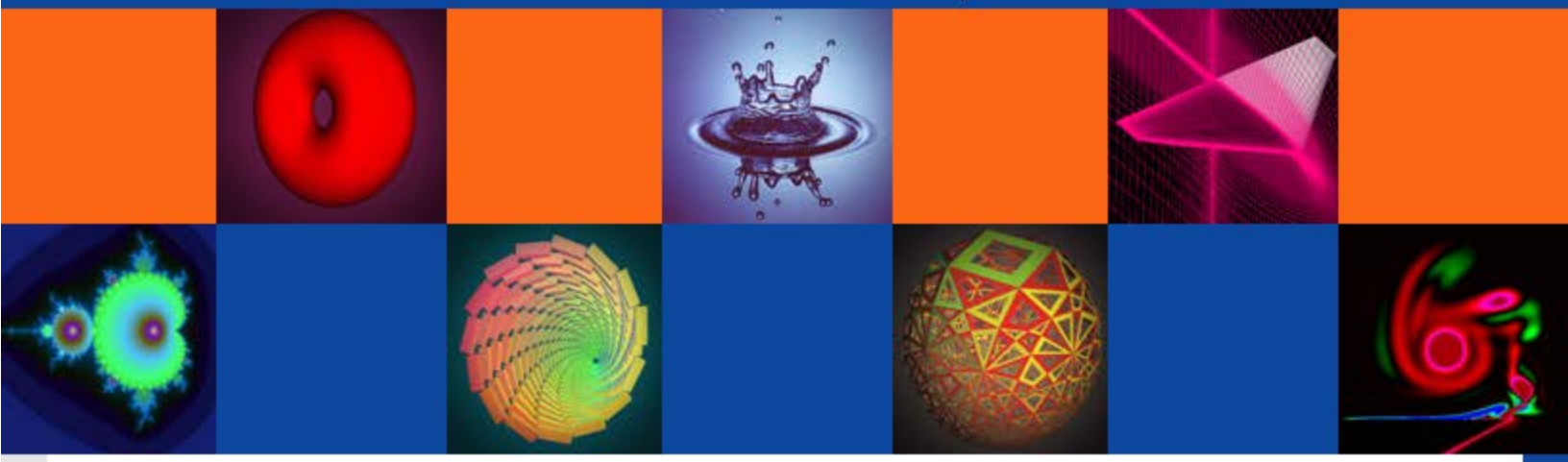
The University of Texas ARLINGTON

MATHEMATICS
Preprint Series



A Multiscale Approach for Nonnegative Matrix Factorization with Applications to Image Classification

Darin Brezeale

Technical Report 2012-01

http://www.uta.edu/math/preprint/

A Multiscale Approach for Nonnegative Matrix Factorization with Applications to Image Classification

Darin Brezeale

February 19, 2012

Abstract

We use a multiscale approach to reduce the time to produce the nonnegative matrix factorization (NMF) of a matrix A, that is, $A \approx WH$. We also investigate QR factorization as a method for initializing W during the iterative process for producing the nonnegative matrix factorization of A. Finally, we use our approach to produce nonnegative matrix factorizations for classifying images and compare it to the standard approach in terms of classification accuracy.

Contents

1 INTRODUCTION

A huge number of images now exist and as the quantity increases, it becomes more difficult for users to find images of interest. Human annotations are effective, but the process of producing these annotations is very time-consuming. Therefore, systems for automatically classifying images have been developed ([SCZ02], [LW07]).

There are numerous methods for classifying data. One general method is to produce a matrix factorization of each class which can then be used to classify new objects. Methods for factorizing a matrix include QR decomposition, singular value decomposition (SVD), and LU decomposition.

A more recently developed factorization is nonnegative matrix factorization (NMF) ([PT94], [LS99]), which approximates a matrix A by a product of matrices, WH. While nonnegative matrix factorization has been shown to be useful in classification ([GSV02], [BB05], [CPF+08]), it suffers from the same problem as other factorizations in that it can take a significant amount of computational effort to produce the factorization.

The standard approach to calculating the nonnegative matrix factorization does so using the original matrix A. Our approach is to reduce the dimensionality of A to produce a smaller version, A_S. We then calculate the nonnegative matrix factorization of A_S to produce $W_S H_S$. H_S is then used as the initial value of H in the iterative process that produces the nonnegative matrix factorization of the original A. Performing part of the process on the smaller matrix A_S reduces the overall time for computing the nonnegative matrix factorization. We refer to this approach as the multiscale approach.

Many of the algorithms for calculating the nonnegative matrix factorization begin by setting the values of W to random values. We look at an alternative approach for initializing W that uses QR factorization.

Since we are interested in using nonnegative matrix factorization for classifying images, our alternative to the standard approach would be of little use if it did not produce approximately the same level of classification accuracy. We investigate this using two different publicly-available data sets of images.

2 NONNEGATIVE MATRIX FACTORIZATION

Given a nonnegative matrix $A \in \mathbb{R}^{m \times n}$, the nonnegative matrix factorization $A \approx WH$ is found by solving

$$\min_{W \geq 0, H \geq 0} ||A - WH||_F$$

where $W \in \mathbb{R}^{m \times k}$, $H \in \mathbb{R}^{k \times n}$, and the terms of each are all nonnegative [Eld07]. The quality of a factorization can be measured using the relative error,

$$\frac{||A - WH||_2}{||A||_2}$$

If each column of matrix A is a sample, then the columns of W form a basis and the columns of H are the combinations of the columns of W for approximating the samples represented in A. For example, the first column of

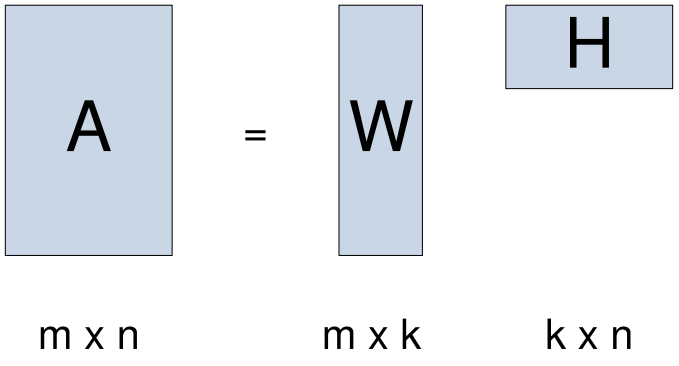


Figure 1: Nonnegative matrix factorization

H is the combination of the columns of W for approximating the first column of A. Because all of the terms of W and H are nonnegative, the nonnegative matrix factorization is additive and probably best-suited for domains in which objects are additive, for example faces [Ski07].

The nonnegative matrix factorization has a number of benefits. In many cases $k \ll \min\{m, n\}$, allowing WH to be a compressed form of A. For example, if A has dimensions of 1000×100, there are 100000 total elements. With a value of $k = 10$, then W has dimensions of $1000 \times 10 = 10000$ and H has dimensions of $10 \times 100 = 1000$. To store W and H we only need to store 11000 elements.

Another benefit of NMF is that the columns of W, which are basis vectors, can be used for classification. An additional benefit is that in some domains negative values don't make sense, so it may not make sense for the matrix factorization to include negative values either.

The NMF is not as well understood as some other matrix factorizations, which leads to several issues. It does not produce a unique factorization since we can produce a nonnegative diagonal matrix D such that $A = WDD^{-1}H$. The solution for W and H may not be a global minimum. Also, convergence can be very slow or not guaranteed at all in some cases [BBL+07].

3 METHODOLOGY

The typical approach to using nonnegative matrix factorization for a set of data is to

1. Create a matrix A in which each column is a sample. If a sample is a matrix, for example an image, produce a column vector from it by stacking the columns of the matrix. That is, starting with the first column of the matrix place the second column of the matrix below it, then below it place the third column, and so forth.

2. Apply nonnegative matrix factorization to A to produce WH.

We use a multiscale approach similar to that described in Li [Li98]. Our hypothesis is that by initially applying nonnegative matrix factorization to a smaller-scale version of our data, we can use the H matrix from this process to bootstrap the generation of the nonnegative matrix factorization for the data at its original dimensionality. Each sample matrix is coarsened by averaging 2×2 grids of matrix values. For example, if we have the matrix

$$\begin{bmatrix} x_{1,1} & x_{1,2} & x_{1,3} & x_{1,4} & \cdots & x_{1,n} \\ x_{2,1} & x_{2,2} & x_{2,3} & x_{2,4} & \cdots & x_{2,n} \\ x_{3,1} & x_{3,2} & \ddots & & & \vdots \\ x_{4,1} & x_{4,2} & & \ddots & & \vdots \\ \vdots & \vdots & & & \ddots & \vdots \\ x_{m,1} & x_{m,2} & \cdots & \cdots & \cdots & x_{m,n} \end{bmatrix}$$

then the set of terms $\{x_{1,1}, x_{1,2}, x_{2,1}, x_{2,2}\}$ would be averaged to produce a single value in the smaller matrix as would $\{x_{1,3}, x_{1,4}, x_{2,3}, x_{2,4}\}$, $\{x_{3,1}, x_{3,2}, x_{4,1}, x_{4,2}\}$, and so forth. The coarsening process can be repeated multiple times. One level of coarsening will reduce a sample matrix to one-fourth of its original size, two levels of coarsening will reduce it to one-sixteenth of its original size, and so forth.

Our approach, shown in Figure 2, is as follows:

1. Coarsen each sample matrix one or more times.
2. A matrix A_S is constructed from these coarsened matrices by converting each of them to a column vector as described above.
3. Nonnegative matrix factorization is applied to A_S to produce $W_S H_S$.
4. Use one of the iterative processes for producing the nonnegative matrix

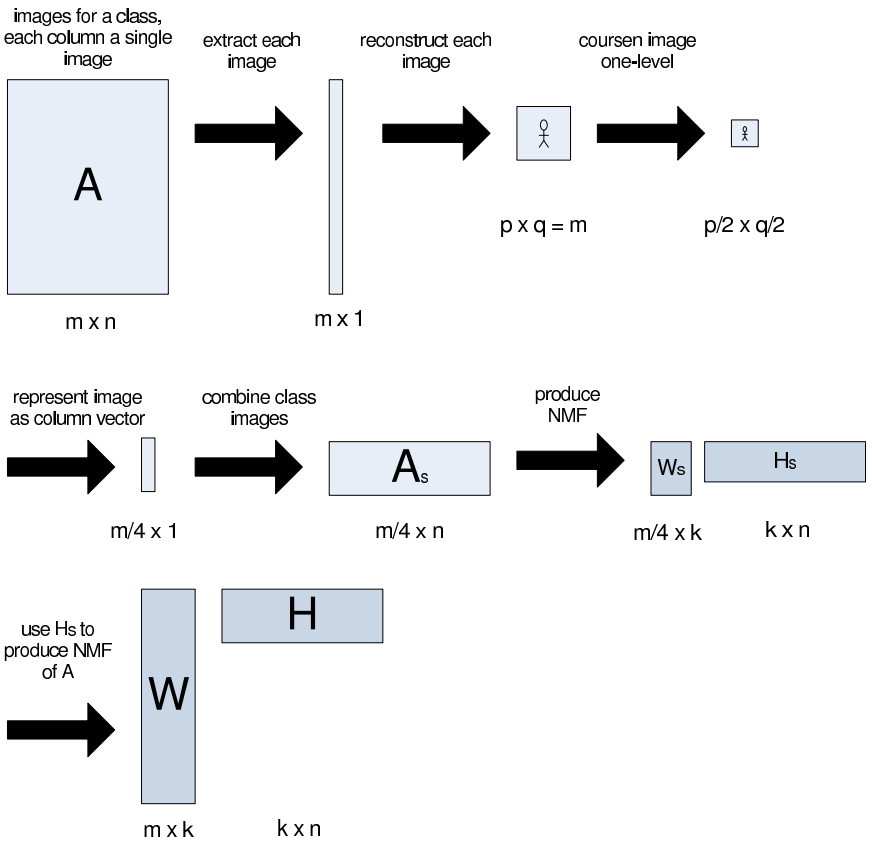


Figure 2: Process of using a multiscale approach with one level of coarsening to reduce the work of producing NMF

factorization of A, but instead of beginning by initializing W, use H_S as an initial value for H and iterate from there.

4 EXPERIMENTS

In this section we describe our data, how it was processed, and the experiments that we conducted. Because of the size and number of tables of results, they have been moved to the appendix.

The experiments were conducted on a computer with an Intel Core 2 Duo

CPU running at 2.33 GHz with 2 GB of memory. The code was implemented in MATLAB ® 7.4.0. We attempted to use as many of MATLAB's built-in functions as possible to reduce the computational inefficiencies associated with programming in interpreted languages. The function to coarsen the matrices was implemented in C to reduce the coarsening time, although this turned out to be a very small part of the overall processing time.

4.1 Data

Our experiments used two publicly available data sets: the Yale Face Database B from Yale University's Center for Computational Vision and Control [GBK01] and the Object and Concept Recognition database from University of Washington's Department of Computer Science and Engineering [Sha10].

The Yale Face Database B data set consists of face images with the light source and face position varying among the images. Each of the 10 classes represents a single individual and has 576 grayscale images with a resolution of 480×640 pixels. To reduce the memory requirements of the images, we chose 384 images from each class and reduced the resolution to 120×160 pixels each.

The Object and Concept Recognition database consists of 21 classes, with each class representing an outdoor concept. The images are mostly in color, but the number of images in each class and the resolution of the images varies. For consistency, we chose 48 images from each of 10 classes and converted them to grayscale images with a resolution of 240×320 pixels each. The 10 classes we chose were `cambridge`, `campusinfall`, `cannonbeach`, `cherries`, `columbiagorge`, `football`, `greenlake`, `sanjuans`, `springflowers`, and `yellowstone`.

To use the images, each image was represented as a column vector, that is, the columns of an image are stacked to form a single column. Therefore, an

image with dimensions $m \times n$ can be represented as a vector with mn elements. All of the images within a class are combined to form a matrix A. Since the images are all grayscale, the values of the matrix elements range from 0 (black) to 255 (white), with the values 1 to 254 representing various shades of gray. For both sets of data, $m \gg n$.

4.2 Calculating NMF

Berry et al. [BBL+07] describe a number of approaches for producing a nonnegative matrix factorization; we chose to use the alternating least squares approach they describe due to its speed and quality of factorization:

Algorithm 1 Alternating Least Squares Algorithm for NMF

```
initialize W
k = 1
while k < max_iterations AND not converged do
    Use least squares to solve for H in W^T W H = W^T A.
    Set all of the negative elements in H to 0.
    Use least squares to solve for W in H H^T W^T = H A^T.
    Set all of the negative elements in W to 0.
    k = k + 1
end while
```

One of the open questions is how to best initialize W since this influences the time to convergence.

There are several ways to produce a least squares solution [TB97]. These are through

1. solving the normal equations
2. QR factorization
3. SVD

Solving the normal equations is what is taught in basic statistics courses, but this approach has stability problems. We settled on the SVD approach. It is

the most stable of the three, but it is also the most computationally expensive.

The singular value decomposition (SVD) of a matrix X produces an $m \times m$ matrix U, an $n \times n$ matrix V, and an $m \times n$ matrix Σ such that

$$X = U\Sigma V^T$$

Σ is diagonal; the diagonal entries are referred to as the singular values of X and are the nonnegative square roots of the eigenvalues of $X^T X$. U and V are both orthogonal matrices. The columns of U are called the left singular vectors and are the eigenvectors of XX^T. The columns of V are called the right singular vectors and are the eigenvectors of $X^T X$ [Dat10].

The Golub-Kahan-Reinsch algorithm is a well-known method for producing the SVD. In this method, X is first transformed into a bidiagonal matrix at a cost of $4mn^2 - 4n^3/3$ flops. This bidiagonal matrix is further transformed to produce Σ. The overall cost for producing U, Σ, and V is $4m^2n + 8mn^2 + 9n^3$ flops. If only the first n columns of U are needed, then the reduced SVD can be produced at an overall cost of $14mn^2 + 8n^3$ flops; Σ will be $n \times n$ in this case [GL96].

The alternating least squares algorithm requires that we solve for W, then use this value to solve for H, which in turn is used to solve for W, and so forth until some stopping criteria have been met. To use the SVD for this, the SVD of W is found. To solve for H, we use

$$H = V\Sigma^{-1}U^T A$$

Then the SVD of H is found, which can be used to solve for W with

$$W^T = V\Sigma^{-1}U^T A^T$$

For our experiments, we chose two different methods for initializing the W matrix. The first was to use random values in the range of 0–255 (the same range as the pixel values in our data) and the second was QR factorization with column pivoting. Many of the nonnegative matrix factorization algorithms suggest using random values, so this gives us a good point for comparison.

All $m \times n$ matrices A can be factored into two matrices, QR, where Q is an $m \times m$ orthogonal matrix and R is an $m \times n$ upper triangular matrix. When A has full column rank, the first n columns of Q form an orthonormal basis for $R(A)$ [GL96]. When A is rank-deficient, a basis for $R(A)$ cannot be found using QR factorization. However, by finding a permutation matrix P such that $AP = QR$, the columns of A can be reordered such that the first r columns of Q will form an orthonormal basis. If the rank of A is $r < n$,

$$R = \begin{bmatrix} R_{11} & R_{12} \\ 0 & 0 \end{bmatrix}$$

where R_{11} is an $r \times r$ upper triangular, nonsingular matrix [Dat10].

4.3 Stop Using Standard Approach

The purpose of this set of experiments, referred to as *stopStandard* in the tables, was to compare the time for both approaches to reach the same level of relative error in the factorizations. This value, *minStop*, was determined by performing the standard approach with the iterative process running until either the change in relative error was less than 0.01 or 25 iterations had been reached. *minStop* was the minimum value of the relative error from this. After determining *minStop*, both the standard and multiscale approaches were performed until their relative errors reached *minStop* or 25 iterations.

When initializing using random values, we used 100 runs per class. When

using QR factorization with column pivoting there was only one run per class since the QR factorization will be the same each time. For the Concepts data, we chose values of k of 5, 10, and 20 while for the Faces data we chose values of k of 10, 20, and 30. Larger values of k were used with the Faces data since each class had more samples that the Concepts data samples, but the specific values of k were somewhat arbitrary. We tested levels of coarsening of 2 and 3. When comparing times, the total multiscale time is the sum of the times to reduce the resolution of the images, produce the NMF of A_S, and produce the NMF of A using H_S.

The distribution of the means was not normal, so one-sided t-tests were not appropriate for significance testing. Instead, we chose the Wilcoxon rank sum test [SS01]. The null hypothesis H_0 was that the mean times are not different; the alternative hypothesis H_A was that the multiscale approach is faster; and the α value was 0.05.

Table 1 shows the results for the Concepts data. We can see that when W was initialized with random values, the multiscale approach took less time than the standard approach. Table 2 shows the average number of iterations for the standard approach and multiscale approaches. For the multiscale approach what is shown is the average number of iterations for calculating the nonnegative matrix factorization of the small matrix A_S as well as the average number of iterations for calculating the nonnegative matrix factorization of the full matrix when the iterative process begins with H_S. From this table, we can see that the mean number of iterations for the standard approach is comparable to the mean number of iterations for the phase of the multiscale approach in which the coarsened matrix A_S is used.

Figure 3 shows the relative error values for a typical run when the multiscale approach is faster than the standard approach. We can see that the results for

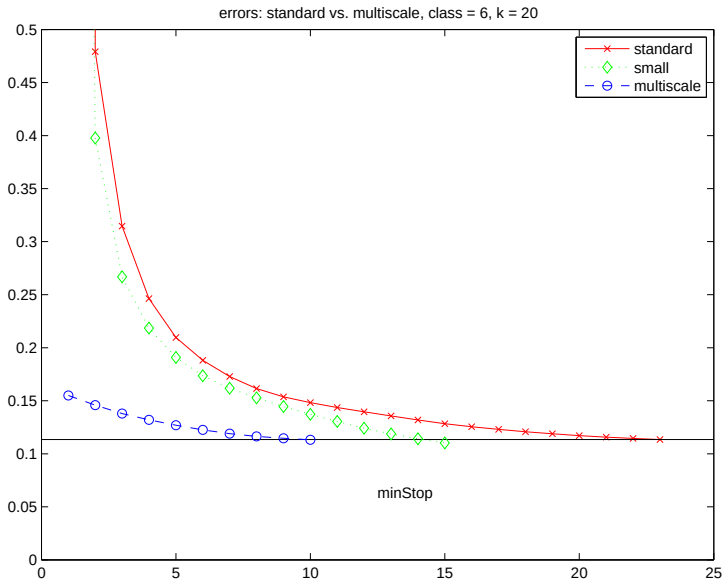


Figure 3: Relative error from a typical run when initializing with random values for $k = 20$.

the first phase of the multiscale approach, which is using A_S, are similar to the standard approach. However, the time is much less since A_S is one-sixteenth or one-sixty-fourth the size of the original A. When the multiscale approach is producing the nonnegative matrix factorization of the full-size matrix A it begins with a lower relative error which helps it reach convergence in much fewer iterations than the standard approach. We found this to be the case for many of the runs. In some cases, it took as little as two iterations to produce the nonnegative matrix factorization of the full-size matrix A when the process began with H_S.

Figure 4 shows the relative error for a typical run when $k = 5$. We can see here why the multiscale approach is not generally faster than the standard approach when $k = 5$. In this case the standard approach often converges in only a few iterations, which means that there is not much room for improvement in the number of iterations required.

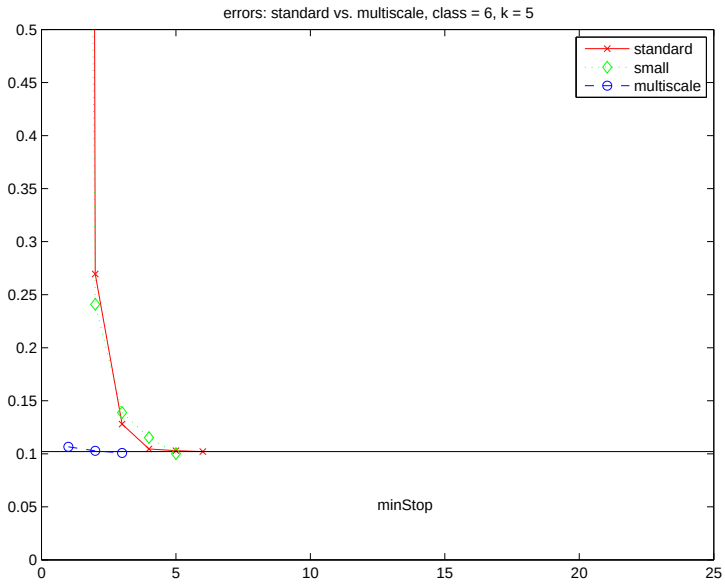


Figure 4: Relative error from a typical run when initializing with random values for $k = 5$.

When W was initialized using QR factorization with column pivoting, our approach was almost always slower. Initializing with QR factorization reduced the overall time for the standard approach while increasing the overall time for the multiscale approach. We can see that when initializing with QR factorization instead of random values, the multiscale approach slowed down because producing the nonnegative matrix factorization of the full-size matrix using H_S took many more iterations, sometimes reaching the maximum of 25. Figure 5 shows the errors from a typical run. We can see that while the relative error for the multiscale approach starts low, it basically remains flat and therefore cannot reach *minStop*.

For the Faces data, the results were mixed when W was initialized with random values. Our approach was always slower when W was initialized using QR factorization with column pivoting. As was the case with the Concepts data, using QR factorization for initializing W resulted in the multiscale approach

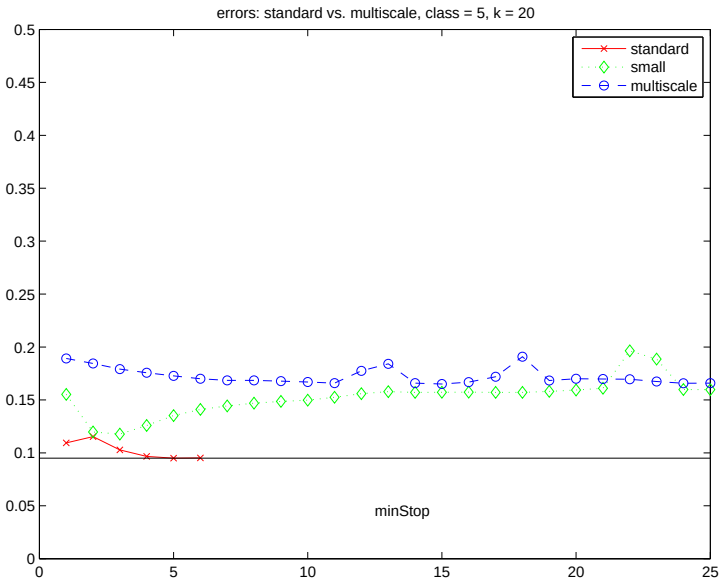


Figure 5: Relative error from a typical run when initializing using QR with column pivoting.

reaching a very large number of iterations while it greatly reduced the number of iterations for the standard approach.

Since we use two different methods for initialization of W, the question arises as to whether either of them is faster than the other. When testing for statistical significance the null hypothesis H_0 is that the mean times are not different while the alternative hypothesis H_A is that QR factorization with column pivoting is faster.

For the Concepts data, initialization with QR factorization with column pivoting is faster than random values for k values of 10 and 20 when using the standard approach. When using the multiscale approach, QR was never faster at a statistically significant level. For the Faces data, using QR factorization with column pivoting was faster for all tested combinations of coarsening and values of k when the standard approach was used, but never for the multiscale approach.

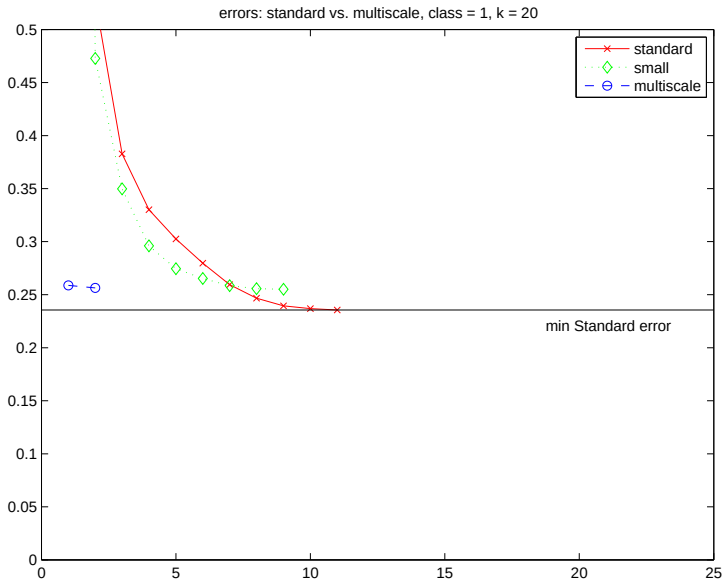


Figure 6: Relative error from a typical run when initializing with random values.

4.4 Stop When Converged

For this set of experiments, referred to as *stopConverged* in the tables, each approach ran until convergence or 25 iterations, independently of the other. Convergence was defined to be when the change in relative error was less than or equal to 0.01. The number of runs, values of k, levels of coarsening, and initialization methods for W were the same as described in Section 4.3. As with the previous experiments, the distribution of the means was not normal, so the Wilcoxon rank sum test was used for significance testing.

Table 7 shows that for the Concepts data, the multiscale approach was faster at all levels of coarsening and values of k when W was initialized with random values. It was also faster for $k = 5$ when W was initialized with QR factorization with column pivoting. Figure 6 shows the results from a typical run when W is initialized with random values.

When applying the multiscale approach to the Concepts data after initial-

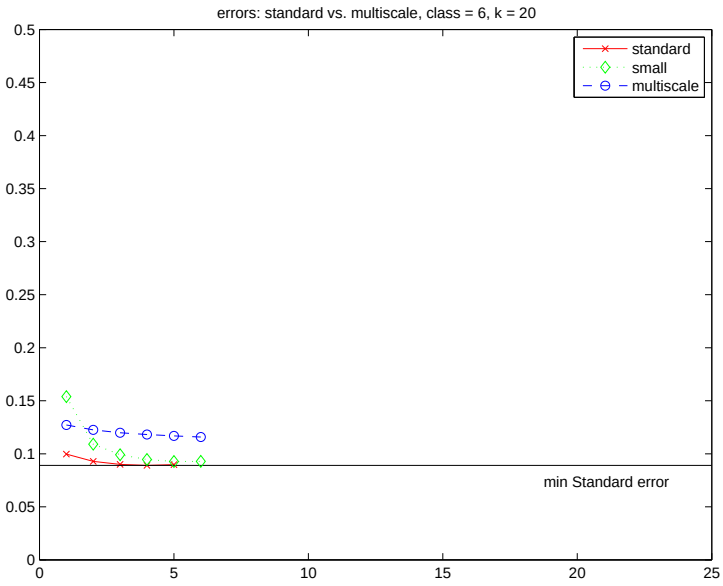


Figure 7: Relative error from a typical run when initializing using QR with column pivoting.

izing W using QR factorization with column pivoting, table 8 shows that the mean number of iterations required to reach convergence was far less than the maximum. This is due to it no longer being necessary to achieve the same level of relative error as produced by the standard approach. However, in general the multiscale approach still did not outperform the standard approach. Figure 7 shows why. For many of the runs, when initializing W using QR factorization with column pivoting the standard approach converged very quickly which made it difficult for the multiscale approach to beat.

Tables 9 and 10 show the results for the Faces data. We can see that when initializing using random values the multiscale approach is always faster, but the multiscale approach is generally not faster when initializing using QR factorization with column pivoting. The standard approach is generally faster when initializing with QR factorization with column pivoting as opposed to random values. The multiscale approach is generally not faster since the mean times are

similar.

4.5 Comparison of Errors

We were also interested in the minimum, maximum, and mean relative errors for various configurations. Tables 13 and 14 show the results from 100 runs per class when W was initialized with random values and one run per class when W was initialized using QR factorization with column pivoting for each combination of configuration options. In each case the algorithms were allowed to iterate 25 times.

We can see that when initializing W with random values, the mean relative errors for the multiscale approach are much lower than those for the standard approach. This is to be expected if the coarsening process has been effective. When initializing W using QR factorization with column pivoting, the mean relative errors for both approaches are similar. For the largest values of k, the standard approach has slightly lower values for mean relative error.

4.6 Classification

In order to use the nonnegative matrix factorization for classification, the training samples for each class i are combined to form a matrix A_i. For each A_i, a factorization is produced. Given a new sample, q, we solve the least squares problem

$$\min_x ||W_i x - q||_2$$

for each class i and classify q as the class in which the error is smallest [Eld07].

When comparing mean classification results using the Standard approach versus the Multiscale approach, the null hypothesis H_0 is that the mean classification accuracy is not different. The distribution of the mean classification results for the Concepts data was approximately normal, so significance test-

Algorithm 2 Classification Process

for i = 1 to number_of_classes **do**
  $A_i \approx W_i H_i$
end for
for i = 1 to number_of_classes **do**
  $[Q_i, R_i] = \text{QR}(W_i)$
end for
for k = 1 to number_of_query_vectors **do**
  for i = 1 to number_of_classes **do**
    $x_i = R_i^{-1}(Q_i^T b_k)$
    $\text{error}_i = ||\text{query\_vector}_k - W_i x_i||_2$
  end for
  classify query_vector_k as class i in which error_i is smallest
end for

ing was performed using one-sided t-tests were performed with an α value of 0.05. For the Faces data, the Wilcoxon rank sum test was used because the distribution of the mean classification results was not normal.

Table 15 shows the classification accuracy for the Concepts data. It is likely that nonnegative matrix factorization did not perform well for this set of data because of the small sample sizes of each class, the large variety within each class, and the large dimensions of each sample. However, our purpose was not to demonstrate the effectiveness of nonnegative matrix factorization but instead was to demonstrate that the multiscale approach did not reduce the classification accuracy of nonnegative matrix factorization. This was the case for the Concepts data.

When using the Faces data the classification results were always very high, with the smallest mean classification results being approximately 87%. In many cases, the standard approach produced better results than the multiscale approach. While they were statistically significant, in most of these cases the differences were not practically different. For example, when initializing using random values with a coarsening level of 2 and a k value of 20, the standard approach had a mean classification accuracy of 99.69% while the multiscale ap-

proach had a mean classification accuracy of 98.19%. However, there were two cases where the standard approach did perform significantly better, both when initializing using random values and the level of coarsening was 3. When k was 20, the standard approach had a mean classification accuracy of 99.19% while the multiscale approach was 93.29%. When k was 30 the standard approach had a mean classification accuracy of 99.20% while the multiscale approach was 86.80%.

5 CONCLUSIONS

Our hypothesis was that we could reduce the overall time to produce the nonnegative matrix factorization of a matrix of images by first finding the nonnegative matrix factorization of a matrix of the coarsened images. We found this to generally be the case when the W matrix was initialized using random values. We also found that when the standard approach was used to produce the nonnegative matrix factorization, initializing W using QR factorization with column pivoting was typically faster than when initializing using random values.

Our method is not always faster than the standard approach, even when initializing W with random values. Future work should look at what values of k and levels of coarsening produce the best results.

This work used two different data sets of images. Images are two-dimensional structures, so coarsening them produces smaller images that are still recognizable. Another area to investigate would be other data types, such as documents, that are not thought of as having two-dimensional structures.

6 ACKNOWLEDGMENT

I would like to thank Dr. Ren-Cang Li of the Department of Mathematics at the University of Texas at Arlington. Dr. Li proposed this as a topic for my Master's project. He also provided feedback during the entire period in which I worked to test the idea and chaired the committee to evaluate the Master's project.

References

[BB05] Michael W. Berry and Murray Browne. Email surveillance using non-negative matrix factorization. *Computational and Mathematical Organzation Theory*, 11(3):249–264, 2005.

[BBL+07] Michael W. Berry, Murray Browne, Amy N. Langville, V. Paul Pauca, and Robert J. Plemmons. Algorithms and applications for approximate nonnegative matrix factorization. *Computational Statistics & Data Analysis*, 52(1):155–173, 2007.

[CPF+08] Wen-Sheng Chen, Binbin Pan, Bin Fang, Ming Li, , and Jianliang Tang. Incremental nonnegative matrix factorization for face recognition. *Mathematical Problems in Engineering*, 2008:17 pages, 2008.

[Dat10] Biswa Nath Datta. *Numerical Linear Algebra and Applications*. Philadelphia, PA, 2^{nd} edition, 2010. Society for Industrial and Applied Mathematics.

[Eld07] Lars Eldén. *Matrix Methods in Data Mining and Pattern Recognition*. SIAM, Philadelphia, PA, 2007. Society for Industrial and Applied Mathematics.

[GBK01] A.S. Georghiades, P.N. Belhumeur, and D.J. Kriegman. From few to many: Illumination cone models for face recognition under variable lighting and pose. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 23(6):643–660, 2001.

[GL96] Gene H. Golub and Charles F. Van Loan. *Matrix Computations*. Johns Hopkins University Press, Baltimore, MA, 3^{rd} edition, 1996.

[GSV02] David Guillamet, Bernt Schiele, and Jordi Vitriá. Analyzing non-negative matrix factorization for image classification. In *16^{th} International Conference on Pattern Recognition*, volume 2, pages 116–119, 2002.

[Li98] Ren-Cang Li. A multi-resolution approach for calculating primary eigenvectors of a large set of images. Technical report, 98-13, Department of Mathematics, University of Kentucky, 1998.

[LS99] Daniel D. Lee and H. Sebastian Seung. Learning the parts of objects by non-negative matrix factorization. *Nature*, 401:788–791, 1999.

[LW07] D. Lu and Q. Weng. A survey of image classification methods and techniques for improving classification performance. *International Journal of Remote Sensing*, 28(5):823–870, 2007.

[PT94] Pentti Paatero and Unto Tapper. Positive matrix factorization: A non-negative factor model with optimal utilization of error estimates of data values. *Environmetrics*, 5(2):111–126, 1994.

[SCZ02] Raimondo Schettini, Gianluigi Ciocca, and Silvia Zuffi. A survey of methods for colour image indexing and retrieval in image databases. In Lindsay W. MacDonald and M. Ronnier Luo, editors, *Colour Image Science: Exploiting Digital Media*. John Wiley, 2002.

[Sha10] Linda G. Shapiro. Project report for object and concept recognition for content-based image retrieval. Technical report, University of Washington, retrieved 2010.

[Ski07] David Skillicorn. *Understanding Complex Datasets: Data Mining with Matrix Decompositions*. Chapman & Hall/CRC, Boca Raton, FL, 2007.

[SS01] P. Sprent and N.C. Smeeton. *Applied Nonparametric Statistical Methods*. Chapman & Hall/CRC, Boca Raton, FL, 3^{rd} edition, 2001.

[TB97] Lloyd N. Trefethen and David Bau, III. *Applied Numerical Linear Algebra*. SIAM, Philadelphia, PA, 1997.

7 Appendix A

Table 1: Processing times for Concepts data, stopStandard experiments

			Standard		Multiscale			
level	k	initialization	mean	std	mean	std	p-value	faster?
2	5	random	7.01	2.16	6.65	7.13	0.000	yes
2	10	random	13.80	4.54	10.18	8.78	0.000	yes
2	20	random	21.60	6.51	17.67	10.95	0.005	yes
3	5	random	7.40	2.03	7.29	7.29	0.000	yes
3	10	random	13.81	4.70	10.48	8.74	0.000	yes
3	20	random	22.12	6.31	18.69	10.53	0.000	yes
2	5	QRColPivot	6.62	2.24	3.41	1.64	0.006	yes
2	10	QRColPivot	6.32	1.59	18.66	7.87	0.009	
2	20	QRColPivot	8.66	3.82	30.16	0.52	0.000	
3	5	QRColPivot	6.53	2.18	5.13	3.77	0.104	
3	10	QRColPivot	6.23	1.58	18.71	7.51	0.006	
3	20	QRColPivot	8.32	3.47	28.13	0.33	0.000	

Table 2: Number of iterations for Concepts data, stopStandard experiments

			Standard		Multiscale			
					small		full	
level	k	initialization	mean	std	mean	std	mean	std
2	5	random	7.45	2.58	6.84	5.13	6.69	8.33
2	10	random	14.46	5.14	15.66	7.86	9.97	9.66
2	20	random	19.06	5.96	20.02	6.76	14.72	9.84
3	5	random	7.34	2.30	5.93	4.32	7.44	8.47
3	10	random	14.34	5.11	15.18	7.82	10.51	9.52
3	20	random	18.94	5.73	20.49	6.74	15.96	9.43
2	5	QRColPivot	6.00	2.71	3.70	2.06	3.10	1.97
2	10	QRColPivot	5.30	1.89	10.90	10.10	20.20	9.02
2	20	QRColPivot	5.90	3.28	21.20	8.23	25.00	0.00
3	5	QRColPivot	6.00	2.71	3.10	1.66	5.40	4.65
3	10	QRColPivot	5.30	1.89	9.60	10.67	20.90	8.72
3	20	QRColPivot	5.90	3.28	22.90	6.64	25.00	0.00

Table 3: Processing times for Faces data, stopStandard experiments

			Standard		Multiscale			
level	k	initialization	mean	std	mean	std	p-value	faster?
2	10	random	30.70	10.87	26.03	22.72	0.000	yes
2	20	random	36.46	11.42	32.86	23.03	0.000	yes
2	30	random	41.58	12.18	39.01	23.61	0.235	
3	10	random	30.77	11.54	26.59	21.57	0.000	yes
3	20	random	37.09	11.79	36.37	21.95	0.694	
3	30	random	41.52	12.24	40.55	21.82	0.024	yes
2	10	QRColPivot	20.26	9.68	47.27	18.52	0.006	
2	20	QRColPivot	20.09	3.28	62.98	2.70	0.000	
2	30	QRColPivot	26.69	5.29	68.88	2.06	0.000	
3	10	QRColPivot	19.54	9.60	44.88	18.01	0.006	
3	20	QRColPivot	20.16	2.98	61.20	1.50	0.000	
3	30	QRColPivot	26.49	5.46	66.00	1.06	0.000	

Table 4: Number of iterations for Faces data, stopStandard experiments

			Standard		Multiscale			
					small		full	
level	k	initialization	mean	std	mean	std	mean	std
2	10	random	13.05	4.94	13.85	7.78	9.79	10.00
2	20	random	15.01	4.97	14.63	7.16	12.27	9.75
2	30	random	16.58	5.17	16.20	7.09	14.30	9.66
3	10	random	12.75	5.10	10.11	7.28	10.70	9.55
3	20	random	15.24	5.14	12.14	7.29	14.73	9.55
3	30	random	16.74	5.14	15.18	7.48	15.91	9.15
2	10	QRColPivot	5.60	4.27	7.60	9.92	18.90	8.10
2	20	QRColPivot	5.60	1.35	15.90	11.84	25.00	0.00
2	30	QRColPivot	7.60	2.17	22.60	7.59	25.00	0.00
3	10	QRColPivot	5.60	4.27	9.50	10.09	19.00	8.11
3	20	QRColPivot	5.60	1.35	19.80	9.58	25.00	0.00
3	30	QRColPivot	7.60	2.17	25.00	0.00	25.00	0.00

Table 5: Random vs. QR factorization for Concepts data, stopStandard

			Random		QR			
level	k	approach	mean	std	mean	std	p-value	QR faster?
2	5	Standard	7.01	2.16	6.62	2.24	0.319	
2	10	Standard	13.80	4.54	6.32	1.59	0.000	yes
2	20	Standard	21.60	6.51	8.66	3.82	0.000	yes
3	5	Standard	7.40	2.03	6.53	2.18	0.069	
3	10	Standard	13.81	4.70	6.23	1.58	0.000	yes
3	20	Standard	22.12	6.31	8.32	3.47	0.000	yes
2	5	Multiscale	6.65	7.13	3.41	1.64	0.381	
2	10	Multiscale	10.18	8.78	18.66	7.87	0.037	
2	20	Multiscale	17.67	10.95	30.16	0.52	0.000	
3	5	Multiscale	7.29	7.29	5.13	3.77	0.925	
3	10	Multiscale	10.48	8.74	18.71	7.51	0.038	
3	20	Multiscale	18.69	10.53	28.13	0.33	0.022	

Table 6: Random vs. QR factorization for Faces data, stopStandard

			Random		QR			
level	k	approach	mean	std	mean	std	p-value	QR faster?
2	10	Standard	30.70	10.87	20.26	9.68	0.001	yes
2	20	Standard	36.46	11.42	20.09	3.28	0.000	yes
2	30	Standard	41.58	12.18	26.69	5.29	0.000	yes
3	10	Standard	30.77	11.54	19.54	9.60	0.001	yes
3	20	Standard	37.09	11.79	20.16	2.98	0.000	yes
3	30	Standard	41.52	12.24	26.49	5.46	0.000	yes
2	10	Multiscale	26.03	22.72	47.27	18.52	0.003	
2	20	Multiscale	32.86	23.03	62.98	2.70	0.000	
2	30	Multiscale	39.01	23.61	68.88	2.06	0.000	
3	10	Multiscale	26.59	21.57	44.88	18.01	0.010	
3	20	Multiscale	36.37	21.95	61.20	1.50	0.000	
3	30	Multiscale	40.55	21.82	66.00	1.06	0.000	

Table 7: Processing times for Concepts data, stopConverged

			Standard		Multiscale			
level	k	initialization	mean	std	mean	std	p-value	faster?
2	5	random	7.40	1.99	2.88	0.81	0.000	yes
2	10	random	14.67	4.82	4.69	3.22	0.000	yes
2	20	random	21.96	6.10	10.89	7.70	0.000	yes
3	5	random	7.35	1.96	2.85	1.14	0.000	yes
3	10	random	13.76	4.62	4.76	3.46	0.000	yes
3	20	random	22.03	6.21	12.32	8.06	0.000	yes
2	5	QRColPivot	6.60	2.20	2.54	0.07	0.000	yes
2	10	QRColPivot	6.27	1.61	4.86	2.89	0.212	
2	20	QRColPivot	8.22	3.50	9.44	6.90	0.791	
3	5	QRColPivot	6.54	2.18	3.12	1.81	0.001	yes
3	10	QRColPivot	6.25	1.60	5.34	3.36	0.273	
3	20	QRColPivot	8.23	3.50	6.39	3.20	0.089	

Table 8: Number of iterations for Concepts data, stopConverged

			Standard		Multiscale			
					small		full	
level	k	initialization	mean	std	mean	std	mean	std
2	5	random	7.36	2.29	7.44	2.38	2.13	0.88
2	10	random	14.28	5.17	13.85	5.32	3.66	3.55
2	20	random	19.12	5.68	16.12	6.03	8.77	7.10
3	5	random	7.29	2.18	7.52	2.67	2.21	1.28
3	10	random	14.57	5.08	13.28	5.41	4.41	3.91
3	20	random	19.18	5.78	14.28	5.69	10.64	7.47
2	5	QRColPivot	6.00	2.71	5.30	2.06	2.00	0.00
2	10	QRColPivot	5.30	1.89	7.20	4.94	4.50	3.44
2	20	QRColPivot	5.90	3.28	8.60	4.88	7.70	6.31
3	5	QRColPivot	6.00	2.71	4.50	1.78	2.90	2.23
3	10	QRColPivot	5.30	1.89	8.20	5.22	5.30	3.92
3	20	QRColPivot	5.90	3.28	6.70	2.91	5.20	2.97

Table 9: Processing times for Faces data, stopConverged

			Standard		Multiscale			
level	k	initialization	mean	std	mean	std	p-value	faster?
2	10	random	30.80	11.44	14.10	9.25	0.000	yes
2	20	random	36.75	11.28	15.73	10.25	0.000	yes
2	30	random	41.32	12.34	18.18	12.01	0.000	yes
3	10	random	31.07	11.22	14.64	9.51	0.000	yes
3	20	random	37.06	11.57	17.68	11.81	0.000	yes
3	30	random	42.77	13.18	23.33	14.11	0.000	yes
2	10	QRColPivot	20.07	9.39	17.68	14.88	0.162	
2	20	QRColPivot	19.88	3.05	18.32	9.44	0.678	
2	30	QRColPivot	25.11	5.10	12.42	5.15	0.001	yes
3	10	QRColPivot	19.44	9.42	18.71	10.71	0.473	
3	20	QRColPivot	19.81	3.05	22.35	14.72	0.791	
3	30	QRColPivot	25.00	5.05	22.38	11.53	0.212	

Table 10: Number of iterations for Faces data, stopConverged experiments

			Standard		Multiscale			
					small		full	
level	k	initialization	mean	std	mean	std	mean	std
2	10	random	13.01	5.17	12.26	5.38	4.47	4.24
2	20	random	14.90	4.93	13.28	5.18	4.96	4.51
2	30	random	16.55	5.20	13.99	5.21	5.70	5.06
3	10	random	12.98	5.07	11.77	5.33	5.33	4.29
3	20	random	15.11	5.09	11.81	5.04	6.57	5.24
3	30	random	16.66	5.30	11.57	5.73	8.40	5.81
2	10	QRColPivot	5.60	4.27	6.50	2.84	6.40	6.82
2	20	QRColPivot	5.60	1.35	8.20	6.29	6.40	4.12
2	30	QRColPivot	7.60	2.17	7.40	6.48	3.70	1.77
3	10	QRColPivot	5.60	4.27	8.20	7.38	7.30	4.83
3	20	QRColPivot	5.60	1.35	7.90	6.35	8.70	6.36
3	30	QRColPivot	7.60	2.17	5.80	3.68	8.40	4.81

Table 11: Random vs. QR factorization for Concept data, stopConverged experiments

			Random		QR			
level	k	approach	mean	std	mean	std	p-value	QR faster?
2	5	Standard	7.40	1.99	6.60	2.20	0.067	
2	10	Standard	14.67	4.82	6.27	1.61	0.000	yes
2	20	Standard	21.96	6.10	8.22	3.50	0.000	yes
3	5	Standard	7.35	1.96	6.54	2.18	0.065	
3	10	Standard	13.76	4.62	6.25	1.60	0.000	yes
3	20	Standard	22.03	6.21	8.23	3.50	0.000	yes
2	5	Multiscale	2.88	0.81	2.54	0.07	0.000	yes
2	10	Multiscale	4.69	3.22	4.86	2.89	0.525	
2	20	Multiscale	10.89	7.70	9.44	6.90	0.353	
3	5	Multiscale	2.85	1.14	3.12	1.81	0.001	
3	10	Multiscale	4.76	3.46	5.34	3.36	0.757	
3	20	Multiscale	12.32	8.06	6.39	3.20	0.018	yes

Table 12: Random vs. QR factorization for Faces data, stopConverged experiments

			Random		QR			
level	k	approach	mean	std	mean	std	p-value	QR faster?
2	10	Standard	30.80	11.44	20.07	9.39	0.001	yes
2	20	Standard	36.75	11.28	19.88	3.05	0.000	yes
2	30	Standard	41.32	12.34	25.11	5.10	0.000	yes
3	10	Standard	31.07	11.22	19.44	9.42	0.000	yes
3	20	Standard	37.06	11.57	19.81	3.05	0.000	yes
3	30	Standard	42.77	13.18	25.00	5.05	0.000	yes
2	10	Multiscale	14.10	9.25	17.68	14.88	0.949	
2	20	Multiscale	15.73	10.25	18.32	9.44	0.420	
2	30	Multiscale	18.18	12.01	12.42	5.15	0.103	
3	10	Multiscale	14.64	9.51	18.71	10.71	0.093	
3	20	Multiscale	17.68	11.81	22.35	14.72	0.375	
3	30	Multiscale	23.33	14.11	22.38	11.53	0.960	

Table 13: Errors for Concepts data

approach	level	k	initialization	mean	std	min	max
standard	2	5	random	0.595	2.400	0.075	19.141
standard	2	10	random	0.623	2.413	0.061	16.708
standard	2	20	random	0.757	2.573	0.081	27.904
standard	3	5	random	0.591	2.376	0.076	18.328
standard	3	10	random	0.621	2.403	0.060	16.007
standard	3	20	random	0.755	2.570	0.080	22.461
standard	2	5	QRColPivot	0.103	0.017	0.077	0.161
standard	2	10	QRColPivot	0.101	0.019	0.062	0.145
standard	2	20	QRColPivot	0.107	0.024	0.051	0.170
standard	3	5	QRColPivot	0.103	0.017	0.077	0.161
standard	3	10	QRColPivot	0.101	0.019	0.062	0.145
standard	3	20	QRColPivot	0.107	0.024	0.051	0.170
multiscale	2	5	random	0.103	0.232	0.075	36.670
multiscale	2	10	random	0.113	0.083	0.060	5.501
multiscale	2	20	random	0.189	0.159	0.085	8.854
multiscale	3	5	random	0.102	0.072	0.075	10.175
multiscale	3	10	random	0.116	0.106	0.063	7.791
multiscale	3	20	random	0.191	0.154	0.088	15.711
multiscale	2	5	QRColPivot	0.100	0.016	0.076	0.139
multiscale	2	10	QRColPivot	0.107	0.017	0.080	0.206
multiscale	2	20	QRColPivot	0.158	0.072	0.092	0.963
multiscale	3	5	QRColPivot	0.100	0.016	0.075	0.139
multiscale	3	10	QRColPivot	0.108	0.032	0.079	0.434
multiscale	3	20	QRColPivot	0.164	0.131	0.098	2.132

Table 14: Errors for Faces data

approach	level	k	initialization	mean	std	min	max
standard	2	10	random	0.390	0.978	0.098	20.593
standard	2	20	random	0.459	0.859	0.143	19.514
standard	2	30	random	0.505	0.772	0.178	18.677
standard	3	10	random	0.390	0.990	0.095	27.426
standard	3	20	random	0.456	0.830	0.144	23.336
standard	3	30	random	0.497	0.743	0.173	22.809
standard	2	10	QRColPivot	0.180	0.194	0.109	2.836
standard	2	20	QRColPivot	0.255	0.510	0.126	8.224
standard	2	30	QRColPivot	0.298	0.447	0.147	6.763
standard	3	10	QRColPivot	0.180	0.194	0.109	2.836
standard	3	20	QRColPivot	0.255	0.510	0.126	8.224
standard	3	30	QRColPivot	0.298	0.447	0.147	6.763
multiscale	2	10	random	0.195	0.361	0.093	24.298
multiscale	2	20	random	0.256	0.178	0.141	8.334
multiscale	2	30	random	0.299	0.142	0.178	12.532
multiscale	3	10	random	0.196	0.439	0.087	29.712
multiscale	3	20	random	0.258	0.252	0.141	24.689
multiscale	3	30	random	0.307	0.114	0.170	6.233
multiscale	2	10	QRColPivot	0.169	0.084	0.115	1.154
multiscale	2	20	QRColPivot	0.261	0.170	0.154	2.386
multiscale	2	30	QRColPivot	0.337	0.123	0.222	1.127
multiscale	3	10	QRColPivot	0.185	0.118	0.115	1.365
multiscale	3	20	QRColPivot	0.325	0.623	0.179	9.306
multiscale	3	30	QRColPivot	0.303	0.083	0.221	0.839

Table 15: Classification accuracy for Concepts data

			Standard		Multiscale			
level	k	initialization	mean	std	mean	std	p-value	accuracy diff.?
2	5	random	35.39%	0.0344	35.42%	0.0368	0.955	
2	10	random	35.56%	0.0316	35.66%	0.0349	0.832	
2	20	random	34.46%	0.0352	34.88%	0.0377	0.415	
3	5	random	35.46%	0.0370	35.51%	0.0336	0.920	
3	10	random	35.48%	0.0349	36.02%	0.0374	0.290	
3	20	random	34.06%	0.0360	34.29%	0.0337	0.644	
2	5	QRColPivot	36.13%	0.0360	36.24%	0.0347	0.819	
2	10	QRColPivot	35.34%	0.0318	35.65%	0.0355	0.510	
2	20	QRColPivot	35.76%	0.0360	34.85%	0.0357	0.075	
3	5	QRColPivot	35.28%	0.0347	35.80%	0.0358	0.297	
3	10	QRColPivot	34.93%	0.0312	35.01%	0.0339	0.853	
3	20	QRColPivot	35.56%	0.0303	35.00%	0.0270	0.172	

Table 16: Classification accuracy for Faces data

			Standard		Multiscale			
level	k	initialization	mean	std	mean	std	p-value	accuracy diff.?
2	10	random	99.87%	0.0101	99.47%	0.0218	0.132	
2	20	random	99.69%	0.0171	98.19%	0.0411	0.001	yes
2	30	random	99.50%	0.0219	96.10%	0.0549	0.000	yes
3	10	random	99.87%	0.0100	97.28%	0.0469	0.002	yes
3	20	random	99.19%	0.0307	93.29%	0.0817	0.000	yes
3	30	random	99.20%	0.0307	86.80%	0.1014	0.000	yes
2	10	QRColPivot	99.97%	0.0004	99.87%	0.0100	0.884	
2	20	QRColPivot	99.89%	0.0100	99.49%	0.0220	0.897	
2	30	QRColPivot	100.00%	0.0001	99.80%	0.0141	0.461	
3	10	QRColPivot	99.97%	0.0004	99.77%	0.0141	0.926	
3	20	QRColPivot	99.99%	0.0003	99.49%	0.0220	0.478	
3	30	QRColPivot	99.90%	0.0100	99.00%	0.0302	0.003	yes