AN ALTERNATING MINIMIZATION ALGORITHM FOR NON-NEGATIVE MATRIX APPROXIMATION

JOEL A. TROPP

Abstract. Matrix approximation problems with non-negativity constraints arise during the analysis of high-dimensional non-negative data. Applications include dimensionality reduction, feature extraction and statistical factor analysis. This paper discusses a general algorithm for solving these problems with respect to a large class of error measures. The primary goal of this article is to develop a rigorous convergence theory for the algorithm. It also touches on some experimental results of S. Sra [1] which indicate the superiority of this method to some previously proposed techniques.

Date: 8 May 2003.

J. Tropp is with The University of Texas, Austin, TX 78712 USA, `jtropp@ices.utexas.edu`.

1. Introduction

Suppose that A is a $d \times N$ matrix whose columns are vectors of non-negative data. The problem of extracting r linear features from A can be stated as solving the approximation problem $A \approx VH$, where V is a $d \times r$ matrix of feature vectors and H is an $r \times N$ matrix of coefficients. We quantify the approximation error using a measure that reflects the provenance of the data. For example, if we are working with probability distributions, it makes sense to use the variational distance or the Kullback-Leibler divergence.

A common way to find features is to compute a partial singular value decomposition (SVD) of the data matrix [2]. Although partial SVD provides the best approximation to A in the sense of total least squares, it yields feature vectors that have both positive and negative components. Such features have no interpretation if the data are non-negative. If the data consist of pixel intensities from images, what does a negative intensity signify? If the data consist of word counts from documents, what does a negative word count mean?

To address this metaphysical complaint, several groups of researchers have proposed adding non-negativity constraints to the approximation problem. Paatero and Tapper advocated solving $\min_{V,H \geq 0} \|VH - A\|_{\mathrm{F}}$ for statistical factor analysis [3]. In their first paper on the subject [3], they suggest using an alternating least-squares algorithm [3] to solve the problem. Later Paatero proposes instead a modified gradient descent [4] and a conjugate gradient method [5]. Independently, Lee and Seung considered the same optimization problem for some applications in machine learning. They originally solved it using an alternating projected gradient technique [6]. More recently, they have provided two optimally scaled gradient descent algorithms which respectively minimize the Frobenius norm and the Kullback-Leibler divergence of the approximation error [7]. None of these algorithms have been compared in the literature, and proper convergence proofs are scarce.

We shall call the problem of approximating $A \approx VH$ subject to the constraints $V, H \geq 0$ *non-negative matrix approximation* (NNMA). The present work discusses an alternating minimization algorithm for solving NNMA problems where the approximation error is measured using a very general type of function. The primary goal is to provide a rigorous account of the convergence of this algorithm. I shall also touch on the work of my colleague Suvrit Sra, who has performed some preliminary experiments to compare the actual behavior of this algorithm with some of the other proposed methods [1].

2. Mathematical Background

2.1. **Point-to-Set Maps.** To understand the convergence of the algorithm, we need some elementary concepts from the theory of point-to-set maps. The *power set* $\mathscr{P}(\mathscr{Y})$ of a set $\mathscr{Y}$ is the collection of all subsets of $\mathscr{Y}$. A *point-to-set map* Ω from the set $\mathscr{X}$ to the set $\mathscr{Y}$ is a function $\Omega : \mathscr{X} \longrightarrow \mathscr{P}(\mathscr{Y})$. The composition of two point-to-set maps $\Omega_1 : \mathscr{X} \longrightarrow \mathscr{P}(\mathscr{Y})$ and $\Omega_2 : \mathscr{Y} \longrightarrow \mathscr{P}(\mathscr{Z})$ is defined by $(\Omega_2 \circ \Omega_1)(x) = \cup_{y \in \Omega_1(x)} \Omega_2(y)$ [8].

Suppose that $\mathscr{X}$ and $\mathscr{Y}$ are endowed with topologies so that we may speak of convergence. A point-to-set map Ω is *closed* at $\widehat{x} \in \mathscr{X}$ if $\{x_k\} \subset \mathscr{X}$, $x_k \longrightarrow \widehat{x}$, $y_k \in \Omega(x_k)$ and $y_k \longrightarrow \widehat{y}$ together imply that $\widehat{y} \in \Omega(\widehat{x})$. In words, every convergent sequence whose elements lie in the sets $\{\Omega(x_k)\}$ must have its limit in the set $\Omega(\widehat{x})$. A map is closed on $\mathscr{X}$ if it is closed at each point of $\mathscr{X}$. The composition of two closed maps is always closed [8].

We also define two different types of stationary points. A *fixed point* of the map $\Omega : \mathscr{X} \longrightarrow \mathscr{P}(\mathscr{X})$ is a point x for which $\{x\} = \Omega(x)$. Meanwhile, a *generalized fixed point* of Ω is a point x for which $x \in \Omega(x)$ [9].

2.2. **Iterative Algorithms.** Many procedures in mathematical programming can be described using the language of point-to-set maps. An *algorithm* is a point-to-set map $\Omega : \mathscr{X} \longrightarrow \mathscr{P}(\mathscr{X})$. Given an initial point x_0, an algorithm generates a sequence of points via the rule $x_{k+1} \in \Omega(x_k)$. Now, suppose that $J : \mathscr{X} \longrightarrow \mathbb{R}_+$ is a continuous, non-negative function. An algorithm Ω is *monotonic* with respect to J whenever $y \in \Omega(x)$ implies that $J(y) \le J(x)$. If, in addition, $y \in \Omega(x)$ and $J(y) = J(x)$ imply that $y = x$, then we say that the algorithm is *strictly monotonic.*

Theorem 2.1 (Zangwill [10])**.** *Let Ω be an algorithm that is monotonic with respect to J. Given an initial point x_0, suppose that the algorithm generates a sequence $\{x_k\}$ that lies in a compact set. Then the sequence has at least one accumulation point $\widehat{x}$, and $J(\widehat{x}) = \lim J(x_k)$. Moreover, if Ω is closed at $\widehat{x}$ then $\widehat{x}$ is a generalized fixed point of the algorithm.*

Theorem 2.2 (Meyer [9])**.** *Assume that the algorithm Ω is strictly monotonic with respect to J and that it generates a sequence $\{x_k\}$ which lies in a compact set. If Ω is closed at an accumulation point $\widehat{x}$ of $\{x_k\}$ then $\widehat{x}$ is a (strong) fixed point of Ω. Moreover, if $\mathscr{X}$ is normed, $\|x_{k+1} - x_k\| \longrightarrow 0$. It follows that $\{x_k\}$ converges in norm to $\widehat{x}$ or that the accumulation points of $\{x_k\}$ form a continuum.*

2.3. **Infimal Maps.** Suppose that $f : \mathscr{X} \times \mathscr{Y} \longrightarrow \mathbb{R}_+$. Then, we may define the *infimal map* $M : \mathscr{X} \longrightarrow \mathscr{P}(\mathscr{Y})$ by $M_y(x) = \arg\min_{y \in \mathscr{Y}} f(x, y)$. Define $y \longmapsto M_x(y)$ similarly.

Theorem 2.3 (Dantzig-Folkman-Shapiro [11])**.** *If $f(\widehat{x}, \cdot)$ is continuous on $\mathscr{Y}$, then M is closed at $\widehat{x}$.*

Theorem 2.4 (Fiorot-Huard [12])**.** *If the infimal maps M_x and M_y are both single-valued then the algorithm $\Omega \overset{\text{def}}{=} M_x \circ M_y$ is strictly monotonic with respect to f.*

In Theorem 2.4, the definition of Ω would more properly read $\Omega \overset{\text{def}}{=} (\{x\} \times M_x) \circ (M_y \times \{y\})$.

3. The Algorithm and Its Convergence

3.1. **The Algorithm.** A formal statement of our problem is

$$\min_{\mathsf{V},\mathsf{H} \geq 0} D(\, \mathsf{VH} \,\|\, \mathsf{A} \,),$$

where $D(\,\cdot \,\|\, \cdot\,)$ is a non-negative function for which $D(\, \mathsf{B} \,\|\, \mathsf{A}\,) = 0$ if and only if $\mathsf{B} = \mathsf{A}$. This function need not be symmetric, so we shall call it a *divergence* rather than a metric.

Except for the most trivial divergences, a global optimization in both variables V and H is intractable. We may trace the difficulty to the appearance of the term VH. On its account, the problem is a quadratic program (or worse), which will generally be NP-complete [13].

Therefore, we cannot expect any algorithm to produce a global optimum. For many divergences, however, it is possible to produce the global minimizer with respect to one variable if the other variable is fixed. This observation serves as the basis for a natural algorithm.

Algorithm 1 (Alternating Minimization)**.** *Let $A \in \mathbb{R}_+^{d \times N}$. Given a positive number $r \leq \min\{d, N\}$ and a tolerance $\varepsilon > 0$, the following procedure attempts to compute matrices $V \in \mathbb{R}_+^{d \times r}$ and $H \in \mathbb{R}_+^{r \times N}$ that minimize the divergence $D(VH \parallel A)$.*

(1) Select an initial matrix $H_0 \in \mathbb{R}_+^{r \times N}$.

(2) For $k = 0, 1, 2, \ldots$, solve the following optimization problems:

$$V_{k+1} \in \min_{V \geq 0} \; D(VH_k \parallel A).$$

$$H_{k+1} \in \min_{H \geq 0} \; D(V_{k+1}H \parallel A).$$

(3) Terminate when $\| V_{k+1}H_{k+1} - V_kH_k \|_{\mathrm{F}} < \varepsilon$.

We may expect this algorithm to perform better than a gradient-based algorithm, since it performs a global optimization of one variable at each step.

3.2. **Remarks on Implementation.** In practice, the algorithm should be executed multiple times, beginning with different H_0. From the multiple solutions, we select one with minimal divergence. The starting points may be chosen at random, although an educated guess about H_0 may produce better results.

The implementation of Step (2) depends on the existence of an algorithm for solving the univariate optimizations. If the divergence is convex in its first argument, we can perform each minimization in polynomial time using standard techniques [13]. For common divergences, specialized procedures are faster. For example, one may use the Non-Negative Least-Squares (NNLS) algorithm of [14] to minimize the Frobenius norm of the error.

The time cost of Step (3) is an exorbitant $O(rdN)$. More realistic implementations would not multiply the matrices together to check for convergence. A faster test would just compute the norm between successive V_k and/or successive H_k. But numerical experience suggests that a constant number of iterations is sufficient, so it may not be necessary to test for convergence at all [1].

3.3. **Convergence I.** Under very mild conditions, the iterates produced by Algorithm 1 converge in a weak sense.

Theorem 3.1. *Assume that the divergence $D(\cdot \parallel \mathsf{A})$ is continuous in its first argument. Every accumulation point of the sequence of iterates $(\mathsf{V}_k, \mathsf{H}_k)$ produced by Algorithm 1 is a generalized fixed point of the algorithm.*

Proof Sketch. At each step k, Algorithm 1 composes the two infimal maps,

$$M_{\mathsf{H}}(\mathsf{V}) = \arg\min_{\mathsf{H} \geq 0} D(\mathsf{VH} \parallel \mathsf{A}) \qquad \text{and}$$

$$M_{\mathsf{V}}(\mathsf{H}) = \arg\min_{\mathsf{V} \geq 0} D(\mathsf{VH} \parallel \mathsf{A}).$$

For fixed V, the product map $\mathsf{H} \longmapsto \mathsf{VH}$ is continuous. The divergence is continuous, so the composition of the product map and the divergence is continuous. Theorem 2.3 shows that the infimal map M_{H} is closed. Similarly, M_{V} is closed. Therefore, the composition of the two infimal maps is closed. Since Algorithm 1 minimizes the divergence at each step, it is monotonic with respect to the divergence. The stated conclusions follow from Theorem 2.1. □

3.4. **Convergence II.** If we place additional hypotheses on the behavior of the algorithm, we obtain somewhat stronger conclusions.

Theorem 3.2. *Assume that the divergence $D(\cdot \parallel \mathsf{A})$ is strictly convex in its first argument, and suppose that the r-th singular values of V_k and H_k are bounded away from zero. Then the sequence of iterates $(\mathsf{V}_k, \mathsf{H}_k)$ produced by Algorithm 1 either converges or has a continuum of accumulation points. Every accumulation point is a (strong) fixed point of the algorithm.*

Proof Sketch. Since the r-th singular value of the iterates is bounded away from zero, both components of an accumulation point $(\widehat{\mathsf{V}}, \widehat{\mathsf{H}})$ of the sequence $\{(\mathsf{V}_k, \mathsf{H}_k)\}$ have full rank. The set of full-rank matrices is open, so there exists a neighborhood $\mathscr{U}$ of $\widehat{V}$ which contains only full-rank matrices. For $V \in \mathscr{U}$, the function $H \longmapsto D(\, VH \parallel A\,)$ is strictly convex on the convex set $\{H : H \geq 0\}$, so it has a unique minimizer. That is, the infimal map M_H is single-valued near $\widehat{V}$. Similarly, the infimal map M_V is single-valued near $\widehat{H}$. It follows from Theorem 2.4 that the algorithm is strictly monotonic near $(\widehat{V}, \widehat{H})$. An application of Theorem 2.2 completes the proof. $\square$

3.5. **The Rank Assumption.** Unfortunately, the hypothesis on the rank of V_j and H_j in Theorem 3.2 may not theoretically justified. Consider the matrices

$$A = \begin{bmatrix} 81 & 0 & 0 \\ 0 & 81 & 0 \\ 0 & 0 & 81 \end{bmatrix}, \qquad V = \begin{bmatrix} 4 & 4 & 3 \\ 4 & 3 & 4 \\ 2 & 4 & 4 \end{bmatrix} \qquad \text{and} \qquad H = \begin{bmatrix} 9 & 9 & 0 \\ 0 & 0 & 4 \\ 0 & 0 & 4 \end{bmatrix}.$$

Even though A and V have full rank, the rank-deficient matrix H is the unique solution to $\min_{H \geq 0} \|A - VH\|_{\mathrm{F}}$. Nevertheless, the iterates have always retained full rank in practice [1].

3.6. **Rate of Convergence.** We can always rewrite $VH = (VT)(T^{-1}H)$ for any invertible matrix T. Observe that any positive, diagonal T has a positive, diagonal inverse. Therefore, any fixed point $(\widehat{V}, \widehat{H})$ is part of a continuum of fixed points. In this situation, an alternating minimization typically has a sublinear rate of convergence [15]. After using the alternating minimization to approach a fixed point, it is better in practice to switch to a linearly or quadratically convergent algorithm, such as gradient descent or Newton's Method.

4. Experimental Results

Suvrit Sra has performed some limited experiments with Algorithm 1 [1]. He compiled 143 images to form a matrix with dimensions 9216×143. He produced Frobenius-norm approximations for several rank r using three different methods: Algorithm 1, Lee and Seung's method [7] and partial SVD. Beginning from identical random starting points, he ran the alternating minimization for five iterations, and he ran the Lee-Seung algorithm for 30 iterations. The time costs for the two procedures were comparable. Since partial SVD

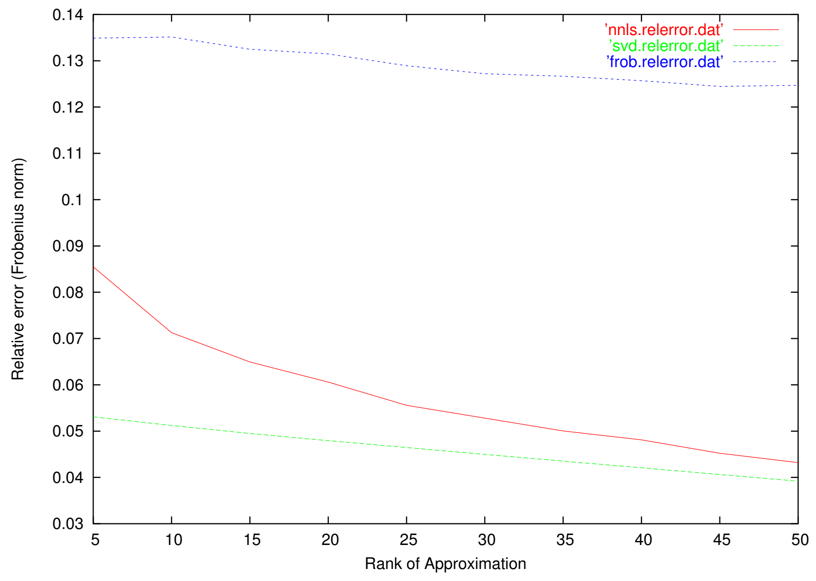

FIGURE 1. Comparison of relative approximation error in Frobenius norm. The top line represents the Lee-Seung algorithm; the middle line represents Algorithm 1; the bottom line represents partial SVD [Sra03].

provides an optimal approximation, it serves as a lower bound on the potential accuracy. Of course, SVD does not respect the non-negativity of the data. The results appear in Figure 1. The relative error in the results from Algorithm 1 is usually less than half the relative error given by the Lee-Seung algorithm, and it even approaches the lowest attainable approximation error. The Lee-Seung error barely drops as the rank increases.

5. Conclusions

I have presented an algorithm for solving non-negative matrix approximation problems with respect to very general divergences, and I have provided a rigorous account of its convergence. Experiments performed by Sra [1] indicate that this method provides better approximations than the Lee-Seung algorithm [7] for Frobenius norm error. Another advantage of the alternating algorithm is that it applies to matrix approximation problems with other types of constraints. For example, Dhillon has argued that certain clustering problems also yield to this approach [1]. This is a promising area for future research.

References

[1] I. S. Dhillon and S. Sra, "Non-negative matrix approximation," 2003, private communication.

[2] I. T. Jolliffe, *Principal Component Analysis*, 2nd ed., ser. Springer series in statistics. New York: Springer-Verlag, 2002.

[3] 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. 111–126, 1994.

[4] P. Paatero, "Least-squares formulation of robust non-negative factor analysis," *Chemometrics and Intell. Lab. Sys.*, vol. 37, pp. 23–35, 1997.

[5] ——, "The Multilinear Engine—a table-driven least squares program for solving multilinear problems, including the n-way parallel factor analysis model," *J. Comput. and Graph. Stat.*, vol. 8, no. 4, pp. 854–888, 1999.

[6] D. D. Lee and H. S. Seung, "Unsupervised learning by convex and conic coding," in *Advances in Neural Information Processing Systems*, M. C. Mozer, M. I. Jordan, and T. Petsche, Eds., vol. 9. The MIT Press, 1997, pp. 515–521. [Online]. Available: citeseer.nj.nec.com/lee97unsupervised.html

[7] ——, "Algorithms for Non-Negative Matrix Factorization," in *Advances in Neural Information Processing*, 2000.

[8] W. W. Hogan, "Point-to-set maps in mathematical programming," *SIAM Rev.*, vol. 15, no. 3, pp. 591–603, July 1973.

[9] R. R. Meyer, "Sufficient conditions for the convergence of monotonic mathematical programming algorithms," *J. Comp. Sys. Sci.*, vol. 12, pp. 108–121, 1976.

[10] W. I. Zangwill, *Nonlinear Programming: A Unified Approach.* Englewood Cliffs: Prentice-Hall, 1969.

[11] G. B. Dantzig, J. Folkman, and N. Shapiro, "On the continuity of the minimum set of continuous functions," *J. Math. Anal. Appl.*, vol. 17, pp. 519–548, 1967.

[12] J. C. Fiorot and P. Huard, "Composition and union of general algorithms of optimization," in *Point-to-Set Maps and Mathematical Programming*, ser. Mathematical Programming Studies. North Holland, 1979, vol. 10, pp. 69–85.

[13] U. Faigle, W. Kern, and G. Still, *Algorithmic Principles of Mathematical Programming.* Kluwer, 2002.

[14] C. L. Lawson and R. J. Hanson, *Solving Least Squares Problems.* SIAM, 1974.

[15] J. de Leeuw, *Information Systems and Data Analysis.* Springer, 1994, ch. Block-Relaxation Algorithms in Statistics.