Sparse Coding

Rishabh Mehrotra
2008B4A7533P
14th October, 2011.

Abstract

Sparse modelling calls for constructing efficient representations of data as a combination of a few typical patterns (atoms) learned from the data itself. Significant contributions to the theory and practice of learning such collections of atoms (usually called Dictionaries), and of representing the actual data in terms of them, have been made thereby leading to state-of-the-art results in many signal and image processing and data analysis tasks. **Sparse Coding** is the process of computing the representation coefficients **x** based on the given signal **y** and the given dictionary **D**. Exact determination of sparsest representations proves to be an NP-hard problem [1]. This report briefly describes some of the approaches in this area, ranging from greedy algorithms to l_1-optimization all the way to simultaneous learning of adaptive dictionaries and the corresponding representation vector.

1. Problem Statement:

Using a dictionary[1] matrix $\boldsymbol{D}\ (\in \boldsymbol{R}^{nXk})$ that contains **k** atoms, $\{\boldsymbol{d}_j\}_{j=1}^{\boldsymbol{k}}$ as its columns, a signal **y** $(\in \boldsymbol{R}^{n})$ can be represented as a sparse linear combination of these atoms, the solution of which may either be exact (**y=Dx**) or approximate (**y≈Dx**). The vector **x** $(\in \boldsymbol{R}^{k})$ expresses the representation coefficients of the signal y. The problem at hand is finding the sparsest representation, x which is the solution of either:

$$\mathbf{min}_x \|\boldsymbol{x}\|_{\boldsymbol{o}} \text{ subject to } \mathbf{y} = \mathbf{Dx} \qquad \textbf{(1)}$$

Or

$$\mathbf{min}_x \|\boldsymbol{x}\|_{\boldsymbol{o}} \text{ subject to} \|\boldsymbol{y} - \boldsymbol{Dx}\|_{\boldsymbol{2}} \leq \boldsymbol{\epsilon} \qquad \textbf{(2)}$$

where $\|.\|_o$ is the 1_o norm, counting the nonzero entries of a vector.

2. Solution Approaches

This section briefly describes few noted approaches to this problem, followed by detailed description of one of the prominent solution (K-SVD algorithm) in the next section.

2.1 Matching Pursuit

Mallat[2] proposed a greedy solution which successively approximates y with orthogonal projections on elements of D. The vector y ($\in H$,Hilbert Space) can be decomposed into

$$\boldsymbol{y} = < y, \boldsymbol{g}_{\boldsymbol{\gamma_0}} > \boldsymbol{g}_{\boldsymbol{\gamma_0}} + \boldsymbol{Ry}$$

[1] Note: The dictionary we refer to on this report is an Overcomplete Dictionary, with k>n.

Where Ry is the residual vector after approximating y in the direction of g_{γ_0}. g_{γ_0} being orthogonal to Ry, hence

$$\|\boldsymbol{y}\|^{2} = \left\|< y, \boldsymbol{g}_{\boldsymbol{\gamma}_{\mathbf{0}}} >\right\|^{2} + \|\boldsymbol{Ry}\|^{2}.$$

To minimize Ry we must choose $g_{\gamma_0} \in D$ such that $|< y, g_{\gamma_0} > |$ is maximum. In some cases it is only possible to find g_{γ_0} that is almost the best in the sense that

$$\left|< y, \boldsymbol{g}_{\boldsymbol{\gamma}_{\mathbf{0}}} >\right| \geq \boldsymbol{\alpha}\, \boldsymbol{sup}_{\boldsymbol{\gamma}\in\boldsymbol{\tau}} \left|< y, \boldsymbol{g}_{\boldsymbol{\gamma}_{\mathbf{0}}} >\right|$$

where α is an optimality factor that satisfies $\mathbf{0}\leq\boldsymbol{\alpha}\leq \mathbf{1}$.

A matching pursuit is an iterative algorithm that sub-decomposes the residue Ry by projecting it on a vector of D that matches Ry at its best, as was done for y. This procedure is repeated each time on the following residue that is obtained.

It has been shown that it performs better than DCT based coding for low bit rates in both efficiency of coding and quality of image. The main problem with Matching Pursuit is the computational complexity of the encoder. Improvements include the use of approximate dictionary representations and suboptimal ways of choosing the best match at each iteration (atom extraction).

2.2 Orthogonal Matching Pursuit (OMP)

In Pati[3] , the authors propose a refinement of the Matching Pursuit (MP) algorithm which improves convergence using an additional orthogonalization step.

As compared to MP, this method performs an additional computation of kth-order model for y,

$$\boldsymbol{y} = \sum_{\boldsymbol{n}=\mathbf{1}}^{\boldsymbol{k}} \boldsymbol{a}_{\boldsymbol{n}}^{\boldsymbol{k}} \boldsymbol{x}_{\boldsymbol{n}} + \boldsymbol{R}_{\boldsymbol{k}}\boldsymbol{y}\ ,$$

with $<\boldsymbol{R}_{\boldsymbol{k}}, \boldsymbol{x}_{\boldsymbol{n}}> = \mathbf{0}$, n=1...k.

Since the elements of D are not required to be orthogonal, to perform such an update, an auxillary model for dependence of x_{k+1} on x_k would be required, which is given by

$$\boldsymbol{x}_{\boldsymbol{k}+\mathbf{1}} = \sum_{\boldsymbol{n}=\mathbf{1}}^{\boldsymbol{k}} \boldsymbol{b}_{\boldsymbol{n}}^{\boldsymbol{k}} \boldsymbol{x}_{\boldsymbol{n}} + \boldsymbol{\gamma}_{\boldsymbol{k}}$$

with $<\boldsymbol{\gamma}_{\boldsymbol{k}}, \boldsymbol{x}_{\boldsymbol{n}}> = \mathbf{0}$ for n=1...k.

For a finite dictionary with N elements, OMP is guaranteed to converge to the projection onto the span of the dictionary elements in a maximum of N steps.

2.3 Basis Pursuit

Basis Pursuit (BP) is an optimization problem, not an algorithm. The authors in [4] have tried to model the Sparse Coding problem as a BP problem. In the BP approach, the sparsest solution in the L_1 sense is desired. BP is a mathematical optimization problem of the type:

$$\boldsymbol{min}_{\boldsymbol{x}}\ \frac{\mathbf{1}}{\mathbf{2}}\ \|\boldsymbol{Y} - \boldsymbol{DX}\|_{2}^{2} + \boldsymbol{\gamma}\|\boldsymbol{x}\|_{\mathbf{1}} \qquad - (\mathbf{3})$$

Where γ is a parameter that controls the trade-off between sparsity and reconstruction fidelity. BP requires the solution of a convex, non-quadratic optimization problem. BP can be seen as minimizing an objective that penalizes the reconstruction error using a linear basis set and the sparsity of the corresponding representation.

Any algorithm from the Linear Programming literature can be used to solve the BP optimization problem, hence finding the sparse representation. Both the interior points method and simplex are used in [4] to solve this problem.

2.4 Iterative Shrinkage-Thresholding Algorithm (ISTA)

In [5] authors have modelled the L_1 constrained sparse coding model presented as eq. (3) above as a general formulation :

$$\mathbf{Min\ \{\ F(x) \equiv f(x) + g(x) : x \in \boldsymbol{R^n}\ \}}$$

Where g(x) is a continuous non-convex function which is possibly non-smooth and f(x) is a convex smooth function with gradient which is Lipschitz continuous, ie, there exist a constant L(f) such that

$$||\boldsymbol{\nabla f(x)} - \boldsymbol{\nabla f(y)}|| \ \leq \boldsymbol{L(f)}\ ||\boldsymbol{x - y}||$$

The general step for ISTA is of the form:

$$\boldsymbol{x_{k+1} = prog_{t_x}(g)\big(x_k - \ t_k \nabla f(x_k)\big)} \qquad - \boldsymbol{(4)}$$

Where prog operator is defined by

$$\boldsymbol{prog_{t_x}(g) = \ argmin_u\ \{g(u) + \frac{1}{2}||u - x||^2\}}$$

When g(x)=0 prog is just the identity operator and ISTA is equivalent to the gradient method. Using the defined formulation, the sparse coding equation (3) cn be modelled as ISTA formulation and sparse representation x can be sought. It is to be noted that F(x_k) converges to the optimal value F_* with the rate of convergence equalling O($^1/_k$).

2.5 Fast Iterative Shrinkage-Thresholding Algorithm (FISTA)

In [6] authors present an improved version of ISTA which has a convergence rate of O($^1/_{k^2}$) as compared to O($^1/_k$). The main difference between FISTA and ISTA is that that the iterative-shrinkage step (4) is not employed on the previous point $\boldsymbol{x_{k-1}}$, but rather at the point $\boldsymbol{y_k}$, which uses a very specific linear combination of the previous two points { $\boldsymbol{x_{k-1}, x_k}$ }.

$$\boldsymbol{x_k = prog_{t_x}(g)\big(y_k - \ t_k \nabla f(x_k)\big)}$$

Readers are directed to [6] for the proof of the O($^1/_{k^2}$) convergence rate of this algorithm. Thus, FISTA preserves the computational simplicity of ISTA, but with a global rate of convergence which is proven to be significantly better, both theoretically and practically.

Note:

Apart from the Sparse Coding algorithms described above, some algorithms are also able to learn the set of basis functions (ie, elements of dictionary D). The learning procedure finds the B matrix that minimizes the same loss of eq. (3). The columns of D are constrained to have unit norm in order to prevent trivial solutions where the loss is minimized by scaling down the coefficients while scaling up the bases. Learning proceeds by alternating the optimization over Z to infer the representation for a given set of bases B, and the minimization over B for the given set of optimal Z found at the previous step.

The following two algorithms find the dictionary along with finding the sparse representations for the learnt dictionary.

2.6 K-SVD Algorithm

K-SVD is an iterative method that alternates between sparse coding of the examples based on the current dictionary and a process of updating the dictionary atoms to better fit the data. The update of the dictionary columns is combined with an update of the sparse representations, thereby accelerating convergence. The K-SVD algorithm is flexible and can work with any pursuit method (e.g., basis pursuit, FOCUSS, or matching pursuit).

Detailed description of the K-SVD algorithm is given in Section 3.

2.7 Predictive Sparse Decomposition

I order to make inference efficient, the authors[7] train a non-linear regressor that maps a input patches Y to sparse representations X. The following non-linear mapping is considered:

$$\mathbf{F(Y;\, G,W,D) = G\ tanh(WY+D)}$$

Where W is the filter matrix, D is the dictionary and G is the diagonal matrix of gain coefficients allowing the outputs of F to compensate for the scaling of input Y. Let P_f denote the parameters leaned in this predictor, P_f= {G,W,D}. The goal of the algorithm is to make the prediction of the regressor F(Y;P_f) as close as possible to the optimal solution of the representation X. The resulting loss function can be framed (based on eq. (3) defined in 2.3):

$$\mathbf{L(Y,Z;B,} \boldsymbol{P_f}\mathbf{)} = \|\boldsymbol{Y}-\boldsymbol{DX}\|_2^2 + \boldsymbol{\gamma}\|\boldsymbol{X}\|_1 + \boldsymbol{\alpha}\left\|\boldsymbol{X}-\boldsymbol{F}(\boldsymbol{Y};\boldsymbol{P_f})\right\|_2^2 \quad \textbf{-(5)}$$

Minimizing this loss with respect to X produces a representation that simultaneously reconstructs the patch, is sparse, and is not too different from the predicted representation.

Learning the parameters P_f proceeds by an on-line block coordinate gradient descent algorithm. Once the parameters have been learnt, inference can be done by Optimal Inference consisting of setting the representation to

$$\boldsymbol{X^{*}=argmin_X\, L}$$

by running an iterative gradient descent algorithm.

3. K-SVD Algorithm: Detailed Description

Given a set of examples Y = [y_1 y_2 ... y_n], the goal of the K-SVD [8] is to find a dictionary D and a sparse matrix X which minimize the representation error,

$$\boldsymbol{argmin_{D,X}}\ \|\boldsymbol{Y}-\boldsymbol{Dx}\|_F^2 \quad \text{subject to } \|\boldsymbol{x_i}\|_0^0 \le \boldsymbol{T}\ \ \forall_i$$

where $\boldsymbol{x}$ represent the columns of X, and the L_0 sparsity measure; $\|.\|_0^0$ counts the number of non-zeros in the representation.

The K-SVD algorithm alternates between two phases:

- Sparse Coding Phase
- Dictionary Update Phase

The sparse-coding is performed for each signal individually using any standard technique. The main contribution of the K-SVD is that the dictionary update, rather than using a matrix inversion, is performed atom-by-atom in a simple and efficient process.

Let us first consider the sparse coding stage, where we assume that is fixed, and consider the above optimization problem as a search for sparse representations with coefficients summarized in the matrix . The penalty term can be rewritten as:

$$\|\boldsymbol{Y}-\boldsymbol{DX}\|_F^2 \;=\; \sum_{i=1}^{N}\|\boldsymbol{Y}-\boldsymbol{D}\boldsymbol{x}_i\|_2^2 \qquad \textbf{-(6)}$$

The problem posed in (6) above can be decoupled to N distinct problems of the form:

$$\boldsymbol{min}_{x_i}\|\boldsymbol{y}_i-\boldsymbol{D}\boldsymbol{x}_i\|_2^2 \quad \text{subject to } \|\boldsymbol{x}_i\|_0 \;\le \boldsymbol{T}_0 \quad \boldsymbol{for\ i = 1, 2, \ldots.., N}$$

This problem is adequately addressed by the pursuit algorithms discussed in Section 2 above, and we have seen that if is small enough, their solution is a good approximation to the ideal one that is numerically infeasible to compute.

We now turn to the second, and slightly more involved, process of updating the dictionary D together with the nonzero coefficients X. Assume that both X and D are fixed and we put in question only one column in the dictionary d_k and the coefficients that correspond to it, the k-th row in X, denoted as x_T^k (this is not the vector which is the k-th column in X). Returning to the objective function eq. (6), the penalty term can be rewritten as:

$$\begin{aligned}\|\boldsymbol{Y}-\boldsymbol{Dx}\|_F^2 \;&=\; \Big\|\boldsymbol{Y}-\textstyle\sum_{j=1}^{K}\boldsymbol{d}_j\boldsymbol{x}_T^j\Big\|_F^2\\ &=\; \Big\|\big(\boldsymbol{Y}-\textstyle\sum_{j\neq k}\boldsymbol{d}_j\boldsymbol{X}_T^j\big)-\;\boldsymbol{d}_k\boldsymbol{x}_T^k\Big\|_F^2\\ &= \Big\|\boldsymbol{E}_k-\boldsymbol{d}_k\boldsymbol{x}_T^k\Big\|_F^2 \qquad\qquad -(\mathbf{7})\end{aligned}$$

We have decomposed the multiplication DX to the sum of k rank-1 matrices. Among those, k-1 terms are assumed fixed, and one—the kth —remains in question. The matrix E_k stands for the error for all the N examples when the k-th atom is removed.

Here, it would be tempting to suggest the use of the SVD to find alternative d_k and x_T^k . The SVD finds the closest rank-1 matrix (in Frobenius norm) that approximates E_k, and this will effectively minimize the error. However, such a step will be a mistake, because the new vector x_T^k is very likely to be filled, since in such an update of we do not enforce the sparsity constraint.

A remedy to the above problem, however, is simple and also quite intuitive. Defining ω_k as the group of indices pointing to the examples $\{\boldsymbol{y}_i\}$that use the atom d_k**,** ie those where x_T^k is zero.

$$\boldsymbol{\omega}_k = \{\boldsymbol{i} :\; \mathbf{1} \le \boldsymbol{i} \le \boldsymbol{K}, \boldsymbol{X}_T^k(\boldsymbol{i}) \neq \mathbf{0}\}$$

$\Omega_k$is defined as the matrix of size $N \times |\omega_k|$ with ones on the $(\omega_k(i), i)th$ position and zeroes elsewhere.

When multiplying $X_R^k = X_T^k \Omega_k$, this shrinks the row vector x_T^k by discarding of the zero entries, resulting with the row vector x_R^k of length $|\omega_k|$.
Thus the equation (7) becomes:

$$\left\| \boldsymbol{E_k \Omega_k - d_k X_T^k \Omega_k} \right\|_{\boldsymbol{F}}^{\boldsymbol{2}} = \left\| \boldsymbol{E_k^R - d_k X_R^k} \right\|_{\boldsymbol{F}}^{\boldsymbol{2}}$$

and SVD can be used to find the final solution. The K-SVD algorithm takes its name from the Singular- Value-Decomposition (SVD) process that forms the core of the atom update step, and which is repeated *K* times, as the number of atoms.

The authors have shown that the dictionary found by the -SVD performs well for both synthetic and real images in applications such as filling in missing pixels and compression and outperforms alternatives such as the non-decimated Haar and overcomplete or unitary DCT. K-SVD has been successfully applied to learn sparse representation for Sentiment Classification tasks as well. Refer [9][10] for details.

References

1- G. Davis, S. Mallat, and M. Avellaneda, “Adaptive greedy approximations,” *J. Construct. Approx.*, vol. 13, pp. 57–98, 1997.

2- S. G. Mallat and Z. Zhang, Matching Pursuits with Time-Frequency Dictionaries, IEEE Transactions on Signal Processing, December 1993, pp. 3397-3415.

3- Orthogonal Matching Pursuit- Recursive Function Approximation with Applications to Wavelet Decomposition, 27th Annual Conference on Signal Systems, Nov 1-3, 1993.

4- S.S. Chen, D.L. Donoho, and M.A. Saunders. Atomic decomposition by basis pursuit. SIAM Review, 43(1):129– 159, 2001.

5- I. Daubechies, M. Defrise, and C. De Mol, “An iterative thresholding algorithm for linear inverse problems with a sparsity constraint,” Comm. Pure Appl. Math., vol. 57, no. 11, pp. 1413–1457, 2004.

6- A. Beck and M. Teboulle. A fast iterative shrinkage-thresholding algorithm for linear inverse problems. 2009. ICASSP (2009).

7- K. Kavukcuoglu, M. Ranzato, and Y. LeCun.(2010) Learning Fast Approximations of Sparse Coding. In proceedings of 27th International Conference of Machine Learning, 2010.

8- M. Aharon, M. Elad, and A. M. Bruckstein, (2006) “The K-SVD: An algorithm for designing of overcomplete dictionaries for sparse representations,” IEEE Trans. Image Process., vol. 54, no. 11, pp. 4311–4322, Nov. 2006.

9- R Mehrotra, SA Haider, AS Mandal (2011). “ Adaptive Dictionary Learning for Sentiment Classification & Domain Adaptation” In Proceedings of 16th Conference on Technologies and Applications of Artificial Intelligence, 2011, Taiwan.

10- J. Mairal, F. Bach, J. Ponce, G. Sapiro, and A. Zisserman. Discriminative learned dictionaries for local image analysis. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2008a.