpha_modify = alpha;
    for i = 1:length(column)
        temp = -U(row(i),column(i))/proj_grad_U(row(i),column(i));
        if abs(temp) < eps
            alpha_modify = 0;
            vertex = [row(i),column(i)];
        elseif  temp < alpha_modify
            alpha_modify = temp;
            vertex = [row(i),column(i)];
        end
    end
    U = U + alpha_modify*proj_grad_U;

    facet(vertex(2)) = vertex(1);                % facet change
else
    U = Unew;
end

U = U.*(abs(U)>eps);
```

3.3. Alternating Direction Iteration. With the above two mechanisms established, we can use them iteratively between U and W to solve the optimization problem (3.2). That is, with a given $U^{(k)} \in \mathcal{D}_m$, we employ Algorithm 3.1 to compute the projection of $\vartheta(Y)$ onto the polytope $\operatorname{conv}(U^{(k)})$ and obtain the convex combination coefficients $W^{(k)} \in \mathcal{D}_p$. From $(U^{(k)}, W^{(k)})$, we then employ Algorithm 3.2 to find a new set of vertices $U^{(k+1)}$ representing an improved low dimensional polytope approximation to $\operatorname{conv}(\vartheta(Y))$. This is a descent method in both directions and the iteration continues until convergence. Needless to say, the limit point depends on the starting value $U^{(0)}$. Some numerical examples will be given in Section 5.

4. Application to the NMF. The NMF formulated in (2.9) differs from the above polytope approximation in two aspects: that the weight $\sigma(Y)$ will change the inner product used in the polytope approximation and that W is no longer required to be on the simplex $\mathcal{D}_p$. Nonetheless, the idea explored in the proceeding section can easily be generalized. In this section, we discuss its application to the NMF.

We first explain where the conventional method might break down. Despite of its failure, the conventional formulation sheds some insight into the geometric understanding which we shall exploit later. The Lagrangian without the constraint $W \succeq 0$ is given by

$$L(U, W; \boldsymbol{\mu}) := \frac{1}{2}\langle (\vartheta(Y) - UW)\sigma(Y), (\vartheta(Y) - UW)\sigma(Y)\rangle + (\mathbf{1}_m^\top U - \mathbf{1}_p^\top)\boldsymbol{\mu}. \tag{4.1}$$

Then with respect to the Frobenius inner product over the product space $\mathbb{R}^{m\times p} \times \mathbb{R}^{p\times n}$, we find the partial gradients of L are given by

$$\frac{\partial L(U, W; \boldsymbol{\mu})}{\partial U} = -(\vartheta(Y) - UW)\sigma(Y)^2 W^\top + \mathbf{1}_m \boldsymbol{\mu}^\top, \tag{4.2}$$

$$\frac{\partial L(U, W; \boldsymbol{\mu})}{\partial W} = -U^\top(\vartheta(Y) - UW)\sigma(Y)^2, \tag{4.3}$$

respectively. The first order optimality conditions requires that U and W must satisfy the equations

$$U = \left(\vartheta(Y)\sigma(Y)^2 W^\top - \mathbf{1}_m \boldsymbol{\mu}^\top\right)\left(W\sigma(Y)^2 W^\top\right)^{-1}, \tag{4.4}$$

$$W = \left(U^\top U\right)^{-1} U^\top \vartheta(Y), \tag{4.5}$$

where $\vartheta(Y)$ and $\sigma(Y)$ are fixed matrices from a given Y. The Lagrange multiplier $\boldsymbol{\mu}$ can be determined from the constraint $\mathbf{1}_m^\top U = \mathbf{1}_p^\top$ and thus

$$\boldsymbol{\mu}^\top = \frac{1}{m}\left(\mathbf{1}_n^\top \sigma(Y)^2 W^\top - \mathbf{1}_p^\top W\sigma(Y)^2 W^\top\right). \tag{4.6}$$

It is interesting to note that the equation (4.5) is precisely the normal equation for the least squares problem of

$$UW = \vartheta(Y), \tag{4.7}$$

expect that we prefer to see $W \succeq 0$. It might be illuminative to consider the geometry associated with the simplest case when $m = 2$ and $p = 1$. For convenience, denote

$$\sigma(Y) = \operatorname{diag}\{\sigma_1, \ldots, \sigma_n\}.$$

Then the NMF problem is equivalent to finding $\mathbf{u} \in \mathcal{D}_2$ and nonnegative scalars $w_1, \ldots, w_n$ such that

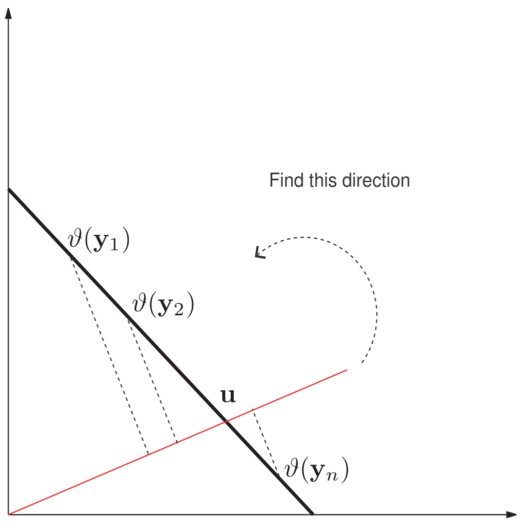
$$f(\mathbf{u}, w_1, \ldots, w_n) := \frac{1}{2}\sum_{i=1}^{n} \sigma_i^2 \|\vartheta(\mathbf{y}_i) - \mathbf{u}w_i\|_2^2, \tag{4.8}$$

is minimized. The relationships (4.4) and (4.5) become

$$\mathbf{u} = \sum_{i=1}^{n} \left(\frac{\sigma_i^2 w_i}{\sum_{i=1}^{n} \sigma_i^2 w_i^2} \vartheta(\mathbf{y}_i) - \frac{\sigma_i^2 w_i - \sigma_i^2 w_i^2}{2\sum_{i=1}^{n} \sigma_i^2 w_i^2} \mathbf{1}_2 \right), \tag{4.9}$$

$$w_i = \frac{\mathbf{u}^\top \vartheta(\mathbf{y}_i)}{\mathbf{u}^\top \mathbf{u}}, \quad i = 1, \ldots, n. \tag{4.10}$$

Note that the vector $\mathbf{u}w_i$ with w_i defined from (4.10) is precisely the projection of $\vartheta(\mathbf{y}_i)$ onto $\mathbf{u}$. In this case, the value of w_i is guaranteed to be positive and is known as soon as $\mathbf{u}$ is given. It remains to find a nonnegative vector $\mathbf{u}$ that satisfies (4.9), (4.10) and $\mathbf{u} \in \mathcal{D}_2$. See Figure 4.1. Unfortunately, the need of $W \succeq 0$ cannot be guaranteed by the projection (4.5) in higher dimensional space. In the following, we explain how the techniques developed for the polytope approximation in the preceding section can be generalized.



Find this direction
$\vartheta(\mathbf{y}_1)$
$\vartheta(\mathbf{y}_2)$
$\mathbf{u}$
$\vartheta(\mathbf{y}_n)$

FIG. 4.1. *NNMF in $\mathbb{R}^2$ when $p = 1$.*

4.1. Nearest Point in the Simplicial Cone of U. With the change of variable (2.10), we shall work with the objective functional

$$h(U, W) = \frac{1}{2}\|(\vartheta(Y) - UW)\sigma(Y)\|_F^2 = \frac{1}{2}\sum_{i=1}^{n} \sigma_i^2 \|\vartheta(\mathbf{y}_i) - U\mathbf{w}_i\|_2^2, \tag{4.11}$$

which is equivalent to $f(U, V)$ in (2.9). Clearly, if each term in (4.11) is minimized, then $h(U, W)$ is necessarily minimized. For each fixed U, instead of (4.5), we want to best approximate each column of $\vartheta(Y)$ within the simplicial cone of U. Since columns of $\vartheta(Y)$ reside on the compact set $\mathcal{D}_m$, their nearest points on the simplicial cone reside in a bounded set as well. In other words, we do not need a very extended region on the cone of U to obtain the nearest points. *We think the portion of the simplicial cone bounded within the unit sphere should be enough.*

With a large enough and fixed positive constant α, define

$$\widetilde{U} = [\mathbf{0}, \alpha\mathbf{u}_1, \ldots, \alpha\mathbf{u}_p]. \tag{4.12}$$

Columns of $\widetilde{U} \in \mathbb{R}^{m\times(p+1)}$ represent $p+1$ vertices of the polytope toward which $\vartheta(\mathbf{y}_i)$, $i = 1,\ldots,n$, is to find its nearest point on the simplicial cone of U. See the geometric illustration depicted in Figure 4.2. By now, Algorithm 3.1 can be applied to find the convex combination coefficients $\widetilde{W} \in \mathbb{R}^{(p+1)\times n}$ for the proximity map of $\vartheta(Y)$ onto $\mathrm{conv}(\widetilde{U})$.

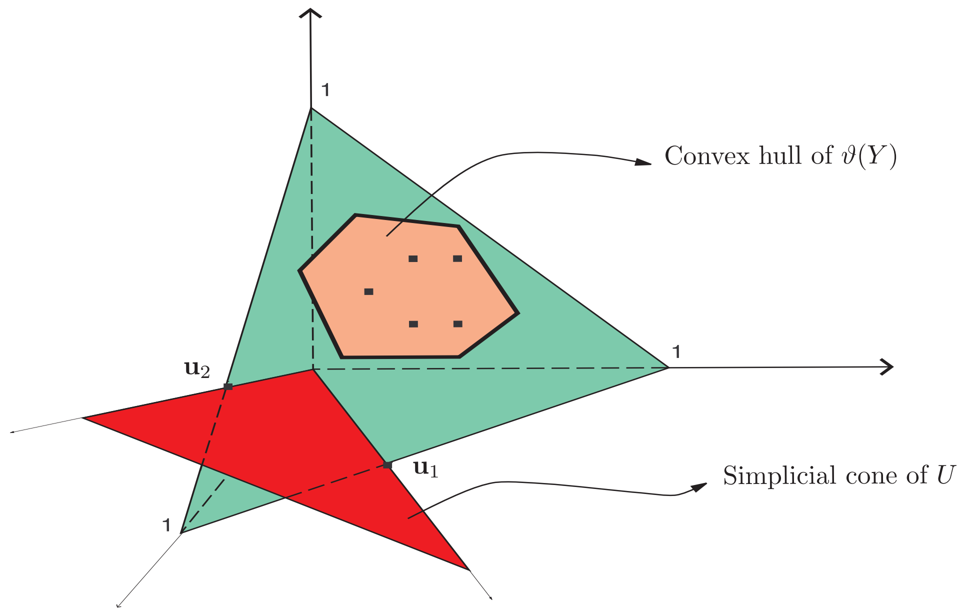


FIG. 4.2. *Simplicial cone of U and convex hull of $\vartheta(Y) \in \mathbb{R}^{3\times n}$ with $p = 2$ and $n = 11$.*

Decompose $\widetilde{W}$ into two blocks,

$$\widetilde{W} = \begin{bmatrix} \mathbf{w}_0^\top \\ W_0 \end{bmatrix}, \tag{4.13}$$

where $\mathbf{w}_0^\top$ denotes the first row of $\widetilde{W}$ and $W_0 \in \mathbb{R}^{p\times n}$. It is clear that the nearest approximation of $\vartheta(Y)$ on the simplicial cone of U is given by

$$\widetilde{U}\widetilde{W} = UW \tag{4.14}$$

with $W = \alpha W_0$. Note that by construction it is guaranteed that $W \succeq 0$, but W is no longer on $\mathcal{D}_p$.

We stress that, given Y and U, our Algorithm 3.1 computes a $p \times n$ nonnegative matrix,

$$V = W\sigma(Y), \tag{4.15}$$

which is a *global minimizer* to the objective functional $f(U,V)$. This is in contrast to the nature of most available NMF algorithms, in particular the well known Lee and Seung's multiplicative update rule [17],

$$V^+ = V.*(U^\top Y)./(U^\top UV), \tag{4.16}$$

where only an approximate minimizer is attained. We stress also that the cost of Algorithm 3.1 is compatible with that of the multiplicative update since the former involves only a few inner products in the process of seeking the supporting hyperplanes.

4.2. Updating U. We describe two ways to update U for the NMF from a given $W \succeq 0$, which could have been found from the above calculation.

The first approach is by means of the steepest descent method. Since the minimum of a function over a larger domain generally is lower than that over a smaller domain, it is reasonable to consider making the "angle" of the simplicial cone as wide as possible. It might be beneficial to expand the simplicial cone of U sideway to until it touches the boundary of $\mathcal{D}_m$. In other ways, we may assume that columns of U are on the boundary $\partial\mathcal{D}_m$ to begin with. See the idea demonstrated in Figure 3.1 and Figure 4.2. With this in mind, we calculate the gradient of $h(U, W)$ with respect to U and obtain

$$\nabla_U h(U, W) := \frac{\partial h}{\partial U} = -(Y - UW)W^\top. \tag{4.17}$$

We may now employ the idea proposed in Section 3.2 to compute the projected gradient of $\nabla_U h(U, W)$ onto proper facets and adjust U along $\partial\mathcal{D}_m$ via the steepest descent method. In short, Algorithm 3.2 is applicable with the modification of setting $Z = Y$ in the input of routine UPDATE.

The second approach is even simpler and more appealing, once Algorithm 3.1 is in place. It is similar in spirit to that of Lee and Seung's algorithm [17] where we can update U in exactly the same way as computing the optimal W by swapping the roles of U and V. More precisely, we write

$$f(U, V) = \frac{1}{2}\| \left(\vartheta(Y^\top) - \vartheta(V^\top)(\underbrace{\sigma(V^\top)U^\top\sigma(Y^\top)^{-1}}_{\Phi}) \right) \sigma(Y^\top)\|_F^2. \tag{4.18}$$

We identify $\vartheta(V^\top)$ as the given points upon whose simplicial cone points in $\mathbb{R}^n$ represented by columns of $\vartheta(Y^\top)$ are to be approximated. We apply the procedures described in Section 4.1 to compute the unique and optimal simplicial combination coefficients $\Phi \in \mathbb{R}^{p\times m}$. Then the optimal U is given by

$$U = \left(\sigma(V^\top)^{-1}\Phi\sigma(Y^\top)\right)^\top. \tag{4.19}$$

Again this procedure is similar to Lee and Seung's multiplicative update,

$$U^+ := U. * (YV^\top)./(UVV^\top), \tag{4.20}$$

except that our U found through (4.19) is a global minimizer for $f(U, V)$ from a fixed V which is calculated in (4.15). For completion, we summarize the second approach in the following algorithm.

ALGORITHM 4.1. *Given a nonnegative matrix $Y \in \mathbb{R}^{m\times n}$, integers $p < \min\{m, n\}$ and maximal allowable number T of iterations, each of the following iterations computes the proximity maps of Y onto the simplcial cones of $U \in \mathbb{R}^{m\times p}$ and $V \in \mathbb{R}^{p\times n}$ alternatively. The resulting nonnegative matrices $U \in \mathcal{D}_m$ and V therefore progress toward minimizing $f(U, V)$.*

```
SigmaY = sum(Y); Z = Y*diag(1./SigmaY);          % pull back of Y
SigmaYt = sum(Y'); Zt =Y'*diag(1./SigmaYt);      % pull back of Y'
alpha = 10;                                      % scaling factor

U = rand(m,p);                                   % initialization of U

for iter = 1:T                           % loop of ADI
    sigmaU = sum(U);
    sigmaU(find(sigmaU~=0))=1./sigmaU(find(sigmaU~=0));
    U = U*diag(sigmaU);                          % pull-back of U
    Utilde = [zeros(m,1),alpha*U];              % simplicial cone of U

    S = []; C = [];
    for i = 1:n
        y = Z(:,i);
        U0 = Utilde - y*ones(1,p+1);
        active = 1:p+1;
        [s,c] = sekitani(U0,active);
        S = [S,s+y];
        C = [C,c];
    end

    W = alpha*C(2:p+1,:);

    V = W*diag(SigmaY);                         % optimal V, given U
    V = V.*(V>eps*10);

    if iter == T
       return                                   % end of iteration
    end

    sigmaVt = sum(V');
    sigmaVt(find(sigmaVt~=0))=1./sigmaVt(find(sigmaVt~=0));
    Vt = V'*diag(sigmaVt);

    Vtilde = [zeros(n,1),alpha*Vt];             % simplicial cone of V

    S = []; C = [];
    for i = 1:m
        y = Zt(:,i);
        V0 = Vtilde - y*ones(1,p+1);
        active = 1:p+1;
        [s,c] = sekitani(V0,active);
        S = [S,s+y];
        C = [C,c];
    end

    U = (alpha*C(2:p+1,:)*diag(SigmaYt))';      % optimal U (w/t scaling), given V
    U = U.*(U>eps*10);
end
```

We emphasize that both matrices W in (4.14) (and hence V) and Φ in (4.18) (and hence U) are readily available from the routine SEKITANI which involves no matrix inversion but only matrix-vector multiplications. Being able to compute the unique global minimizer in each alternating direction efficiently this way is quite noteworthy. We have to comment on, however, two possible drawbacks of our algorithm. First, it is difficult to predict in advance the depth of recursion needed for the calculation. When p is large, the recursion might take several levels deep to complete the process and thus drives up the overhead. Fortunately, in most NMF applications, p is rather small. Since fewer vertices of the polytopes, namely, the truncated simplicial cones of either U or $V^{\top}$, need to be checked for supporting hyperplanes, the recursion can be accomplished reasonably fast. Secondly, the way we compute the proximity map by SEKITANI thus far is to work on one column of Y a time. This is inefficient when m or n is large, which unfortunately is the case for NMF applications. Of course, parallel computation is an alternative since each column can be handled independently. But it is still desirable and perhaps more appropriate to rewrite the SEKITANI algorithm in such a way so that it can compute the proximity maps of multiple columns simultaneously. We are exploring this possibility and will report our investigation somewhere else.

5. Numerical Experiment. In this section, we provide testing results of several numerical experiments to demonstrate the working of our algorithms.

Example 1. In order to demonstrate the polytope approximation on the probability simplex graphically, we limit ourselves to $\mathbb{R}^3$, although the theory works for general m. We randomly generate $n = 18$ points on the simplex. Normally, for low rank polytope approximation of a given set of points, we would have $p < \min\{m, n\}$. With $p = 3$ in this example, the polytope approximation turns into a problem of seeking a "triangle" to enclose the 18 given points [8]. Needless to say, the probability simplex $\mathcal{D}_3$ itself can do the job. Does there exist at least another triangle that circumscribe these points?

Applying Algorithm 3.1 followed by Algorithm 3.2 iteratively 100 times, we see in the left drawing of Figure 5.1 that a "minimal" triangle with vertices on $\partial\mathcal{D}_3$ is found. The triangle is minimal in the sense that some of the given points reside on the sides of the triangle which, thus, cannot be reduced further without leaving some points outside. The dynamics of adjustment by our algorithm on the triangles with vertices on $\partial\mathcal{D}_3$ is illustrated in the right drawing of Figure 5.1.

We caution that, just like most methods of iterative nature, the limiting behavior of our method depends on the initial values. It is possible that a limiting triangle does not include all given points to its interior when the iteration is terminated because the method has reached a local solution.

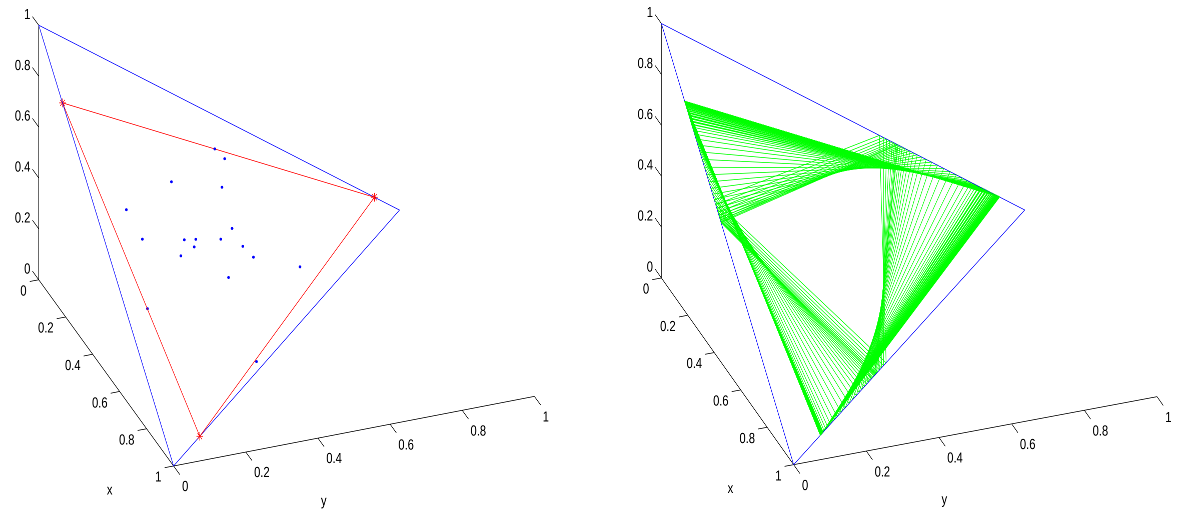


FIG. 5.1. *Triangle (polytope) enclosing a prescribed set of points on $\mathcal{D}_3$.*

Example 2. One of the most attractive features in Algorith 4.1 is that the method finds global minimizer per iteration. To demonstrate this point, we design our experiment as follows. Let $A \in \mathbb{R}^{m\times p}$ and $B \in \mathbb{R}^{p\times n}$ with $p < \min\{m, n\}$ be two randomly generated nonnegative matrices. Define $Y = AB$ which will be our target data matrix. Starting from the same initial value (U_0, V_0), we compare the objective values $f(U,V)$ per iteration with the popular Lee-Seung multiplicative update algorithm.

With a test case where $m = 100$, $n = 50$ and $p = 5$ in 100 iterations, Figure 5.2 demonstrates that our algorithm consistently produces much closer approximation to Y per iteration than the Lee-Seung algorithm. In the case on the left, the objective value is reduced to 0.1909 by our method as opposed to 13.1844 obtained by the Lee-Seung method, although these values continue to be improved with more iterations. In yet another random case on the right, the objective value attained by our method is 3.3035×10^{-4} as opposed to 1.1989 by Lee-Seung. We consistently discover, as are indicated by these two experiments, that our method always produces an impressive improvement by multiple orders.

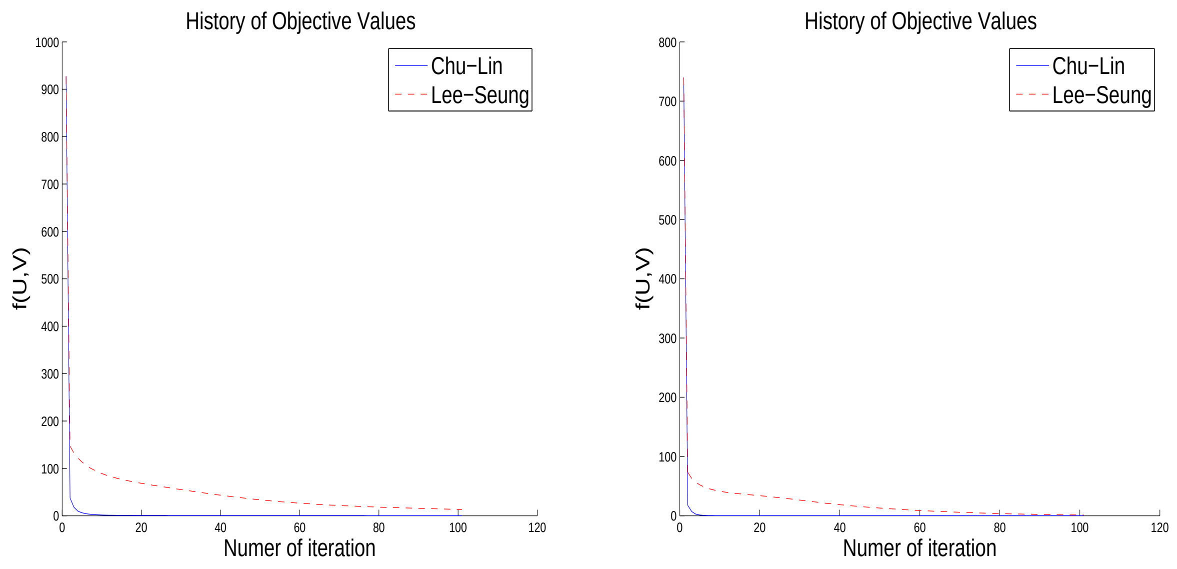


Fig. 5.2. *Objective values $f(U,V)$ per iteration by the Chu-Lin algorithm versus the Lee-Seung algorithm.*

Example 3. To further demonstrate that our method decreases the objective values $f(U,V)$ more rapidly than the Lee-Seung algorithm, we generate randomly a 50×30 matrix Y of rank 20. We systematically seek its nonnegative matrix approximation of rank p, $p = 1, \ldots, 15$, but allowing only 20 iterations. Each point in Figure 5.3 represents the objective value $f(U,V)$ after 20 iterations of the corresponding method. It appears that our method decreases the objective value as p increases while the Lee-Seung method does not seem to be sensitive to p during the first 20 iterations.

The continuous curve in the middle of Figure 5.3 represents the objective values obtained by the Lee-Seung algorithm after 100 iterations. We choose not to draw the objective values obtained by our method after 100 iterations because they are essentially the same as those after 20 iterations with mild differences at the second decimal digit. In other words, our method not only yields a better factorization than the Lee-Seung method, but also seems to converge quickly to an optimal solution in the early iterations.

Example 4. The "swimmer" database characterized in [6] consists of a set of black-and-while stick figures satisfying the so called s*eparable factorial articulation criteria* under which the NMF is known in theory to be able to identify the standard parts in the articulation model (1.3). More specifically, each figure consists of a "torso" of 12 pixels in the center and four "limbs" of six pixels that can be in any one of four positions. With limbs in all possible positions, there are a total of 256 figures of dimension 32×32 pixels. Our data set is provided to us with the courtesy of Alberto Pascual-Motano [22] and Victoria Stodden [6]. A subset of these images are sketched in Figure 5.4.

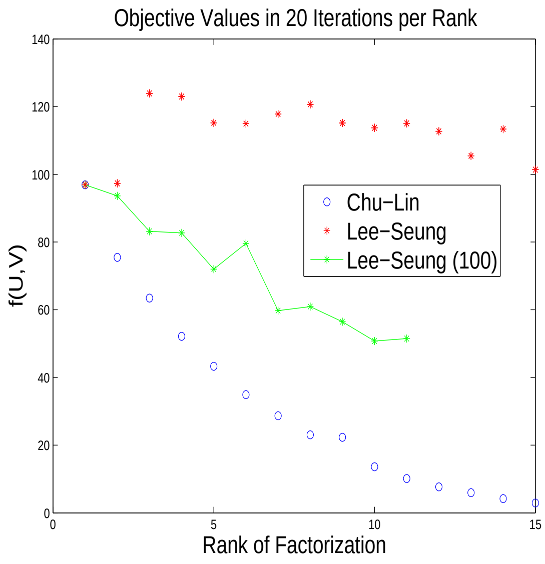


FIG. 5.3. *Objective values $f(U, V)$ in 20 iterations per specified rank by the Chu-Lin and the Lee-Seung algorithms.*

After vectorizing each image into a column, our target matrix Y is of size 1024×256. The numerical rank of Y is 13, but we shall decompose it with $p = 17$ in anticipation of 17 standard parts. We remark that the effect of this overestimation of rank of Y in the NMF is something that deserves further study, but we shall not elaborate in this discussion. Applying our algorithm with only 10 iterations, we obtain the "parts" sketched in Figure 5.5. It is seen that not all limbs are separated from the torso, but all positions are clearly identified. It is worth noting that frame 16 contains entirely zero information, perhaps having something to do with the overestimation of the rank in the NMF.

6. Conclusion. The notion of low dimensional polytope approximation is investigated in this paper. Given a polytope in the first orthant, we stress the benefits of projecting the polytope back to the probability simplex, The pull-back has the advantages that the resulting polytopes are, in as sense, regulated to a more manageable compact set and that proximity map which identifies the unique nearest point on a polytope to an arbitrarily given point can be calculated in finitely many steps.

Two kinds of low dimensional polytopes are used in the discussion. The polytopes of the first kind are restricted entirely to the probability simplex whereas those in the second kind are truncated simplical cones generated from polytopes of the first kind. We apply the ideas to tackle the nonnegative matrix factorization problem. The fact that we are using the proximity maps to compute the unique global minimization in each alternating direction strongly suggests that we should have obtained the best possible approximation per iteration. Numerical experiments seem to evidence much smaller residual errorsin the factorization by our method than by the Lee-Seung multiplicative updating scheme.

In attempting to convey our ideas more easily, the main task of computing the proximity map is accomplished at present column by column and thus is less competitive in speed with the Lee-Seung algorithm which can be executed under matrix-to-matrix operations. A major topic for the next phase of study should be to investigate the possibility of computing the proximity map for multiple columns simultaneously. Such a vectorization, if realizable, would be an added power to our method which in theory should produce the best possible approximation per alternating direction.

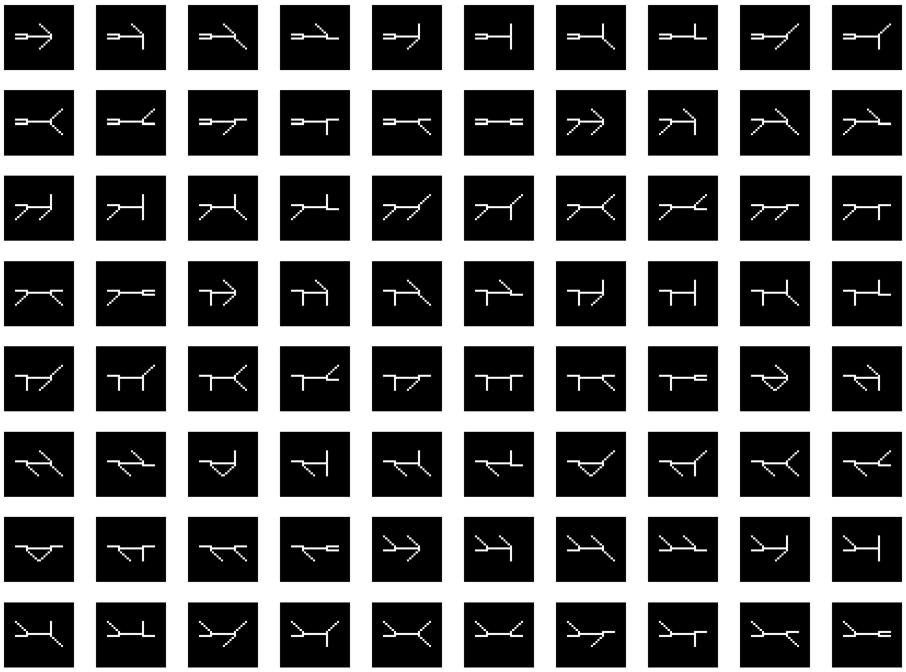


Fig. 5.4. *80 sample images from the swimmer database.*

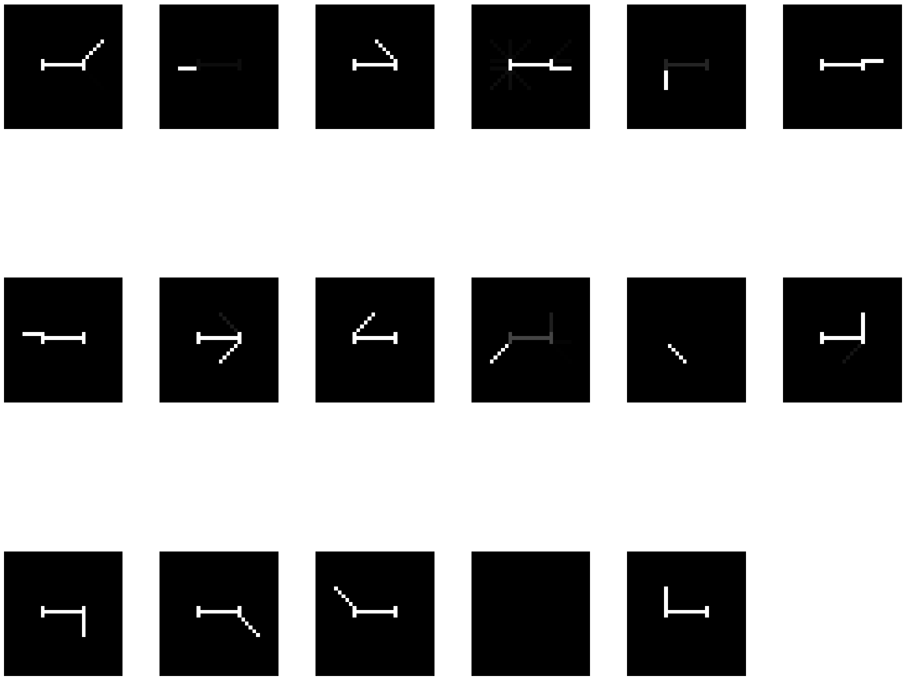


Fig. 5.5. *17 "parts" decomposed from the swimmer database by Algorithm 4.1 in 10 iterations.*

REFERENCES

[1] K. P. Bennett, C. Campbell, Support vector machines: Hype or hallelujah? SIGKDD Explor. Newsl., 2(2000), 1-13.

[2] P. S. Bradley and O. L. Mangasarian, k-Plane clustering, J. Global Optim., 16(2000), 23-32.

[3] L. Chen, New analysis of the sphere covering problems and optimal polytope approximation of convex bodies. J. Approx. Theory, 133(2005), 134–145.

[4] K. L. Clarkson, Algorithm for polytope covering and approximation, in Algorithms and Data Structures, Lecture Notes in Comput. Sci., 709(1993), 246-252.

[5] J. H. Conway and N. J. A. Sloane, Sphere Packings, Lattices, and Groups, 3rd ed., Springer-Verlag, NY, 1999.

[6] D. Donoho and V. Stodden, When does nonnegative matrix factorization give a correct decomposition into parts, Stanford University, 2003, report, available at http://www-stat.stanford.edu/~donoho, also in Proc. 17th Ann. Conf. Neural Infromation Processing Systems, NIPS 2003.

[7] P.M. Gruber, Volume approximation of convex bodies by inscribed polytopes, Math. Ann., 281(1988), 229-245.

[8] P.M. Gruber, Volume approximation of convex bodies by circumscribed polytopes, DIMACS Series 4, AMS, Providence, RI, 1991.

[9] P. Hall and B. A. Turlach, On the estimation of a convex set with corners, IEEE Trans. Pattern Anal. Machine Intell., 21(1999), 225-234.

[10] P. K. Hopke, Receptor Modeling in Environmental Chemistry, Wiley and Sons, New York, 1985.

[11] P. K. Hopke, Receptor Modeling for Air Quality Management, Elsevier, Amsterdam, Hetherlands, 1991.

[12] P. O. Hoyer, Nonnegative sparse coding, Neural Networks for Signal Processing XII, Proc. IEEE Workshop on Neural Networks for Signal Processing, Martigny, 2002.

[13] T. Kawamoto, K. Hotta, T. Mishima, J. Fujiki, M. Tanaka, and T. Kurita. Estimation of single tones from chord sounds using nonnegative matrix factorization, Neural Network World, 3(2000), 429-436.

[14] E. Kim, P. K. Hopke, and E. S. Edgerton, Source identification of Atlanta aerosol by positive matrix factorization, J. Air Waste Manage. Assoc., 53(2003), 731-739.

[15] 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, IEEE Trans. Nerual Networks, 11(2000), 124-136.

[16] D. D. Lee and H. S. Seung, Learning the parts of objects by nonnegative matrix factorization, Nature, 401(1999), 788-791.

[17] D. D. Lee and H. S. Seung, Algorithms for nonnegative matrix factorization, in Advances in Neural Information Processing 13, MIT Press, 2001, 556-562.

[18] W. Liu and J. Yi, Existing and new algorithms for nonnegative matrix factorization, University of Texas at Austin, 2003, report, available at http://www.cs.utexas.edu/users/liuwg/383CProject/final_report.pdf.

[19] E. Lee, C. K. Chun, and P. Paatero, Application of positive matrix factorization in source apportionment of particulate pollutants, Atmos. Environ., 33(1999), 3201-3212.

[20] P. Paatero and U. Tapper, Positive matrix factorization: A non-negative factor model with optimal utilization of error estimates of data values, Environmetrics, vol. 5, pp. 111126, 1994.

[21] P. Paatero and U. Tapper, Least squares formulation of robust nonnegative factor analysis, Chemomet. Intell. Lab. Systems, 37(1997), 23-35.

[22] A. Pascual-Montano, J. M. Carzzo, K. Lochi, D. Lehmann and R. D. Pascual-Marqui, Nonsmooth nonnegative matrix factorization (nsNMF), IEEE Trans. Pattern Anal. Machine Intell., 28(2006), 403-415.

[23] J. Piper, V. P. Pauca, R. J. Plemmons, and M. Giffin, Object characterization from spectral data using nonnegative factorization and information theory. In Proc. Amos Technical Conf., Maui, HI, September 2004, see http://www.wfu.edu/~plemmons.

[24] K. Sekitani and Y. Yamamoto, A recursive algorithm for finding the minimum norm point in a polytope and a pair of closest points in two polytopes, Math. Programming, 61(1993), 233-249.

[25] J. Shen and G. W. Israël, A receptor model using a specific nonnegative transformation technique for ambient aerosol, Atmospheric Environment, 23(1989), 2289-2298.

[26] S. Sra and I. S.Dhillon, Nonnegative matrix approximation: algorithms and applications, Technical Report #TR-06-27, University of Texas, Austin, 2006.

[27] P. Wolfe, Finding the nearest point in a polytope, Math. Programming, 11(1976), 128-149.