Image compression using Generalized Low Rank Approximation of Matrices

Rishu Saxena
Computer Science Department
Virginia Tech
Blacksburg, VA
rishus@vt.edu

ABSTRACT

Image processing scientists and associated domain scientists are striving to accomodate the dulge of data that has come about in the past decade thanks to rapid advancements of technology as well wider availability of data. The data dulge, the wide gamut of utilities and increasingly multidimensional nature of modern images has created an impending need for more efficient storage and processing mechanisms for images and has encouraged much research in the direction. The goal here is to achieve data compression without compromising on the quality of images.

In this project, we study the use of generalized low rank approximations of matrices for image compression. This approach was proposed in [6] in 2005. Low rank approximations of single matrices have been used very successfully for various purposes. In this paper, the author generalizes the approach to multiple matrices, thus adapting the approach for analyzing datasets of images (rather than having to process one image at a time). We run our experiments on a facial data dataset that was used by the author as well as on a remote sensing dataset. Significant compression is achieved. Comparison between The results of reconstruction and classification obtained from generalized low rank approximation approach and the results obtained using SVD is also presented; the recontructions with the current approach are much better. The proposed approach also achieves much better time complexity. Since remote sensing datasets, in particular, are plagued by massive amounts of missing data, such methods of compression can be particularly useful to them.

Categories and Subject Descriptors

H.3.3 [**Information Search and Retrieval**]: Miscellaneous

General Terms

Application

Keywords

Low rank approximation, classification, Singular Value Decomposition (SVD), time-series.

1. INTRODUCTION (PAPER SUMMARY)

Low rank approximation is a minimization problem that seeks to minimize the fit between a given matrix (the data) and an approximating matrix subject to the condition that the approximating matrix has a lower rank than the original (given) matrix. Mathematically, the general idea is to solve the following problem:

$$\min \|A - \hat{A}\|, \tag{1}$$

such that

$$rank(\hat{A}) \leq k, \tag{2}$$

where k is the desired rank. Low rank approximations find application in several domains such as machine learning, recommender systems, natural language processing, image processing and the like. The problem is well studied by researchers from various communities and several strategies for computing low rank approximations have been proposed. Traditionally, is singular value decomposition (SVD) has been the most popular mthod of choice for computing low rank approximations. Other popular algorithms availabale today include: Alternating projections algorithm [1], Variable projections algorithm [4], CUR matrix approximation [5], and the like. However, all of these algorithms are designed for single matrices. To process a dataset such as a collection of matrices, these algorithms wrap the individual images (say, of size $r \times c$) into vectors (say, of size $rc \times 1$) stack together these vectors into columns of a matrix (so each column corresponds to one image) and implement the available algorithm on this large image. Unfortunately, as the datasets get bigger (due to increase in size of image datasets as well as increasingly multidimensional nature of images) this approach and the traditional algorithms lose scalability. This is a tremendous drawback in the current era of Big Data and when most images are multidimensional. Remote sensing datasets, for example, are already of the size of terabytes and are expected to grow to petabytes in the coming few years. Before this paper was published, some other incremental algorithms had also been proposed [2, 3] but these algorithms did not guarantee the much needed attributes of stability, convergence and the like.

The paper we discuss here was published in 2005. It proposed a novel iterative algorithm for low rank approximations of a sequence of matrices, which, as we will show in the

paper, achieves remarkable data compression. The spatial and temporal efficiency achieved in the proposed algorithm is still better than that of any of the other state-of-the-art approaches, including the ones mentioned earlier. This paper, thereofore, is of significant interest. Broadly, the algorithm presented in the paper it is designed to work directly on a sequence of images (and thus, altogether avoids the need for building the large matrix mentioned earlier). Secondly, it demostrated more accurate results in data compression, image retrieval and image classification. Finally, it's time complexity is much less than that of traditional algorithms.

The authors make a quick summary on SVD as they observing a theorem on SVD:

THEOREM 1. *Let the Singular Value Decomposition of* $A \in R^{N\times n}$ *be* $A = UDV^t$, *where* U *and* V *are orthogonal matrices and* $D = diag(\sigma_1, \ldots, \sigma_r, 0 \ldots, 0), \sigma \geq \ldots, \sigma_r > 0$ *and* $r = rank(A)$. *Then for* $1 \leq k \leq r$, $\sum_{i=k+1}^{r} \sigma_i^2 = \min\{\|A - B\|_F^2 : rank(B) = k\}$. *The minimum is achieved with* $B = best_k(A)$, *where* $best_k(A) = U_k diag(\sigma_1, \ldots, \sigma_k)V_k^t$ *are the matrices formed by the first k columns of* U *and* V, *respectively.*

To construct the main algorithm, mathematical formulations and manipulations are vividly discussed. The authors go on to prove three different theorems:

THEOREM 2. *Let* L, R *and* $\{D_i\}_{i=1}^n$ *be the optimal solution to the minimization problem in Eq. (1). Then* $D_i = L^t A_i R$, *for every* i.

Meaning, once L and R are determined, D_i is uniquely determined by $D_i = L^t A_i R$.

THEOREM 3. *Let* L, R *and* $\{D_i\}_{i=1}^n$ *be the optimal solution to the minimization problem in Eq. (1). Then* L *and* R *solve the following optimization problem:*

$$\max \sum_{i=1}^{n} \|L^t A_i R\| \tag{3}$$

where, $L \in R^{r\times l_1}, R \in R^{c\times l_2}$ *and* L *and* R *have orthogonormal column vectors, i.e.,* $L^t L = I_{l_1}$ *and* $R^t R = I_{l_2}$.

THEOREM 4. *Let* L, R *and* $\{D_i\}_{i=1}^n$ *be the optimal solution to the minimization problem in Eq. (1). Then*

1. *For a given* R, L *consists of the* l_1 *eigenvectors of the matrix* $M_l = \sum_{i=1}^{n} A_i R R^t A_i^t$ *corresponding to the largest* l_1 *eigenvalues.*
2. *For a given* L, R *consists of the* l_2 *eigenvectors of the matrix* $M_R = \sum_{i=1}^{n} A_i^t L L^t A_i$ *corresponding to the largest* l_2 *eigenvalues.*

Theorem 4 is the main theorem that helps us build an iterative method for computing L and R. Basically, for a given L, we can compute R by computing the eigenvectors of the matrix M_R. With the computed R, we can then update L by computing the eigenvectors of the matrix M_L. The procedure can be repeated until convergence. The pseudocode presented in the paper is displayed in algorithm 1 (with some minor details additionally included)

Algorithm 1 GLRA($\{A_i\}_{i=1}^n$)

1: Initialize L
2: Assign the error tolerance for convergence τ.
3: **while** $\epsilon > \tau$ **do**
4: Compute $M_R = \sum_{j=1}^{n} A_j^t L L^t A_j$.
5: Compute the eigenvectors $\{\phi_i^R\})_{i=1}^n$ of M_R corresponding to the l_2 largest eigenvalues.
6: $R \leftarrow [\phi_1^R, \ldots, \phi_{l_2}^R]$
7: Form the matrix $M_L = \sum_{j=1}^{n} A_j^t R R^t A_j$.
8: Compute the eigenvectors $\{\phi_i^L\})_{i=1}^n$ corresponding to the l_1 largest eigenvalues of M_L.
9: $L_{new} \leftarrow [\phi_1^L, \ldots, \phi_n^L]$
10: $\epsilon = \|L_{new} - L\|_F$
11: $L = L_{new}$
12: **end while**
13: **for** $j = 1 : n$ **do**
14: $D_j = L^t A_j R$.
15: **end for**

Convergence and accuracy:. By theorem 4, the updates in lines 6 and 9 of algorithm 1 increases the value of $\sum_{i=1}^{n} \|L_i^A R\|_F^2$. Therefore, by theorem 3, it is straightforward to conclude that the root mean square error in recostruction using Frobenius norm, given as

$$RMSRE = \sqrt{\frac{1}{n}\sum_{i=1}^{n} \|A_i - L D_i R^t\|_F^2} \tag{4}$$

also decreases. Since, by formula itself, RMSRE is bounded from below by 0, the algorithm is guarantedd to converge.

Complexity:. The time complexity of the algorithm is $O(\mu(r+c)^2 \max(l_1, l_2)n)$, where μ is the number of iterations in the while loop.

The space complexity can be calculated to be $O(rc)$.

The details of these calculations are presented by the authors in the paper.

2. EXPERIMENTS

I implemented the proposed algorithm in python. For the proposed algorithm itself, the only packages I needed were os, numpy, and re. The implementation was straightforward. The results if the proposed algorithm were compared with those of the popular SVD approach. For SVD calculations,

I used the inbuilt commands from linalg package of python. All the experiments were carried out on a Lenovo-ThinkPad, x86, 64-bit 4-core laptop with 8GB memory and 2GHz CPU. All the results shown here use $d = 20$.

2.1 Datasets

I ran the code on the ORL and other datasets discussed in the paper first. Results for ORL dataset are presented in the subsections to follow. ORL dataset is a freely available face recognition dataset containing pictures of 40 subjects, ten pictures per subject (so, 400 pictures in all). Subsequently, I also experimented with a few other datasets. Two of those are discussed in this presentation.

First was a leaves dataset available freely a computer vision group at Caltech (http://www.vision.caltech.edu/html-files/archive.html). This dataset consists of 186 images of leaves of three different species. Each image is 896 x 592 pixels in jpg format.

The second dataset is the Menu-Match dataset, available on the website of a visions group at UCSD (http://vision.ucsd.edu/content/menu-match-dataset). The Menu-Match dataset include images of meals from three restaurants: an Asian restaurant offers a buffet-style setup where customers select 1-3 toppings that are served with a fixed serving size on top of brown or white rice; an Italian restaurant offers a variety of pizzas, lasagnas, and pastas, served with sides of breadsticks or salad; and a soup restaurant offers 10 soups with a side of one of 5 breads. For this dataset, we had the choice of how many classes we want to work with. For example, we could have done classification of images to precisely pin-point the individual menu item it belonged to; that would have lead to 41 classes. However, for this presentation, we decided to adhere to classify a food-image in terms of the cuisine it belonged to. So we were working with only three classes, each class corresponded to one cuisine. Relevant details of the datasets are presented in table 1.

Dataset	size	image size	# classes
Face	400	10304	40
Leaves	186	530432	3
Menu	646	originally varied but cropped to 512 $512 \times 512 =$	3

Table 1: Datasets presented in this report.

2.2 Performance

That the algorithm certainly brings improved time complexity is demostrated in figure 1. For ORL dataset, the matrices L, D and R were computed in less than one second ($\approx$ 0.64741 second, specifically). Figure 1 compares the time taken by the proposed algorithm with the time taken by SVD to process each of the three input datasets. For Leaves and Lenu-Match datasets, SVD resulted in segmentation fault on my computer because of it's memory requirements. (For presentaion sake, the SVD times have been displayed as 130 seconds, a placeholder number.) The proposed algorithm, on the other hand, ran without any difficulty, hence proving that it not only runs faster than SVD but it also requires lesser memory than SVD. Using $d = 20$, the algorithm achieved compression ratios of about 25% for the ORL dataset and and 30% for the Leaves and the Menu-Match datasets.

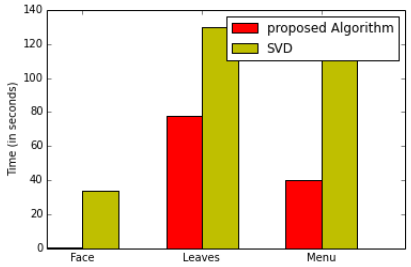

Figure 1: Time take for the proposed algorithm to run on the three datasets compared to the time take by SVD. SVD times are (displayed as 130 seconds here) but in reality, SVD never finished running for the Leaves and Menu datasets on my computer due to memory constraints.

Effect of value of d.. The accuracy of classification varies with the parameter d. Some results are shown in figure 2. Of the calues we experimented with (also based on the original paper), $d = 20$ seems to be working best on all the three datasets we use in this writ-up. Therefore, all the experiments from here on are presented using this value.

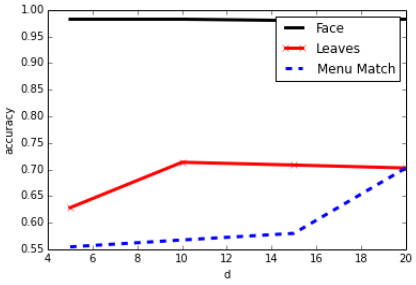

Figure 2: Accuracy of classification as a function of the parameter d when the compressed images are used.

2.3 Classification

The compressed images obtained using the proposed approach were used as inputs to a K-nearest neighbor classifier for classification and the accuracy was observed. I used ten-fold cross validation. The experiments were performed with different values of K but the best results were obtained using $K = 1$. The same strategy was applied on the compressed images obtained using SVD. As displayed in Figure 3, accuracy with the proposed approach is always better than that with SVD. Figure 4(a) compares the precisions, recall and fstatic values of classification results obtained using the proposed algorithm with those using SVD on ORL dataset.

Proposed algorithm clearly outperforms SVD on all counts. Figure 4(b) displays the results with respect to the three datasets we use in this paper.

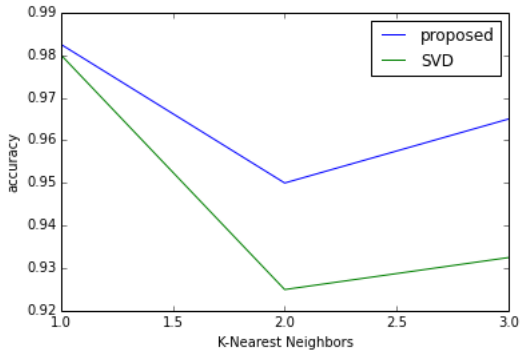

Figure 3: ORL dataset. Classification accuracy as a function of number of neighbors when classification is carried out using K-Nearest Neighbor method with 10-fold cross-validation.

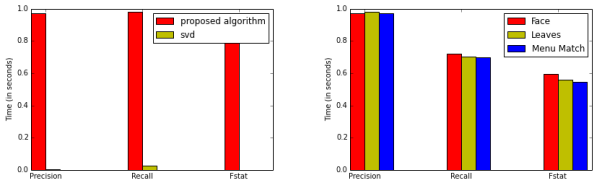

Figure 4: $d = 20$. Precision, recall and fstatic values for the classification results obtained using K-NN (with K = 1) using ten fold cross validation and the compressed images that were produced by the proposed algorithm. (a) Proposed algorithm compared to SVD on ORL dataset; (b) proposed algorithm on each of the three datasets.

2.4 Reconstruction

Figure 5 displays the images of five subjects. The first row is the original image. The second row shows the compressed images obtained using the proposed approach. The third row shows the images obtained using SVD. Clearly, the proposed approach preserves much more visual quality than SVD.

Figure 6 displays the images of three leaves, one of each specie (there were three species in in this dataset). The first row, again, has the original images while the second row displays the corresponding compressed images. SVD's memory requirements exceeded the memory available on this machine.

Figure 7 shows randomly picked (though, at least one of kind) sample images from the Menu-Match dataset and the compressed images obtained using the proposed algorithm. We notice some blurring in some of the compressed images. This blurring can be attributed to the zero-padding and huge

Figure 5: ORL dataset, $d = 20$. The original image (top row), images compressed by the proposed approach (middle row) and the images compressed by SVD (bottom row).

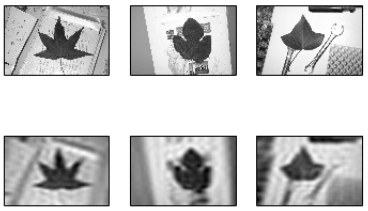

Figure 6: Leaves dataset, $d = 20$. The original image (top row), images compressed by the proposed approach (second row). SVD ran out on memory for this dataset on the computer being used.

amounts on image cropping that we did to ensure that all the input images were of equal size. Better preparation of input will reduce the blurring effect largely. Once again, SVD failed to run for this dataset on our computer.

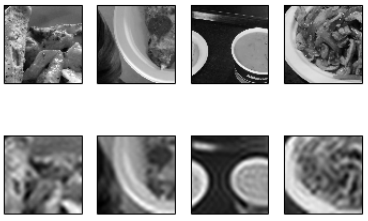

Figure 7: Menu-Match dataset, $d = 20$. The original image (top row), images compressed by the proposed approach (second row). SVD ran out on memory for this dataset on the computer being used.

3. DISCUSSION AND CONCLUSIONS

The most significant feature of the algorithm proposed in [6] lies in it's capability to process sequences of data while still being able to avoid cumbersome, and frequently computationally intractable, data structures (eg., large matrices). The design is based on sound mathematical derivation presented in the paper. The algorithm is indeed computationally and spatially more efficient than other traditional algorithms. The numerical experiments presented here demonstrate that SVD becomes intractable as the size of datasets increases but the proposed algorithm still runs in seconds. Summarizing, the author's claims are verified both on the datasets he used and on the new datasets we experimented with:

1. Data compression: The algorithm efficiently compresses data and the compression obtained using the proposed algorithm preserves the data quality much better than SVD. Classification using this compressed data is still accurate.

2. The algorithm is temporally more efficient than other state-of-the-art-algorithms (of the time when the paper was published), including SVD.

In general, the proposed algorithm outperforms SVD inevery respect. Finally, although the experiments in this paper were presented with respect to image processing application only, the algorithm can actually be used for a large gamut of applications.

Difficulties faced. The actual implementation of the algorithm as well as the SVD went smooth for us. The primary difficulty that we faced in this project was obtaining the datasets we wanted to work with. Our original idea was to work with remote sensing images and medical images. However, we could not find freely available sufficiently large datasets of these images and had to present our results with the datasets we could find. Secondly, working with the menu dataset was turned out to be more involved that we had expected. The images in this dataset do not come in an organized format (eg., organized by class labels). While downloading the the image stack was trivial, arranging the more than six hundred images required more work. To this end, we wrote some bash scripts that downloaded the html files from the website, parsed those to get the names of the menu items and the respective cuisines, then collected the names of the image files for each menu item, and finally created three folders ('asian', 'italian', and 'soup') and put the respective image files in the corresponding folders. Secondly, while reading the image files into the code (matrix), the images turned out to be of random sizes. Fitting them into a matrix in an educated way, both for the proposed framework as well as for the SVD framework, took time. Apart from these more successful experiments, a lot of our time went in preparing the landsat imagery dataset we had been able to download for this experiment. Unfortunately, we could not get those ready to be used for the code and had to settle with the smaller datasets presented in this paper.

3.1 Future Work

The proposed algorithm is indeed very promising in every way. However, although it has achieved much scalability and space efficiency, both these aspects can be further improved upon. Improving the running time and memory requirements further is one potential area of future work in this context.

Satellite imagery (or, remote sensing data), that we had originally wished to experiment with, is characterized by large chunks of missing data. In other words, the data is sparse. In addition, remote sensing datasets are also very large (order of terabytes and growing exponentially). These datasets, in particular, can benefit immensely from scalable image processing algorithms, and thus, present a powerful avenue to implement/demostrate the algorithm on. Doing so will be a grat test for the algorithm and also a great avenue to further develop the algorithm for large, sparse, multidimensional datasets.

Another arena that is overwhelmed with large, multidimensional image is medical imaging.

In general, we wish to advance the algorithm for scalable (and parallel) computing. Parallel implementation will alleviate the intractability of SVD. Additionally, while the presented algorithm relies only on eigenvectors, it would be interesting to replace/complemet the use of linear algebra with ideas from robust statistics. This is our longer term goal.

4. AUTHOR CONTRIBUTIONS

I did the project by myself. So all the implementation and writing was done by me.

5. REFERENCES

[1] J. P. Boyle, and R. L. Dykstra, *A method for finding projections onto the intersection of convex sets in Hilbert space*, Lecture Notes in Statistics, Vol. 37, pp 28 – 47, 1986.

[2] M. Gu, and S. C. Eisenstat, *A fast and stable algorithm for updating the singular value decomposition*, Technical Report Technical Report YALEU/DCS/RR-966, Department of Computer Science, Yale University, 1993.

[3] K. V. R. Kanth, D. Agrawal, A. E. Abbadi, and A. Singh, *Dimensionality reduction for similarity searching in dynamic databases*, CM SIGMOD Conference Proceedings, 166 – 176, 1998.

[4] D. P. O'Leary, and B. W. Rust, *Variable Projection for Nonlinear Least Squares Problems*, Computational Optimization and Applications, Vol. 54, No. 3, pp 579 – 593, 2013.

[5] M. W. Mahoney, and P. Drineas, *CUR matrix decompositions for improved data analysis*, Proceedings of the National Academy of Sciences of the United States of America, Vol. 106, No. 3, pp 1 – 3, 2008.

[6] J. Ye, *Generalized Low Rank Approximations of Matrices*, Machine Learning, Vol. 61, No. 1 – 3, 2005.