ROTATION INVARIANT SPARSE CODING AND PCA

NATHAN PFLUEGER, RYAN TIMMONS

Abstract. We attempt to encode an image in a fashion that is only weakly dependent on rotation of objects within the image, as an expansion of Roger Grosse's work on Translation Invariant sparse coding.

Our approach is to specify only a small set of basis images, from which some reasonably large number rotated bases are calculated. The image is trained to this set of rotated bases, so that ultimately the spare code representation is given in terms of rotations of a small set of vectors.

We develop an image representation scheme and several algorithms suited to this problem, particularly gradient descent methods for sparse coding and an investigation of rotation invariant Principal Components Analysis. Our experimental results suggest that PCA is significantly more fruitful, unless better algorithms for the sparse coding side can be developed in the future.

1. Motivation

We are motivated ultimately by image recognition and classification. All existing algorithms are overly sensitive to rotations, and often translations, of the image, which can drastically alter the way it is encoded and thus the way an algorithm might classify it. This research is aimed at producing an image coding method in which even rotations of individual objects in the image will not drastically alter the image representation. As an example, an image might contain several objects, each of which can rotate freely. We develop a system in which all possible rotations of these objects would be coded nearly identically (differing only in the index of some coefficients), in a manner that is not predisposed to any orientation of the overall image, or indeed the orientation of any subportion of it.

2. Rotation Formulation and Image Construction

The base for both our Sparse Coding and PCA attempts is a robust formalism for representing the image in terms of rotated bases.

The building blocks of our image representation are circular image patches. A parameter n, the radius of the patches, is given, and each patch is represented in two forms: as a $2n+1 \times 2n+1$ matrix, where those entries of distance greater than n from the center are ignored, and as a column vector listing all the elements within distance n of the center in an arbitrary order.

The vector representation allows rotations to be defined simply as linear transformations of vectors. In particular, before construction begins, we define the number of rotations r, and rotation matrices $T_1, \ldots, T_r$ are calculated to represent rotating a vector by $\frac{2\pi i}{r}$ radians. Note that while ideally these would form a group under multiplication, there is some distortion due to fitting images into pixels, so we generally compute each T_i individually rather than composing them with each other.

Given a basis set B of b vectors, a radius n, and a number of rotations r, we use the following procedure to construct the image from these bases. The image specification is given by a $b \times r \times W \times H$ matrix S, where W, H are the dimensions of the image; S gives the weights of each possible rotation of each basis, centered on each point in the image.

We first construct the intermediate matrix Z, which is $W \times H \times m$:

Date: 14 December 2006.

$$Z_{x,y,:} = \sum_{j=1}^{r} T_{:,:,j} B S_{:,j,x,y}. \tag{1}$$

In words, Z gives the content of the image patches centered at a given point, in vector form. Each value of j constitutes a different rotation of the basis vectors, and these are summed to gain the overall patch centered at the point. Next Z must be expanded into Y. The value at each point (x, y) is the sum of the contributions from all nearby pixels. In particular, for each pixel within n units of (x, y), there is a unique value of i such that the point (x, y) corresponds to the i^{th} entry in the vector for the patch centered at the nearby point. Thus Y is given as follows:

$$Y_{x,y} = \sum_{i=1}^{m} Z_{x+u_{xy}(i),y+v_{xy}(i)}. \tag{2}$$

In the above summation, the functions u, v simply transform from a point x, y to a new point u, v, according to that point in the image where component i of the basis patches centered on (u, v) contribute to the shading at (x, y).

Above, we interpret the indices of Z to be numbers in modulo W and H respectively, so that bases in fact wrap around the edge of the image. This is in the style of [2], and achieves a pleasing symmetry in the representation, though the problem could certainly be formulated otherwise. Note we can also write this explicitly as the following horrific summation:

$$Y_{x,y} = \sum_{i=1}^{m} \sum_{j=1}^{r} \sum_{k=1}^{b} (T_{:,:,j} B)_{i,k} S_{k,j,u_{xy}(i),v_{xy}(i)}. \tag{3}$$

The is the basic image construction model on which all of our procedures are built.

3. Sparse Coding

Denote the image constructed from S by Y. Then we pose the following sparse coding optimization problem, for some constant β:

$$\text{minimize}_{S,B} f_{obj} = ||X - Y||_2 + \beta||S||_1 \text{subject to} ||B_i||_2 \leq 1. \tag{4}$$

Note that this formulation is identical to that used in [3] and [2], though the method of constructing Y from S differs; it is as described in the previous section.

As in [3] and [2], the optimization problem is divided into two problems: image reconstruction and basis learning. We solve these two problems in alternation. Due to the complexity of dealing with the four-dimensional matrix S, the only algorithms we were able to make practical used some form of gradient descent, coupled with a modified version of the feature-sign algorithm in [3].

3.1. Algorithms.

The relevant gradients can be computed as follows. We must introduce some new notation: the functions u', v' are the reverse functions for u, v discussed above; $(u'_{zw}(i), v'_{zw}(i))$ converts from the coordinates of the center (z, w) of a basis patch to the point influenced by the i^{th} component of the basis vector centered at (x, y).

$$\frac{\partial}{\partial S_{c,d,z,w}} = -2 \sum_{i=1}^{m} (T_{:,:,d} B)_{i,c} (X - Y)_{u'_{zw}(i),v'_{zw}(i)}. \tag{5}$$

For implementation purposes, it is most efficient to precalculate two matrices for each value of i, $M_i = (T_{i,:,:} \cdot B)^T$ and $N_i = (X - Y)_{u',v'}$, noting that

$$(\nabla_S ||X-Y||_2)_{:,:,z,w} = -2\sum_{i=1}^{m} M_i \cdot (N_i)_{z,w}. \tag{6}$$

This is the equation used to calculated gradient in our algorithm; it must be calculated for each coordinate pair (z, w).

Similarly to the feature-sign algorithm in [3], we implicitly keep track of the desired sign of each coefficient in S, compute the gradient of f_{obj} assuming these signs, and then perform a line search in the direction of this gradient. This cannot be done in closed form since the gradient involves both the coefficients of $\nabla_S ||X-Y||_2$ and also $\nabla_S ||S||_1$. Thus we use the following algorithm to optimize (4) in terms of S.

(1) Initialize S = 0
(2) Compute the gradient G of $||X-Y||$, with respect to S
(3) Set to zero all elements of G_0 of magnitude less then β
(4) Guess the signs of all elements of S according to either their current sign, or the sign that they will become according to the gradient, and use this to compute $\nabla_S ||S||_1$ and thus $\nabla_S f_{obj}$.
(5) Compute in closed form the ideal α so that $S - \alpha\nabla_S f_{obj}$ minimizes the objective function, assuming that no signs will change.
(6) Using this value of α as an upper bound, find the true optimal α (allowing for sign changes) by successively increasing and decreasing α, and computing the resulting objective function.
(7) Once the previous step converges, return to step (2).

In essence, this algorithm guesses a descent direction which will certainly be locally true, and then using efficient guess-and-check (a much more efficient calculation that computing the gradient anew), performs a line search along this direction, before computing a new gradient and thus a new direction.

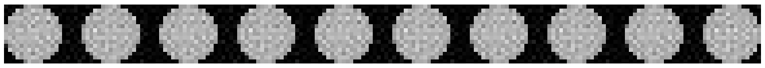
Likewise, the basis solving step is accomplished by gradient descent. Here we trade the troublesome $||S||_1$ error term in equation (4) (which is invariant with B) for the regularization condition $||B_i||_2 \leq 1$. The gradient can be computed, after some difficulty, as:

$$\nabla_B f_{obj} = -2\sum_{i=1}^{m} T(i,:,:) \cdot \left(\sum_{x,y} (X-Y)_{i,y} S^T_{u,v}\right).$$

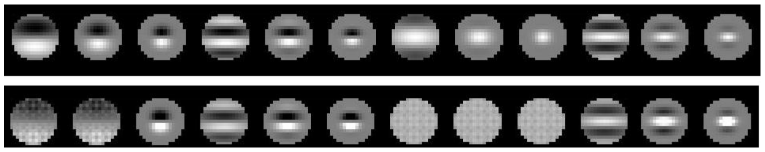
Here u, v are functions discussed above to transform from vector to matrix representational forms. Our algorithm successively computes the gradient, performs a small step in that direction, and then projects to satisfy the regularization term, much as in [2]. This algorithm is not terribly fast, but converges reasonably well in only 5 to 10 iterations.

3.2. **Experimental Results.**

We applied these two algorithms, in alternation, to images from the Caltech101 dataset ([1]). In particular, we tested on images from the categories "yin-yang," "soccer ball," and "water lily." In all cases, we found that our algorithms only tended to produce very simple sparse coding bases, not nearly as robust as those found in traditional efforts. More often than not, bases would train to gray sets such as the following.

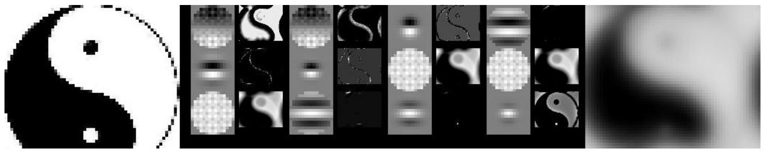


Bases such as these resulted from random initilialization as well as most hand-made bases that we attempted to use. A notable exception was the use of hand-defined Gabor filters. The original bases, and the result of the learning, are shown.



This resulted in those bases sufficiently suited to edge detection being spread out into pure edge detectors. This was further reinforced by the resulting basis activations, shown for a ying-yang image below. Note that the only bases being used for image coding are the edge detectors and solid white patches to give the overall brightness.

In this diagram, and in several in the future, the image on the left is the original image, the image on the right is the reconstructed. In between are shown all basis vectors, and the region on which they are used, regardless of rotation (dark is high negative activation, light is high positive activation; gray is zero).



In general, attempts to apply our sparse coding algorithms in their current form, for all values of β we attempted, simply created the blurry images shown above, though it is interesting that general edge detectors have a tendency to arise.

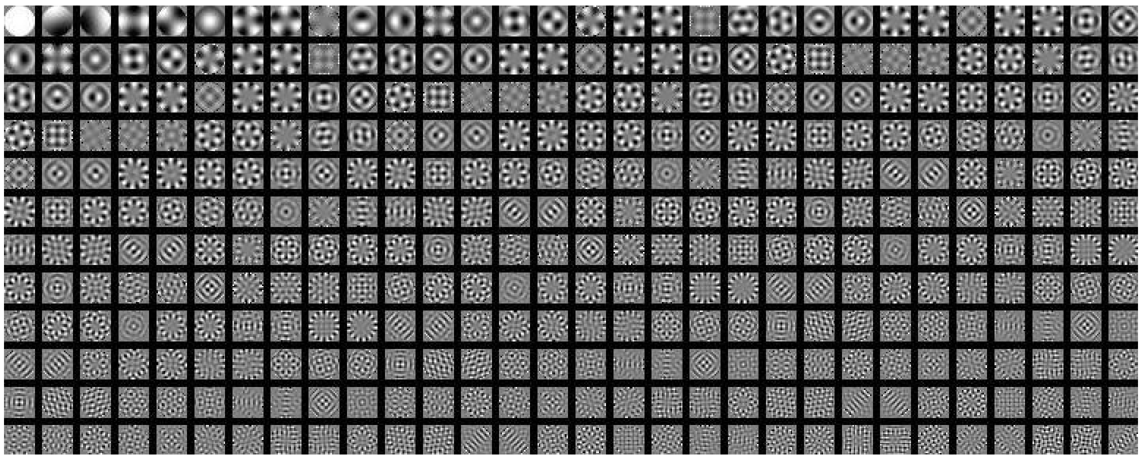
4. Principal Components Analysis

An unanticipated but quite fruitful alternate effort in our research was instead to learn rotation invariant Principal Components, rather than sparse bases. There are a number of ways to pose this problem. Our first attempt, and ultimately the only tractable option we were able to uncover, is simply to compute the principal components of all rotations of all patches in an initial image set. The net effect achieves significant rotation invariance; a particular patch being rotated will not affect the principal components found, since all possible rotations are given equal weight in finding these components.

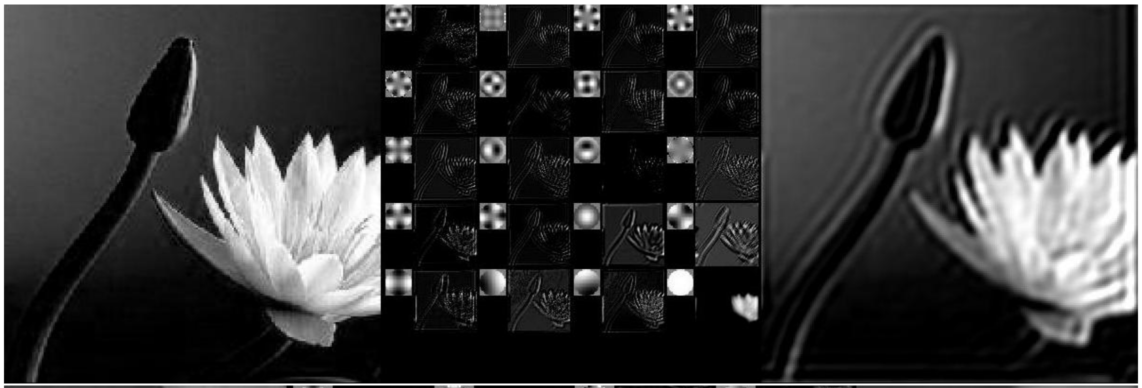
Algorithmically, we applied the PCA formulation from class to the set of all rotations of all patches from the images being trained. In particular, each $W \times H$ image generates $W \cdot H$ patches of radius n, each of which is rotated r times to produce a total of $W \cdot H \cdot r$ vectors for each image. The sum of the outer products of all these vectors is computed, and the rotation invariant principal components are the eigenvectors of this sum.

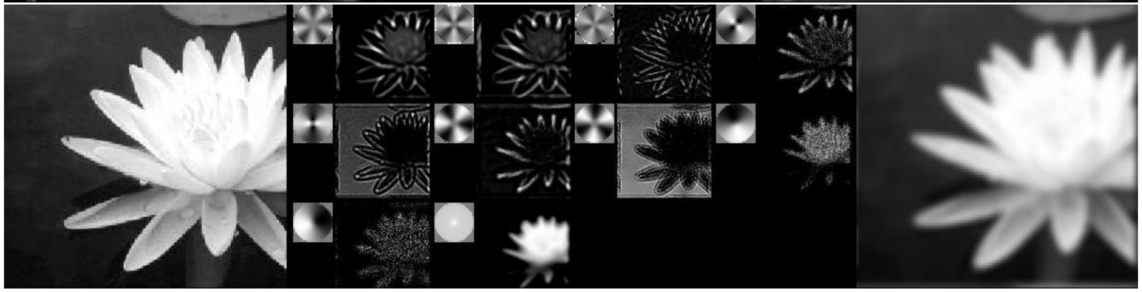
4.1. Results.

The resulting components learned are fascinating in their own right as curious visual patterns, and reveal interesting things about the structure of natural images. We ran trials on the "yin-yang," "soccer ball" and "water lily" categories of [1], and all produced qualitatively similar components, so we shall present only the result of training on the water lilies. Below is shown the full set of principal components learned on the water lily category, with patches having radius 10 pixels, and 36 total rotations allowed.



It is also of interest to observe the activation patterns of the top several eigenpatches on one particular image. Shown below are the encodings of two images from [1], which were encoded with our same algorithm from Sparse Coding (altough we set $\beta = 0$). As before, the images on the left are the originals, the images on the right are encoded, and in the center are shown each basis and the density of its activation, where all rotations are considered equally. Note the two example below come from different trials, so they use different basis sets.





We note first of all that, like in rotation invariant sparse coding, rotation invariant PCA also leads to patches easily recognizable as edge detectors, of varying complexity. Substantial further investigation might be possible into the utility of these more complex rotational edge detectors, and their applications to image description, classification, and recognition.

5. Future Work

First and foremost, it would be of interest for future research to refine our Sparse Coding algorithms until they can produce results comparable to existing procedures, even under the added duress of the computational difficulty. Both the results of successful PCA and Sparse Coding would likely be very robust if applied to object classification problems, having achieved a robust form of rotation invariance.

6. Acknowledgements

We would like to thank Honglak Lee and Roger Grosse for their invaluable advice and support throughout this project. We would also like to recognize Mark Linsey, who contributed to the early stages of this research but was unable to remain on the team.

References

[1] Fei-Fei, L., R. Fergus and P. Perona. Learning generative visual models from few training examples: an incremental Bayesian approach tested on 101 object categories. IEEE. CVPR 2004, Workshop on Generative-Model Based Vision. 2004

[2] Grosse, R. “Translation Invarient Sparse Coding.” Unpublished write-up, 2006.

[3] Lee, H., Alexis Battle, Rajat Raina and Andrew Ng. “Efficient sparse coding algorithms, end-stopping, and nCRF surround suppression.” Unpublished Draft, 2006.