An Improved Projected Gradient Method for Nonnegative Matrix Factorization

Stephen Ingram*

Abstract

Nonnegative Matrix Factorization is an unsupervised learning method for uncovering latent features in high-dimensional data. This report describes a modification of Lin's Projected Gradient Method for NMF which employs a Newton direction in the line search until any constraints become active. Empirical evidence shows that this technique converges faster than existing methods for NMF.

1 Introduction

High-dimensional data frequently contain fewer true dimensions than measured dimensions. For example, a facial database may contain hundreds of pixels per sample, but only exhibit a few degrees of freedom like expression and orientation. The task of finding the latent features, clusters, or dimensions "hidden" in high-dimensional data is variously called dimensionality reduction, factor analysis, or unsupervised learning. This report investigates a particular method called Nonnegative Matrix Factorization or NMF that is aimed at this task.

NMF is deceptively simple. Given some nonnegative $m \times n$ matrix V we compute a nonnegative $m \times r$ matrix W and $r \times n$ matrix H where $r \ll m, n$ and $V \approx WH$. The subtlety of the method comes from the interpretation of this approximate factorization and is best explained with an example taken from [7]. Suppose the columns of our matrix V represent pictures of faces where each value in a column vector represents the intensity of a pixel in an image. Next, we factorize this matrix into a nonnegative W and H. The resulting columns of W represent the common features across the faces like cheeks or eyes and H contains the coefficients used to combine these common features into faces. More succinctly, the columns of W are the latent features which form a basis and H is a projection of our samples onto this basis.

Algorithms for performing NMF attempt to minimize an objective function representing the difference between the original data V and the approximation WH. Though other measures exist, in this report I only consider the following quartic objective function

*sfingram@cs.ubc.ca

$$\frac{1}{2}\|V - WH\|_F^2 \tag{1}$$

This report contributes an improved projected gradient method for performing NMF. Like previous projected gradient methods, it takes advantage of the structure of this objective function, but it achieves a lower objective value in less time.

2 Previous Work

2.1 Parts-Based Learning

Though introduced in [11] as "Positive Matrix Factorization" in 1994, NMF was more popularly described in [7] by Lee and Seung in 1999. They provided an intuitive description of this factorization as "learning the parts of objects." In their interpretation, the r columns of W are different parts of objects from V and the column i of H tells us which parts to combine best to get an approximation of the corresponding object in column i of V.

Unfortunately, NMF doesn't always provide a parts-based deconstruction of data. Donoho [3] and Stodden describe the ideal conditions under which NMF "finds" these parts. Using a series of geometric arguments they prove that a specific set of conditions must hold for NMF to determine the parts of a dataset. One of these conditions is that the dataset must exhibit factorial sampling, that is the data must contain samples with every feature in every allowable combination. If these conditions do not hold, then a given factorization is neither guaranteed to be unique nor correctly determine the parts in a dataset. In practice, however, even a weakly approximate factorization may be good enough.

The paper also hints at the difficulty of determining the r parameter of the factorization (the number of parts or basis vectors). Briefly, NMF is only applicable to data in the positive orthant which guarantees that all data lies in a simplical cone. If the data aren't all strictly positive, then there exists a unique simplical cone containing the data cloud V. NMF seeks to compute the extreme rays W of this (hopefully) unique simplical cone. Determining r is now a question of knowing how many facets f the simplical cone containing V has. Unfortunately no further work has been done to exactly determine this value for an arbitrary dataset. As of this report, one must make an estimate based on a priori knowledge of V or make adjustments to r as a result of an unsatisfactory factorization.

2.2 Applications

Because of its generality, NMF has found use in a wide variety of applications.

NMF outperforms other partition-based text clustering methods like k-means or PCA [12]. It is currently employed by several digital libraries to perform unsupervised clustering of text documents. In this case, the columns of W are

the subjects in the text database and H gives membership information for each document. NMF has also been used in computer graphics to provide factored representations of Bidirectional Reflectance Distribution Functions or BRDFs [6]. These factored BRDFs are used to improve importance sampling in Monte Carlo image synthesis on lighting models that usually require expensive processing. NMF has also been recently applied to molecular pattern discovery [1]. These disparate examples illustrate that the use of NMF is highly application dependent both in interpretation and application. A clear set of heuristics regarding when to use or when not to use NMF has yet to emerge.

Many applications benefit from a *sparse coding* of the factors W. While the interpretation of sparse coding is application specific, it is typically sought to yield a more exclusive clustering of the data. Work has been done to improve the sparsity of W and H in [4]. It is currently unclear how these sparseness constraints effect the previous methods or how to geometrically interpret the results and it remains an open and important problem.

It is important to note that NMF is frequently just a building block in many systems and not a one-step solution. The factorization usually requires some application-specific augmentation to be used in a real-world system. For example, in text clustering systems [12], it is important to maintain the column size of W and normalize the rows of H against this update.

2.3 Methods

This section briefly discusses previous work on how to minimize our objective (1).

2.3.1 Optimality conditions

Chu et al's NMF survey paper [2] formally derives the first order optimality conditions for (1). They prove that the following effectively characterize the KKT complimentarity conditions for the minimization of our objective:

$$W \otimes ((V - WH)H^T) = 0 \in \mathcal{R}^{m \times r} \tag{2}$$

$$H \otimes (U^T(V - WH)) = 0 \in \mathcal{R}^{r \times n} \tag{3}$$

where $\otimes$ denotes the Hadamard product. In this case $-((V - WH)H^T)$ and $-(W^T(V - WH))$ are the Lagrange multipliers. Note that these are two separate conditions for W and H. As we shall see, most practical methods for NMF solve separate quadratic minimization problems in an alternating fashion.

2.3.2 Algorithms

A thorough discussion of numerical methods applied to NMF is outside the scope of this report, but can be found in [2] which surveys many of the existing methods. This section focuses on the most popular method, the Reduced Quadratic Model algorithm in [8].

Lee and Seung's Reduced Quadratic Model is the most common approach to performing NMF because of the simplicity of its implementation and the low computational complexity per iteration. We first break our quartic objective into two quadratic objectives, one for W and one for H. The method involves solving a sequence of simpler quadratic "auxiliary functions" of our quadratic objectives. These lead to the so-called multiplicative update which the method is sometimes named. An important consequence of using a multiplicative update is that none of the values in W or H can initially be zero without resorting to special processing to avoid divide-by-zero errors in computation. Also, this method is empirically shown to have slower convergence rates on many datasets compared to other approaches [10].

3 Alternating Nonnegative Least Squares

This section discusses minimizing our objective by the method of alternating nonnegative least squares.

Nonnegative least squares seeks to minimize $\frac{1}{2}\|Ax-b\|_2^2$ with $x \geq 0$. Unfortunately our objective (1) is a matrix equation. Chu [2] suggests constructing a series of nonnegative least squares problems from the columns and rows of W and H with the following objectives

$$\frac{1}{2}\|v - Wh\|_2^2 \tag{4}$$

and

$$\frac{1}{2}\|v^T - H^T w^T\|_2^2 \tag{5}$$

for each row of W and column of H. Cursory results from [2] indicate these methods are computationally expensive but yield the lowest objective values.

Here I describe an alternative formulation outlined in [10]. Consider the operator $vec(X)$ which concatenates the columns of any matrix X into a single column vector. Next, consider the operator $bldiag(X, y)$ which constructs for any $m \times n$ matrix X a block diagonal matrix $\widetilde{X}$ of size $(ym) \times (yn)$ containing y blocks consisting of the matrix X. Given these operators we can solve for all the columns of H at once by holding W constant and solving the following nonnegative least squares problem:

$$\min_H \frac{1}{2}\|vec(V) - bldiag(W, n)vec(H)\|_2^2 \text{ subject to } H_{kl} \geq 0, \forall k, l.$$

Likewise we can solve for all the rows of W at once by holding H constant and solving this nonnegative least squares problem:

$$\min_W \frac{1}{2}\|bldiag(H^T, m)vec(W^T) - vec(V^T)\|_2^2 \text{ subject to } W_{ij} \geq 0, \forall i, j$$

These two problems are equivalent to

$$\min_H \frac{1}{2}\|V - WH\|_F^2 \text{ subject to } H_{kl} \geq 0, \forall k, l$$

and

$$\min_W \frac{1}{2}\|H^T W^T - V^T\|_F^2 \text{ subject to } W_{ij} \geq 0, \forall i, j.$$

Also, the KKT conditions (2) and (3) match these two problems respectively. This is relatively easy to see. The gradients of our inequality constraints are vectors of respective sizes rn and rm of all ones. To ensure the gradients of our Lagrangian functions are 0, our Lagrange multiplies must equal to the gradient of our objective function. This is exactly the situation in (2) and (3).

3.1 Using BCLS

Given the above formulations it is now possible to feed these models to a non-negative least-squares solver and obtain very low objective values. Empirical results below show that using the Bound Constrained Least Squares solver or BCLS, which uses a projected gradient method, can reduce the number of outer iterations to reach a given objective value by two orders of magnitude over those required by the Reduced Quadratic method. We do this by solving first for W and then H repeatedly until the size of our projected gradient falls beneath a given tolerance.

Unfortunately, the computational complexity per iteration is too large to compete with the multiplicative method. This complexity arises from evaluating the objective function in our line search. The cost per outer iteration of evaluating our objective is $O(tnmr)$ where t is the number of inner iterations. This is typically much greater than the complexity of the multiplicative method which is simply $O(nmr)$ per iteration (there are no inner iterations in this method).

Lin [10] describes a strategy for dramatically reducing the cost of our inner iterations. In a nutshell, he takes the Taylor expansion of our objective functions and rewrites the sufficient descent conditions of our line-search using this expansion. By taking advantage of the structure of the problem the line-search is reduced to $O(tmr^2)$ to minimize H and $O(tnr^2)$ to minimize W. Since r is typically much smaller than n or m, this is a dramatic improvement.

3.2 Initial Newton Step

An important consequence of Lin's formulation in the previous section is that we solely operate with the "compressed" versions of our objectives. That is, we use $\frac{1}{2}\|V - WH\|_F^2$ and $\frac{1}{2}\|H^T W^T - V^T\|_F^2$. The Hessian matrices of these objectives are $W^T W$ and HH^T respectively. Because these are $r \times r$ and positive definite, it seems natural to use them in computing a Newton direction in our line search. Unfortunately we must only compute the Newton step of the *free* variables of the equation. That is, if any of our constraints become active (if any values

of W or H become 0), then the corresponding column of the Hessian must be excluded from a Newton step computation. Unfortunately this cannot be done with our "compressed" objectives, we must use the full nonnegative least squares objectives whose Hessians are $bldiag(W^TW, n)$ and $bldiag(HH^T, m)$. In other words, if any of our constraints become active, we cannot employ a Newton step without losing the benefits of Lin's projected gradient method.
Fortunately, the following two properties hold:

1. All variables in both of our objectives are initially free.
2. Determining whether any constraints are active is $O(mr)$ and $O(nr)$.

The first property holds because most algorithms initialize W and H with nonzero noise. The second property holds because we simply need to check if any of the values of these matrices are equal to 0. Using these properties we can make a simple and effective modification of Lin's projected gradient method. First we test whether any constraints are active in the matrix we are updating. If none are active, we then perform Newton's method by inverting our "compressed" Hessian, multiplying our gradient and performing a projected line search. If any subsequent constraints become active then we fall back on projected gradient descent. Obviously, if r becomes large, it is better to use a Cholesky Factorization rather than simply inverting the matrix.

As we observe in the next section, this modification yields lower objective values in less time.

4 Results

Lin [10] and [2] compare the results of several algorithms with the original multiplicative method in [8] . We primarily focus on comparing the performance of the method described in section 3.2 with Lin's Projected Gradient Method.

4.1 Synthetic Data

As in [10] we construct a synthetic dataset V by sampling randomly from the normal distribution and initialize W and H similarly. We average the results of 10 tests on two different sizes of datasets. The following table summarizes the results for running Alternating Nonnegative Least Squares with Projected Gradient without (alspgrad) and with (alspgradn) an initial Newton step on a synthetic dataset of size 300×1000 with $r = 20$ for different stopping condition tolerances.

	Seconds Elapsed			Iterations			Objective$\times 10^4$		
tolerence	10^{-3}	10^{-4}	10^{-5}	10^{-3}	10^{-4}	10^{-5}	10^{-3}	10^{-4}	10^{-5}
alspgrad	**0.12**	**0.53**	**0.69**	**2**	**5**	**9**	5.59	5.07	4.87
alspgradn	0.38	0.67	0.98	3	7	15	**5.22**	**4.90**	**4.80**

The table illustrates that for a given tolerance, using an initial Newton step results in a lower objective value. On the other hand, these results aren't completely conclusive of better performance. For example, using projected gradient

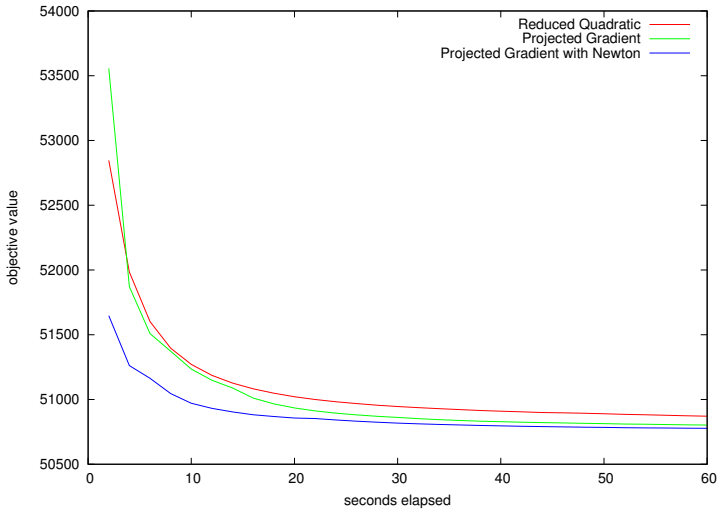

Figure 1: The objective function over time using the three different methods. We omit the initial time-step to preserve lower scales.

without a Newton step appears to result in fewer iterations and lower runtime. It could be that by simply taking a single step or two more yields the lower objectives of the "improved" method. On our second run, we increase the size of the data to 1000×1000 with $r = 50$.

tolerance	Seconds Elapsed			Iterations			Objective$\times 10^5$		
	10^{-3}	10^{-4}	10^{-5}	10^{-3}	10^{-4}	10^{-5}	10^{-3}	10^{-4}	10^{-5}
alspgrad	**0.89**	6.9	19.5	**2**	4	8	1.85	1.69	1.62
alspgradn	1.19	**4.9**	**14.7**	3	**5**	**8**	**1.79**	**1.66**	**1.61**

This table better illustrates the improvement of using the initial Newton step. Except at a low tolerances, alspgradn outperforms alspgrad in runtime, iterations, and objective value.

Due to the difference in iterations, it is still uncertain whether alspgrad performs better than alspgradn in the short term. To better understand the actual behavior of the two algorithms over time, we peformed another experiment. In this experiment we ignore stopping conditions and iterations and only sample the objective function over the elapsed execution time of three different factorization methods. Figure 1 shows the resulting values of our objective (1) achieved by these three different algorithms over time. Here V is 300×1000, W is 300×10, and H is 10×1000. The figure clearly illustrates the benefit of using alspgradn over either Lee and Seung's multiplicative method or just alpsgrad.

4.2 Image Data

We compare alspgradn with the performance on a real facial dataset. The following results exhibit behavior similar to the synthetic results from the previous

section. Here V is 900×64 and $r = 30$.

tolerance	Seconds Elapsed			Iterations			Objective$\times 10^6$		
	10^{-3}	10^{-4}	10^{-5}	10^{-3}	10^{-4}	10^{-5}	10^{-3}	10^{-4}	10^{-5}
alspgrad	15.0	19.7	19.9	1	1	2	**2.61**	2.58	14.3
alspgradn	**7.8**	**8.6**	**14.7**	1	1	2	2.72	**2.56**	**14.0**

The disparate values in runtimes between the synthetic and facial datasets are due to the fact that the facial results are from Java implementations of alspgrad and alspgradn as opposed to the much more efficient Matlab implementations.

5 Conclusions and Future Work

I have provided only preliminary results regarding an initial Newton line search. Before deeming it a true improvement, several important questions should be answered. First, it is unclear exactly when an initial Newton step may yield worse results than projected gradient descent. This obviously isn't the average case, but it appears occasionally in practice. Next, at what point does the cost of computing the Newton direction outweigh its benefits? Since r is small, we assume it is cheap, but when is r too large, and does this depend on the size of m and n? Finally, how do we best initialize our W and H matrices to take advantage of a cheap Newton direction? It is clear that we want to keep our variables free to compute these fruitful search directions, but what isn't clear is exactly how long it is prudent to do this.

The algorithm appears to be best applied to problems where the Reduced Quadratic Model [8] is inadequate. For example, real-time applications like robotics or signal processing will likely find the factorization produced from a single iteration of alspgradn useful. The algorithm is also useful for any application which requires the lowest possible objective value possible. Our experiments indicate that for any given amount of time, alspgradn achieves the lowest objective value for many datasets. Outside of these kinds of applications, the Reduced Quadratic Model will likely be adequate because of its simplicity.

There are many avenues worthy of experimenting with alspgradn. One such avenue is large-sparse systems for which other multiplicative NMF algorithms [12] have been devised. An example of this is in metadata clustering [5] which deals with sparse matrices of size 40000×40000. Another avenue is in applying the algorithm to factorizations similar to NMF, like Concept Factorization [13] which seeks to alleviate the nonnegativity constraints of the factorization. Finally, it would be particularly worthwhile to devise an estimation method for determining r rather than relying on ad-hoc guesswork. Recent intrinsic dimensionality estimators have been developed for projective and manifold methods [9], and it would be interesting to apply their reasoning to NMF.

References

[1] Jean-Philippe Brunet, Pablo Tamayo, Todd R. Golub, and Jill P. Mesirov. Metagenes and molecular pattern discovery using matrix factorization. Proceedings of the National Academy of Science,101(12):4164-4169, 2004.

[2] M. Chu, F. Diele, R. Plemmons, and S. Ragni, Optimality, Computation, and Interpretations of Nonnegative Matrix Factorizations, preprint, 2004, available at http://www.wfu.edu/∼plemmons/papers.htm

[3] 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.

[4] P. O. Hoyer, Non-negative Matrix Factorization with Sparseness Constraints, Journal of Machine Learning Research 5 (2004), 1457-1469.

[5] A. Krowne and M. Halbert, An Evaluation of Clustering and Automatic Classification For Digital Library Browse Ontologies, 2004, report, available at http://www.metacombine.org/reports/research/metacombine_a1_draft_20040704.pdf.

[6] J. Lawrence, S. Rusinkiewicz, and R. Ramamoorthi, Efficient BRDF importance sampling using a factored representation, ACM Transactions on Graphics, Volume 23, Issue 3 (August 2004), 496-505.

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

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

[9] E. Levina and P.J. Bickel, Maximum Likelihood Estimation of Intrinsic Dimension, NIPS 2004.

[10] C.-J. Lin. Projected gradient methods for non-negative matrix factorization. Technical report, Department of Computer Science, National Taiwan University, 2005.

[11] P. Paatero and U. Tapper, Positive matrix factorization: A nonnegative factor model with optimal utilization of error. Environmetrics, 5:111-126, 1994.

[12] W. Xu, X. Liu, and Y. Gong, Document clustering based on non-negative matrix factorization, Proceedings of the 26th annual international ACM SIGIR conference on Research and development in informaion retrieval, 267-273.

[13] W. Xu and Y. Gong, Document clustering by concept factorization, Proceedings of the 27th annual international ACM SIGIR conference on Research and development in information retrieval, 202-209.

APPENDIX A: MATLAB CODE

The following code should replace the nlssubprob function in Lin's alspgrad code. It is modified directly from his code at http://www.csie.ntu.edu.tw/∼cjlin/nmf/.

```
function [H,grad,iter] = nlssubprob_Newton(V,W,Hinit,tol,maxiter)

% H, grad: output solution and gradient
% iter: #iterations used
% V, W: constant matrices
% Hinit: initial solution
% tol: stopping tolerance
% maxiter: limit of iterations

H = Hinit;
WtV = W'*V;
WtW = W'*W;
iWtW = inv(WtW);
Newton = sum( sum( Hinit == 0 ) ) == 0;
alpha = 1; beta = 0.1;
oprojgrad = tol;
for iter=1:maxiter,
  grad = WtW*H - WtV;
  if Newton
     p = iWtW*grad;
  else
     p = grad;
  end

  projgrad = norm(grad(grad < 0 | H >0));
  if projgrad == oprojgrad
    Newton = 0;
    p=grad;
  end
  if projgrad < tol | projgrad == oprojgrad
    break
  end
  oprojgrad = projgrad;

  % search step size
  for inner_iter=1:20,
    Hn = max(H - alpha*p, 0); d = Hn-H;
    gradd=sum(sum(grad.*d)); dQd = sum(sum((WtW*d).*d));
    suff_decr = 0.99*gradd + 0.5*dQd < 0;

    if ~Newton
```

```
      if inner_iter==1,
        decr_alpha = ~suff_decr; Hp = H;
      end
      if decr_alpha,
        if suff_decr,
          H = Hn; break;
        else
          alpha = alpha * beta;
        end
      else
        if ~suff_decr | Hp == Hn,
          H = Hp; break;
        else
          alpha = alpha/beta; Hp = Hn;
        end
      end
    else
      if ~suff_decr
        alpha = alpha * beta;
      else
        Newton = sum( sum( Hn == 0 ) ) > 0;
        alpha = 1;
        H = Hn; break;
      end
    end
  end
end

if iter==maxiter,
  fprintf('Max iter in nlssubprob\n');
end
```

CLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE REGENTS OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.