CLUSTERING BEFORE TRAINING LARGE DATASETS - CASE STUDY: K-SVD

Cristian Rusu

University "Politehnica" of Bucharest
Department of Automatic Control and Computers
Spl. Independenţei 313, Bucharest 060042, Romania (e-mail: cristian.rusu@schur.pub.ro)

ABSTRACT

Training and using overcomplete dictionaries has been the subject of many developments in the area of signal processing and sparse representations. The main idea is to train a dictionary that is able to achieve good sparse representations of the items contained in a given dataset. The most popular approach is the K-SVD algorithm and in this paper we study its application to large datasets. The main interest is to speedup the training procedure while keeping the representation errors close to some specific values. This goal is reached by using a clustering procedure, called here T-mindot, which reduces the size of the dataset but keeps the most representative data items and a measure of their importance. Experimental simulations compare the running times and representation errors of the training method with and without the clustering procedure and they clearly show how effective T-mindot is.

Index Terms— sparse representations, clustering, K-SVD.

1. INTRODUCTION

The field of sparse representations [1] has enjoyed increased popularity in recent years mainly due to the theoretical developments [2] [3] and the large class of applications where it is used [4] [5]. Now, the focus remains on two central problems of the field: given a signal $\boldsymbol{a} \in \mathbb{R}^n$ find its sparse (or sparsest) representation in a given base $\boldsymbol{D} \in \mathbb{R}^{n \times d}$ called dictionary; and given a whole dataset $\boldsymbol{A} \in \mathbb{R}^{n \times N}$ find a dictionary in which the data items have a good approximate sparse representation.

To solve the first class of problems, algorithms like the orthogonal matching pursuit (OMP) [6] were developed. This is a greedy approach that builds a representation by picking, in an iterative procedure, the item of the dictionary that significantly reduces the representation error. Due to its simplicity and with an efficient implementation [7], OMP is a very fast algorithm. For the second class of problems, the most popular and effective method is the K-SVD algorithm [8] [9]. This is also an iterative procedure that uses OMP to build the representations of the data items and then the singular value decomposition (SVD) to update the dictionary columns such that the overall mean squared error is reduced. Because of this structure, the K-SVD is relatively slow when applied for large datasets. We address this issue by substantially reducing the dataset in such a way that we keep its basic structure and do not negatively affect the performance of the training procedure. The K-SVD runs on the reduced dataset and the running time and representation error are both compared with the application of K-SVD on the full dataset.

A similar idea to the one proposed in this paper that comes from the field of computational geometry is the use of coresets [10]. In the field of machine learning, in the context of supervised learning, reduction techniques using exemplar-based algorithms [11] have been used to reduce the memory consumption, speedup the training procedures and reduce the sensitivity to noise. From this class of methods we mention here the nearest neighbor algorithm [12] and case-based reasoning [13].

The paper is organized as follows: Section 2 describes the proposed clustering procedure, called T-mindot, Section 3 presents the experimental results obtained on various datasets and Section 4 concludes the paper.

2. THE PROPOSED CLUSTERING PROCEDURE

The goal of this paper is to present a clustering procedure that takes place before applying the K-SVD algorithm in order to reduce the size of the training set and consequently to reduce the running time of K-SVD.

This approach is determined by the practical need to have the K-SVD algorithm run fast even when the training set is very large (contains at least a few tens of thousands of items). In this research direction we can find for example the approximate K-SVD (AK-SVD) [7] algorithm, in which the whole SVD step is replaced by few iterations of the power method. The proposed clustering procedure is inspired by the observation that the AK-SVD algorithm produces very good results even though the power method step runs for only one iteration

This work was supported by the Romanian National Authority for Scientific Research, CNCS UEFISCDI, project number PN-II-ID-PCE-2011-3-0400 and by the Sectoral Operational Programme Human Resources Development 2007-2013 of the Romanian Ministry of Labour, Family and Social Protection through the Financial Agreement POSDRU/88 /1.5/S/61178.

(the whole SVD step is replaced by a few matrix multiplications).

It seems that K-SVD can provide very good results even if it is provided with a rough approximation of the general direction in which the training samples group.

In this case, the first idea that comes to mind is to cluster the training samples by a proximity measure, taken here to be the absolute value of the normalized dot product, such that the general direction is kept but the number of data samples is greatly reduced. In such a context, the k-means clustering procedure [14] could be used. We employ a similar strategy but, in our case, we fix the maximum allowed distance between the training samples and any of the centroids such that similar data items can be replaced with a single representative centroid. When none of the current centroids can provide such a distance to an item, we increase the number of centroids by adding the new training sample to the set of centroids.

Given the training samples $\boldsymbol{a_i} \in \mathbb{R}^n, \forall i = 1, \ldots, N$ with $||\boldsymbol{a_i}||_2 = 1$ and the maximum allowed distance between the centroids and their clustered items T, the ouput of the clustering procedure is the centroid set $\mathbb{K} = \{\boldsymbol{b_j}\}, j = 1, \ldots, M$ with $M \ll N$ and the general formulation of the optimization problem is given by

$$\begin{aligned} \underset{\boldsymbol{b_j}}{\text{minimize}} \quad & M \\ \text{subject to} \quad & |\boldsymbol{a_i}^T \boldsymbol{b_j}| \geq T, \forall i, \text{ for at least one } j \\ & ||\boldsymbol{b_j}||_2 = 1. \end{aligned} \tag{1}$$

Since this is not a convex optimization problem, we develop a heuristic that finds a feasible solution that is able to reach, in the general case, a conveniently small M in a reasonable amount of time. Thus, the heuristic has to balance a compromise between the running speed of the algorithm and the quality of the solution found.

The proposed clustering procedure, called T-mindot, is presented in Table 1.

The input matrix $\boldsymbol{A}$ contains the normalized training vectors columnwise concatenated.

The initialization sets: the centroid set $\mathbb{K}$ containing the first training vector $\boldsymbol{a_1}$, the frequency vector $\boldsymbol{f} = 1$ that keeps track of how many training vectors each centroid clusters, the parameter S that decides the dimension of the working blocks, S_{max} the upper bound of the dimension of the blocks imposed mainly because of memory limitations and two parameters D_{fast} and D_{slow} that control the speed of the procedure with respect to the dimensions of the data blocks.

The goal of T-mindot is to reduce the dimension of the dataset by replacing data items that are close with only one representative item whose weight is proportional to the number of items it clusters. Since computing all the dot products between all the data items is too time consuming, this is done by computing dot products between the current set of centroids and a subset of size S of the yet unclustered data items.

Table 1: General overview of the T-mindot clustering procedure

```
Input:       $\boldsymbol{A} \in \mathbb{R}^{n \times N}$ - training matrix
             $T \in \mathbb{R},\ 0 \ll T < 1$ - the maximum allowed
             error (minimum dot product)
Parameters:  $\mathbb{K}$ - set of centroids
             $\boldsymbol{f}$ - number of items clustered by each
             centroid
             $S$ - current working dimensions
             $S_{\text{max}}$ - maximum working dimensions
             $\text{D}_{\text{fast}}$ - control of fast dynamic
             $\text{D}_{\text{slow}}$ - control of slow dynamic
Procedure:
  1. while (training set is not empty)
     (a) S = $\min\{S, S_{\text{max}}\}$
     (b) extract set $\mathbb{W}$ of size $S$ with the first available
         training vectors from the training matrix $\boldsymbol{A}$
     (c) compute the distances between the extracted
         vectors and the centroids in $\mathbb{K}$
     (d) for each training vector $\boldsymbol{a_i} \in \mathbb{W}$ that has a
         centroid $\boldsymbol{c_j}$ such that $|\boldsymbol{a_i}^T \boldsymbol{c_j}| \geq T$
           i. add $\boldsymbol{a_i}$ to the set of vectors clustered by
              $\boldsymbol{c_j}$ and eliminate them from $\mathbb{W}$
          ii. $f_j = f_j + 1$
     (e) if more than $5\%$ of vectors are not clustered
         after the previous step then $S = \lceil S \times \text{D}_{\text{slow}} \rceil$,
         else $S = \lceil S \times \text{D}_{\text{fast}} \rceil$
     (f) while ($\mathbb{W} \neq \emptyset$)
           i. find $\boldsymbol{a_i} \in \mathbb{W}$ that groups around it the
              most training samples from $\mathbb{W}$
          ii. remove $\boldsymbol{a_i}$ and its grouping from $\mathbb{W}$ and
              add it as new centroid
  2. for each centroid $\boldsymbol{c_j} \in \mathbb{K}$
     (a) compute new $\boldsymbol{c_j}$ to be the average of the vectors $\boldsymbol{a_i}$ clustered by $\boldsymbol{c_j}$
     (b) normalize new centroid
Output :     $\boldsymbol{C} \in \mathbb{R}^{n \times M}$ - matrix with centroids
             columnwise concatenated
             $\boldsymbol{f} \in \mathbb{R}^M$ - number of vectors clustered by
             each centroid
```

In order to speedup the running time of the algorithm some operations are done in bulk. For example, the computation of the dot products in step (c) is done efficiently with a matrix-matrix multiplication whose dimensions depend on the parameter S.

If a large percentage (over 95%) of the training vectors find a centroid in step (d) then we are encouraged to increase the current working size S since all computations are done in bulk and are thus more efficient. If not, then the dot products computed are useless and the training vectors are dealt with in step (f). The issue is that step (f) is much slower than steps (c) and (d) because it has to take care of the remaining training vectors one by one. Varying the value of S we balance the two steps and we try to make sure that: useful computations are done in bulk in step (d) and we reduce the number of items that end up in step (f). For a given T, the quality of the solution is measured by the difference $P = N - M \geq 0$. We are looking to find solutions that have a neglijable magnitude for M, such that P is very close to N.

The values of $D_{\text{fast}} = 1.1$, $D_{\text{slow}} = 0.9$, the current working block size $S = \max\{\lfloor 0.01 \times N \rfloor, 500\}$, the maximum working block size $S_{\max} = \max\{\lfloor 0.02 \times N \rfloor, 2000\}$ and the 5% limit that decides whether the algorithm employs a fast or slow dynamic were chosen after a set of experimental runs on large datasets were conducted. These parameters seem to best balance between the running speed and the performance of clustering. The training samples are processed in the order they appear in the dataset, no randomization takes place in this implementation.

The second step of T-mindot centers every centroid in its training samples group in order to reduce the average error, allowing for the possibility that the dot products may be lower than T (which happens in only a small fraction of cases). Experiments show that this centering step helps improve the representation errors.

The output of T-mindot, the matrix $\boldsymbol{C}$ and the vector $\boldsymbol{f}$, is used in all experimental runs of K-SVD. The initial training dataset is replaced by a new training matrix consisting of the weighted centroids. This new training matrix should be considerably smaller than the initial one and provide a significant speedup while the weights ensure that the dictionary behaves well also against the original training dataset by providing a small representation error.

The fact that the clustered items are highly correlated indicates the fact that they most likely have the same sparse support and applying the training procedure on the centroids weighted by the square root of the number of items they cluster is equivalent (from the SVD point of view) to applying the training procedure on a dataset consisting of the centroids repeated as many times as their weights indicate. The weight of each centroid is computed by taking the square root of its frequency since we are interested in the singular values and vectors computed from the data items and we want to preserve their relative importance.

Table 2: Dimensions of the audio training sets

training set	set dimension (N)
classical	245498
electronic	100215
jazz-blues	22747
metal-punk	38524
rock-pop	86050
world	105802

This arrangement leads to an important training speedup and a low representation error for the *original* dataset.

Albeit the coresets approach [16] is similar, when compared to T-mindot there are differences in key points: the items are selected to become centroids by a slower method that is based on a probability measure that takes into account the distance from the items to the subspace spanned by the dictionary, the number of centroids is an input parameter of the algorithm and all centroids have the same weight in the reduced dataset.

3. EXPERIMENTAL RESULTS

Next, we present a set of results obtained in two circumstances using T-mindot before the application of the AK-SVD algorithm.

In the first setting, the AK-SVD algorithm is applied on a large set of vectors that contain features (linear predictive coding coefficients [15]) extracted from audio signals. As source, the publicly available ISMIR 2004 [17] audio database was used. The audio signals are partitioned in windows of 3000 samples (about 250 ms) with overlap 750 samples (about 60 ms) and 120 LPC coefficients are computed for each partition. We are interested in two major performance indicators: the speed of the training procedure and the quality of the results which is represented by the Frobenius norm of the remaining error matrix ($||\boldsymbol{E}||_F^2 = \sum_i \sum_j E_{ij}^2$). In terms of speed, we are interested to compare the AK-SVD algorithm applied on the training vectors with and without T-mindot executed. Clearly, since we reduce the number of training samples, we expect the running time of AK-SVD to go down significantly in both of its main stages: the computation of the new atoms using the SVD and the computation of the new representations of all the training vectors using OMP.

After the training is done, using the same initial dictionary and the same number of items in each sparse representation, we compare the representation errors of both computed dictionaries against the *original* set of training vectors. The representation error is described by the Frobenius norm of the difference between the initial training set and the reconstructed set of vectors. Here, we expect the errors introduced by T-mindot to be small.

Table 3: Frobenius norms of the audio representation error matrices

	AK-SVD	T-mindot + AK-SVD (+cut)	
training set	-	$T = 0.95$	$T = 0.9$
classical	251	252 (253)	252 (252)
electronic	150	150 (154)	151 (151)
jazz-blues	69	70 (75)	70 (72)
metal-punk	85	85 (92)	86 (87)
rock-pop	148	148 (154)	149 (150)
world	172	172 (177)	172 (174)

Table 4: Running times for audio simulations (in seconds)

	AK-SVD	T-mindot + AK-SVD (+cut)	
training set	-	$T = 0.95$	$T = 0.9$
classical	1886	1238 (570)	745 (371)
electronic	796	474 (120)	348 (110)
jazz-blues	103	73 (35)	57 (51)
metal-punk	182	120 (55)	72 (51)
rock-pop	410	242 (103)	153 (97)
world	929	561 (231)	357 (172)

Table 5: Frobenius norms of the image representation error matrices

		AK-SVD	RND+AK-SVD	T-mindot + AK-SVD	
k	k_0	-	-	$T = 0.95$	$T = 0.9$
3	4	35.47	40.96	36.48	35.86
	8	28.55	32.61	29.71	28.33
	16	20.91	24.21	21.84	19.58
4	4	34.44	40.98	35.50	35.10
	8	27.63	32.34	28.88	27.40
	16	19.89	23.50	20.72	18.49
5	4	35.80	39.75	35.14	34.57
	8	27.23	30.18	28.19	26.66
	16	19.79	23.15	19.98	17.54

Table 6: Running times for image simulations (in seconds)

		AK-SVD	RND+AK-SVD	T-mindot + AK-SVD	
k	k_0	-	-	$T = 0.95$	$T = 0.9$
3	4	62.52	2.34	12.27	9.22
	8	79.96	4.5	15.76	11.84
	16	234.28	6.24	26.15	19.68
4	4	98.80	3.12	13.44	10.37
	8	143.27	7.91	17.73	13.51
	16	270.40	8.96	30.50	23.61
5	4	121.32	5.58	14.85	11.69
	8	164.91	9.01	20.15	15.28
	16	293.16	14.28	36.27	26.75

We describe two variants of the procedure, one in which all centroids are kept and one in which the centroids that do not cluster any other training samples around them are eliminated (cut). Each training vector has a fixed length of $n = 120$ and the dimensions of the training sets are described in Table 2. On average, T-mindot manages to reduce the size of the training sets by 70% and with 85% if the lone centroids are also cut. This large cut leads to a much faster running time for AK-SVD. Also, a further positive effect that this cut might have is the elimination of outliers from the data. When the cut is performed better performance is achieved when T is smaller since a lower T generally outputs fewer lone centroids, so less data items are cut. Tables 3 and 4 describe the results obtained for the running speed and the representation quality in the computed dictionaries.

From the tables it is clear that the speedup is significant for all the datasets while the error levels are very close. Using lower values for the parameter T induces a higher speedup at the expense of less accurate clustering and, in this case, higher representation errors. Also, further removal of the lone clusters reduces the running time even more while the increase to the error is insignificant.

Another set of simulations runs for training vectors extracted from a few 1024×1024 popular test images (lena.bmp, peppers.bmp, airplaneU2.bmp, man.bmp, airfield2.bmp, testpat.bmp). The idea is again to show that applying T-mindot before the AK-SVD procedure greatly reduces the running time while keeping the representation error low.

The training set consists of 98304 vectors of size 64 extracted from 8×8 non-overlapping patches from the test images mentioned earlier. The results of the simulations are presented in Tables 5 and 6. The parameter k establishes the dimension of the dictionary (for example, a k-overcomplete dictionary is of size $\boldsymbol{D} \in \mathbb{R}^{n\times kn}$), k_0 dictates the number of atoms that participate in the reconstruction (OMP step) of the training vectors. All simulations start from the same random initial dictionary. The run that considers also T-mindot is an exception from this rule. In this case, we take the initial dictionary to be composed from the centroids with the highest number of items grouped around them. All runs have the same number of AK-SVD iterations, which in this particular case is 20 since no significant decrease in the error happens after this limit. Using the T-mindot clustering procedure with parameter $T \in \{0.9, 0.95\}$ on such data reduces the size of the dataset by more than 90%.

The procedure called RND+AK-SVD runs by randomly selecting the same number of training vectors as the number of centroids selected by T-mindot with $T = 0.95$ and then applying the AK-SVD algorithm. We expect this procedure to run the fastest but we also expect it to run worst in terms of the representation error. Such a result validates the extra effort allocated in the T-mindot step.

From Table 5 it is clear that better results are obtained for T-mindot with $T = 0.9$ than with $T = 0.95$. One explanation for this result is the way in which the training procedure is initialized. The centroids computed with $T = 0.9$ seem to be more relevant than the ones computed with $T = 0.95$ and this leads to a lower representation error. Simulation show that further lowering T increases the representation error significantly, as expected.

All simulations prove conclusively that applying T-mindot greatly reduces the running time of the training algorithm (in this case, AK-SVD) while keeping the representation error close to the error obtained if the training algorithm would have been applied without the clustering first. On average, the speedup obtained using T-mindot before the K-SVD, as opposed to using K-SVD directly, is approximately 7 while the magnitude of the error is kept low, in some cases even lower than the full training algorithm can achieve. Of course, the speedup is heavily dependent on the dataset, but in the context of training dictionaries for sparse representation the data items are usually highly correlated.

All comparisons are conducted against AK-SVD since the running time of the K-SVD method is extremely high for large datasets like the ones used in this setting, while the representation errors are very similar. The fact that K-SVD computes the best direction by the exact SVD does not seem to have a crucial effect on the final result.

4. CONCLUSIONS

This paper presents an efficient clustering method, called T-mindot, which is applied on a dataset before the K-SVD algorithm is used to build a dictionary that describes a sparse linear model of the data. The main advantage of this procedure is that it enables the application of K-SVD on large datasets, where the speed of the training procedure would have been prohibitively long, while being able to keep the error at a relative close level to the error obtained by applying the K-SVD algorithm on the entire dataset.

5. REFERENCES

[1] A. M. Bruckstein, D. L. Donoho and M. Elad, From Sparse Solutions of Systems of Equations to Sparse Modeling of Signals and Images, SIAM Review, vol. 51, no. 1, 34–81, 2009.

[2] E. J. Candes and T. Tao, Decoding by linear programming, IEEE Trans. Inform. Theory, vol. 51, 4203–4215, 2004.

[3] D. L. Donoho, Compressed sensing, IEEE Transactions on Information Theory, Vol. 52, no. 4, 1289–1306, 2006.

[4] R. G. Baraniuk, E. J. Candes, M. Elad and Ma Yi, Applications of Sparse Representation and Compressive Sensing, Proceedings of the IEEE, vol. 98, 906–909, 2010.

[5] K. Huang and S. Aviyente, Sparse Representation for Signal Classification, Advances in Neural Information Processing Systems 19, 609–616, 2007.

[6] S. G. Mallat and Z. Zhang, Matching Pursuits With Time-Frequency Dictionaries, IEEE Trans. Signal Processing, vol. 41, no. 12, 1993.

[7] R. Rubinstein, M. Zibulevsky and M. Elad, Efficient Implementation of the K-SVD Algorithm using Batch Orthogonal Matching Pursuit, Technical Report - CS Technion, 2008.

[8] M. Aharon, M. Elad and A. Bruckstein, K-SVD: An Algorithm for Designing Overcomplete Dictionaries for Sparse Representation, IEEE Trans. Signal Processing, vol. 54, no. 11, 4311–4322, 2006.

[9] J. Mairal, M. Elad and G. Sapiro, Sparse Representation for Color Image Restoration, IEEE Transactions on Image Processing 17, 53-69, 2008.

[10] P. K. Agarwal, S. Har-Peled and K. R. Varadarajan, Geometric approximations via coresets, Combinatorial and Computational Geometry - MSRI Publications, vol. 52, 1-30, 2005.

[11] D. R. Wilson and T. R. Martinez, Reduction Techniques for Instance-Based Learning Algorithms, Machine Learning, vol. 38, no. 3, 257–286, 2000.

[12] L. Huawen, Z. Shichao, Z. Jianming, Z. Xiangfu and M. Yuchang, A New Classification Algorithm Using Mutual Nearest Neighbors, 9th International Conference on Grid and Cooperative Computing (GCC), 52–57, 2010.

[13] H. M. Jani and S. P. Lee, Applying machine learning using case-based reasoning and rule-based reasoning approaches to object-oriented application framework documentation, ICITA, vol. 1, 52–57, 2005.

[14] T. Kanungo, D. M. Mount, N. S. Netanyahu, C. D. Piatko, R. Silverman and A. Y. Wu, An efficient k-means clustering algorithm: Analysis and implementation, IEEE Trans. PAMI, 24, 881-892, 2002.

[15] L. R. Rabiner and B. H. Juang, Fundamentals of speech recognition, PTR Prentice Hall, 1993.

[16] M. Feigin, D. Feldman and N. A. Sochen, From High Definition Image to Low Space Optimization, SSVM, 459–470, 2011.

[17] ISMIR 2004 web page, http://ismir2004.ismir.net /genre_contest/index.htm .