Parameterized Deformation Sparse Coding via Tree-Structured Parameter Search

Brandon Burdge, Kenneth Kreutz-Delgado
Dept. of Electrical and Computer Engineering
University of California San Diego
bburdge@ucsd.edu, kreutz@ece.ucsd.edu

Joseph Murray
jfmurray@jfmurray.org

***Abstract*—Representing transformation invariances in data is known to be valuable in many domains. We consider a method by which prior knowledge about the structure of such invariances can be exploited using a novel algorithm for sparse coding across a learned dictionary of atoms combined with a parameterized deformation function that captures invariant structure. We demonstrate the value of this on both reconstructing signals, as well as improved unsupervised grouping based on invariant sparse representations.**

I. Introduction

When observing the world, or a limited subset of the world, observations of phenomena often have the property that a very rich sensory input can be decomposed into a small set of conceptual features. As humans we regularly perform this task, when looking at the street outside the window, we see an enormous array of colors, textures, and shapes. However, we can efficiently reduce this extremely high-bandwidth observation into a very compressed description, enabling us to communicate to a non-observing party that, "A man is getting into his car, and a dog is walking across the street."

Sparse coding has taken many roles in different applications, with a heavy emphasis in the domains of signal compression, and pattern recognition. The different purposes for sparse coding produce different interpretations on the components in the model. For example, if the purpose is image compression, then dictionary elements are meant to be highly efficient at encoding pixel data, but they are not assumed to have any particular meaning.

Alternatively, in some domains the goal is to produce sparse representations that in some way grasp the essence of the sources in the world. If one wanted to use sparse coding as part of an object classification scheme, then the goal would be to have different dictionary atoms represent some concept of objects in the world, ideally in such a manner that knowing the atoms used in an observation provided direct knowledge of the objects. Of course, the world is rarely so nice as to make this easy, when a car is observed it can be at any angle, under many different lighting conditions, it can be colored or shaped in an enormous variety of ways, the combination of particular choices for these conditions combine to make a unique observation. However, the state of the observation conditions in no way change the essence of the car.

As such, in order for a sparse coding algorithm to find invariant concepts from observations of the world, the models need to make some account for the observational variations, and should attempt to dissociate those instantiation variations from the invariant concepts.

[0]With thanks to the National Science Foundation for partial support. NSF Grant CCF-0830612

II. Sparse Coding for Invariant Representation

In the majority of literature regarding the sparse coding problem, the forward generative model is assumed to take an underdetermined linear form:

$$y = Ax + \nu, \tag{1}$$

where y is a densely populated observation, A is an overcomplete dictionary or frame of generating atoms, x is a set of loadings onto those atoms, taken to be sparse in the sense that no more than a small number of elements of x are non-zero, and ν is a noise term for which differing assumptions are made depending on the sparse coding model [1].

However, this model is limited when trying to deal with invariants that arise in the world. For instance, imagine that dictionary atoms represented items found in natural images, e.g. vector a_k represented a generic model of a car. Unfortunately in the case that the car is distorted (as compared to the dictionary example) in an observed image - by planar, or 3D rotation, scale, or 2D translation, etc. - the model may fail to recognize the item since the model does not, and cannot, account for any of these deformations of dictionary atoms.

Ways to deal with this problem have been explored. One is to simply expand the dictionary to large size and have within it elements that represent objects under the commonly seen deformations [8]. An obvious drawback of this method is computational complexity, which for most current sparse coding algorithms grows rapidly with the size of the dictionary. Further, in the case that the dictionary is not a priori known, the problem of learning a very large dictionary with the desired structure becomes quite difficult.

On the other end of the spectrum, dictionaries where atoms are represented as a set of parameters for a functional form have been used[4], [3], [5]. This approach attacks the problem of deformation, and produces invariant dictionaries. However, given continuous parameters these are continuous dictionaries with difficulties in producing computationally tractable sparse coding algorithms. The genetic algorithm used in [3] produces suboptimal results that are not generally repeatable which hinders the usefulness of this approach for producing invariant concepts.

III. Parameterized Deformation Sparse Coding

We develop a approach that lies between these two extremes and develop algorithms derived from a generalization of the linear model (1). The model can be described formally as,

$$y = \sum_{i,j,k} \alpha_{ijk} f_j(a_i, \psi_k) + \nu, \tag{2}$$

and we call the resulting framework Parameterized Deformation Sparse Coding. In PDSC, we assume that each observation is formed from a set of atoms a_i distorted by a set of possible deforming functions f_j each parameterized over some fields (continuous or discrete) Θ_j, linearly combined and with additive noise.

The choosing of a set of possible distortion functions, f_j, is very much a function of the application domain. Careful choice of deformation functions allow us to input prior knowledge of the variability of observation of invariant aspects of the world into model (2), increasing the likelihood that we learn fundamental sources in the world.

The completely general model (2) is a starting point, from which intelligent model choices and simplifications are made to conform to practical limitations on the complexity of inverse solution algorithms. To provide empirical results on practical problems, a limited model is considered for the remainder of this paper. We restrict our model to a single, known, distortion function, with known inverse, and restrict the function to have a one dimensional parameter that takes a finite number of discrete values. We also assume that for a particular value of ψ the function becomes the identity function. Continuing with our desire to learn fundamental sources with practical constraints on complexity, we have a small number of atoms, and describe an algorithm for iteratively learning those atoms to fit observed data. This leaves us with a simplified model in the form of

$$y = \sum_{i,j} \alpha_{ij} f(a_i, \psi_j) + \nu. \tag{3}$$

While less general than (2), this model is still far more flexible than the straight-forward linear generative model (1).

Given (3) we define an optimization problem to describe the sparse coding problem for a single observation y,

$$\arg\min_{\theta} ||y - \sum_{i,j} \alpha_{ij} f(a_i, \theta_j)|| \tag{4}$$
$$\text{subject to } \sum_{i,j} \mathrm{I}(\alpha_{ij}) \leq T.$$

Where $\mathrm{I}(x)$ is the binary indicator function. This defines a combinatorially large search over i, j to find an optimal solution, hence approximate methods must be found to approach the problem. This is the same issue faced in any sparse coding environment and we can extend known algorithms to find approximate solutions. We choose in particular to base our approach on the Orthogonal Matching Pursuit algorithm [6].

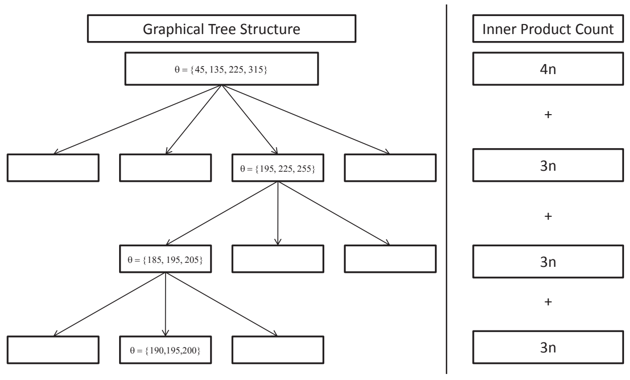

Fig. 1. An example tree. This tree searches over angles of rotation for sparse coding 2D images.

A. Tree-structured Parameter Search

To extend the OMP algorithm to the parameterized deformation model, it is necessary to perform the sequential search over not only atoms, but also parameter values. Considering that OMP already has a computation complexity as $\mathcal{O}(Tmn)$ [1] then we see that naively searching over the parameter would result in complexity $\mathcal{O}(KTmn)$, where K is the number of parameter values. Therefore, to mitigate the computational cost, we implement the parameter search on a tree structure. For the tree-structure to give valid results, we must assume that f is smooth on ψ. Figure 1 shows an example structure, this one used for a search over plane image rotations on five degree angle increments.

While equation (4) shows the general form of the reconstruction using the set of $\{\alpha_{ij}\}$'s stored as a matrix, since we are working specifically with sparse representations, most entries in this matrix will be zero and we suggest a compressed storage representation. We define a vector of indices into the set of atoms, I. For each element in I, there is a corresponding deformation parameter value, stored in the vector θ, and for each atom/deformation pair there is a corresponding loading constant stored in the vector $\mathcal{A}$. The vector triple $(\mathcal{A}, I, \theta)$ then stores all the necessary information to reconstruct a single data example, as

$$\hat{y} = \sum_{j=1}^{k} \mathcal{A}[j] f(a_{I[j]}, \theta[j]). \tag{5}$$

As such, the sparse inverse problem requires finding an index vector I, loading vector $\mathcal{A}$, and parameter vector θ, which produces the $\hat{y}$ by model (5) that best represents the observed y. Algorithm 1 shows the steps for learning this triple.

Algorithm 1 Tree-Structured Parameter Search for PDSC

```
Require: $k = 1, r_0 = y, \hat{y}_0 = 0, I_0 = [\emptyset], \mathcal{A}_0 = [\emptyset], \theta_0 = [\emptyset], S_0 = [\emptyset]$
  Tree Structure: for layers $l = 1, ..., L$ with $B_l$ branches, $\Phi_l = \{\phi_1, ..., \phi_{B_l}\}$

  while $k \leq T$ and $||y - \hat{y}|| \geq \epsilon$ do

    for $l = 1, ..., L$ do
      $b_l = \arg\max_c | < r_{k-1}, f(a_j, \phi_c) > | \forall j$
      Follow branch $b_l$
    end for

    $I_k = [I_{k-1}, \arg\max_j | < r_{k-1}, f(a_j, \phi_{b_L}) > |]$
    $\theta_k = [\theta_{k-1}, \phi_{b_L}]$
    $S_k = [S_{k-1}, f(a_{I[k]}, \theta[k])]$
    $\mathcal{A}_k = \arg\min_x ||S_k x - y||_2$
    $\hat{y} = \sum_{j=1}^{k} \mathcal{A}[j] f(a_{I[j]}, \theta[j])$
    $r_k = y - \hat{y}$
    $k = k + 1$

  end while
```

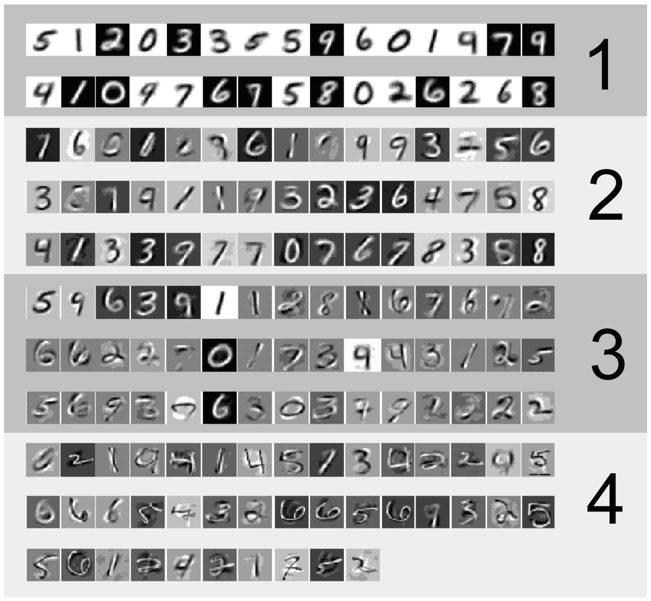

Fig. 2. A 4 Stage Hierarchical Dictionary for MNIST Digits.

IV. Hierarchical Dictionary Construction

To find a dictionary learning technique to well match our PDSC model, we view sparse coding as a generalization of vector quantization. In particular we consider the Multi-Resolution Vector Quantization algorithm [2] which produces quantization vectors with a particular truncation property. For given truncation points, the MRVQ algorithm produces quantization vectors which produce an optimal reconstruction for that level of compression. We can extend this property to the sparse coding domain by considering a dictionary constructed such that sparse coding to a certain number of non-zero elements leads to an approximately optimal reconstruction given that sparsity level.

Such a dictionary can be constructed in a hierarchical fashion, by performing successive dictionary learning passes and concatenating the results. Thus the first pass is used to learn a set of atoms that best represent the observed data given one non-zero element. The second pass then learns a set of atoms that code with two non-zero elements, given the first element is chosen from the atoms learned in the first pass. We can continue this for as many passes as we wish. This dictionary will be well suited to the sequential pursuit algorithm used in PDSC. As a bonus, we gain an efficiency improvement during sparse coding with this dictionary, at each sequential pass we only need consider the elements appropriate for that sparsity level, which greatly constrains our selection process.

This enforces an interesting structure on the dictionary. By attempting to encode as much of the observed data as possible in the first pass, we produce a set of atoms that represent the most average structure of typically observed objects in the domain. Further passes contain information about commonly observed variations from these averages. This forms an expectation-based deconstruction of an observation, where expected structure is removed first and variations from the expectation are coded separately. To best adapt this to PDSC, we train this dictionary using a set of observations that are registered to have a fixed reference point in the deformation parameter space.

To perform the atom learning, we extend the Approximate K-SVD algorithm [7]. The algorithm, applied to basic model (1) is as follows: A sparse coding x is given for each observation y, learned from a current estimate of the dictionary. For each dictionary element a_k the indices of all sparse code vectors with non-zero loading on that element is collected into $C(k)$. We form Y_C, a matrix of observations matching indices $C(k)$, taken as columns, and the reconstructions of those elements without using element k; $R_C = \sum_{j \neq k} a_j X_{j,C}$. Then we update a_j and $X_{k,C}$ as

$$\begin{aligned} a_k &\leftarrow \frac{(Y_C - R_C) X_{k,C}}{||(Y_C - R_C) X_{k,C}||} \\ X_{k,C} &\leftarrow (Y_C - R_C)^T a_k \end{aligned} \tag{6}$$

Multiple iterations of these updates could be applied between sparse coding steps, but it is claimed in [7] that a single iteration is most often sufficient. The necessary extension for our Hierarchical Dictionary Construction (HDC) is to restrict the reconstruction term $R_C = \sum_{j<k} a_j X_{j,C}$. So that at each stage the atom attempts to fit as much of the residual as not explained by previous hierarchical passes.

We make one further note on the HDC technique. As the number of stages increases, the residual content of observations being coded becomes increasingly uncorrelated and noise-like because the gross structure of the object has been

TABLE I
AVERAGE AGREEMENT OF ATOM INDICES OVER RANDOM ROTATIONS

Angle Range	Stage 1	Stage 2	Stage 3	Stage 4
0,355	0.75	0.33	0.59	0.66
-25, 25	0.76	0.34	0.61	0.68

removed. These residuals become difficult to sparse code since they have little similarity with each other. In some cases this point will have sufficiently small reconstruction error, and the dictionary learning simply stops. However, it may be that the lack of improvement in reconstruction error occurs before the desired level of accuracy is attained. In such case, we suggest that rather than invest significant training time and dictionary size to attempt to sparse code these noisy residuals, that the hierarchical dictionary algorithm be stopped, and a section of dictionary with other properties used. One possibility is to densely code these residuals on a fixed complete basis, such as an appropriate Fourier or wavelet dictionary. Another possibility is to sparse code the residuals on a fixed size dictionary using many terms, this can be done either with PDSC, or to save computation time with simpler flat methods.

V. EXPERIMENT

To demonstrate the utility of the PDSC and HDC framework we perform two experiments using the MNIST handwritten digits dataset, these are 28x28, greyscale images, with the digits centered. The deformation chosen for testing is a planar rotation of the digits. To perform these tests a hierarchical dictionary was trained using 60000 training images from the MNIST set, roughly uniformly distributed across digits. Four hierarchical sections were trained each containing 40 atoms for a total of 160 atoms, and the residuals were blocked into 7x7 blocks and coded with 400 learned atoms. These residual codes were used to provide accurate reconstruction, but were ignored in experimental processing. The test set consisted of 2000 new examples, 200 for each digit. The first

TABLE II
UNSUPERVISED CLASSIFICATION FALSE NEGATIVE RATES

Flat	0	1	2	3	4	5	6	7	8	9
30	0.04	0.01	0.15	0.17	0.52	0.31	0.09	0.15	0.30	0.37
60	0.03	0.02	0.14	0.19	0.38	0.34	0.09	0.17	0.32	0.27
120	0.04	0.01	0.12	0.19	0.36	0.20	0.09	0.17	0.28	0.30
Rotated										
30	0.06	0.13	0.19	0.31	0.38	0.27	0.24	0.26	0.28	0.67
60	0.06	0.03	0.16	0.24	0.45	0.28	0.22	0.24	0.30	0.33
120	0.03	0.02	0.18	0.19	0.40	0.27	0.17	0.25	0.28	0.40

experiment focuses on the ability for PDSC to find appropriate deformation parameters for observations. To test this ability, each observation in the test set was coded using PDSC, first with no deformation, and then 4 additional times, each time with the digit rotated to a random angle on either [0,355] or [-25,25] in discrete 5 degree increments. The results in TABLE I show the rate of agreement between chosen atom indices for each test observation.

We note that there is high agreement in the important first element, this is an important result since the first element contains much of the information about the class of digit. Further we comment that the significant drop in stability for the second element seems to be caused by the sudden loss of structure in the image caused by the removal of the major first element, though no detailed analysis has yet been completed.

The second experiment is to examine the ability for the PDSC algorithm to retain useful information when applied to deformed observations. To demonstrate this we performed an unsupervised learning task. First the undeformed test set was coded using PDSC with the learned hierarchical dictionary, the output from this process was then clustered using a simple K-means algorithm with 3 different values for cluster number, and 20 different initial states. Clusters were then labeled to a particular digit by majority vote. The false negative rates for each digit were obtained, and the best clustering chosen to have the lowest average over all 10 digits. This process was again repeated, however in the second case after the test images were rotated to a random angle on [0,355] in 5 degree increments.

The exact performance of the resulting *unsupervised* classification trails modern supervised systems by a wide margin, as expected, since performance of unsupervised systems in general will fall significantly below supervised on nearly all tasks. What is of importance in these results is the ability of PDSC to extract important information, in an usupervised manner, even after deformation. We note that the false negative rates only rise a few percent in most cases for the rotated clusterings. A major exception being 6 and 9, which have a natural rotational ambiguity.

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, 51 (2009), pp. 34–81.

[2] M. EFFROS AND D. DUGATKIN, *Multiresolution vector quantization*, IEEE Transactions on Information Theory, 50 (2004).

[3] R. I VENTURA, P. VANDERGHEYNST, AND P. FROSSARD, *Evolutionary Multiresolution Matching Pursuit and its Relations with the Human Visual System*, in Proc. EUSIPCO, vol. 2, Citeseer, 2002, pp. 395–398.

[4] F. MENDELS, P. VANDERGHEYNST, AND J. THIRAN, *Rotation and Scale Invariant Shape Representation and Recognition using Matching Pursuit*, in International Conference on Pattern Recognition, vol. 16, 2002, pp. 326–329.

[5] F. MENDELS, P. VANDERGHEYNST, AND J. THIRAN, *Matching Pursuit-based Shape Representation and Recognition using Scale-space*, International Journal of Imaging Systems and Technology, 16 (2006), pp. 162–180.

[6] Y. PATI, R. REZAIIFAR, AND P. KRISHNAPRASAD, *Orthogonal Matching Pursuit: Recursive Function Approximation with Applications to Wavelet Decomposition*, in Signals, Systems and Computers, 1993. 1993 Conference Record of The Twenty-Seventh Asilomar Conference on, 1993, pp. 40–44.

[7] R. RUBINSTEIN, M. ZIBULEVSKY, AND M. ELAD, *Efficient Implementation of the K-SVD Algorithm using Batch Orthogonal Matching Pursuit*, tech. report, Israel Institute of Technology, Computer Science Dept., 2008.

[8] J. WRIGHT, A. YANG, A. GANESH, S. SASTRY, AND Y. MA, *Robust Face Recognition via Sparse Representation*, IEEE Transactions on Pattern Analysis and Machine Intelligence, (2008), pp. 210–227.