Project Report

190050034-190050097

May 11, 2021

The project is aimed to study the paper :

Approximation and Compression using Sparse Orthogonal Transform
by
Osman G. Sezer, Onur G. Guleryuz and Yucel Altunbasak

which aims at constructing a **data driven Orthogonal Transform** which extends over the Karhunen-Loeve Transform which is known to be optimal on Gaussian Processes. For gaussian processes, this method reduces to the KLT, and performs siginificantly better on other processes.

KLT is equivalent to SOT on Gaussian Processes

The major theoretical point made in the paper is this. The definition of SOT is in terms of solution to a minimization problem given by :

$$SOT = argmin_G[E(min_\alpha\{||x - G\alpha||_2^2 + \lambda||\alpha||_0\})] \ni GG^T = G^TG = I$$

Proposition 1. *Let x be a zero mean random variable ($\mathbb{R}^n$) with covariance matrix Σ. Given any function W over orthonormal matrices ($\in \mathbb{R}^{n\times n}$) of the form*

$$W(G) = \sum_{i=1}^{n} f(g_i^T \Sigma g_i)$$

where f is a concave function. The matrix P of eigenvectors of Σ is a minimizer of W.

Proof. Let G be arbitrary. It is clear that $\Sigma = P\Lambda P^T$ for a diagonal matrix Λ. Let $R = P^TG$ be another orthogonal matrix. Then,

$$f(g_i^T\Sigma g_i) = f(g_i^T P\Lambda P^T g_i) = f(r_i^T \Lambda r_i) = f(\sum_{j=1}^{n} r_{i,j}^2 \lambda_{jj})$$

Since, f is concave and $\sum_{j=1}^{n} r_{i,j}^2 = 1$, using Jensen's inequality it is clear that

$$f(g_i^T\Sigma g_i) f(\sum_{j=1}^{n} r_{i,j}^2 \lambda_{jj}) \geq \sum_{j=1}^{n} r_{i,j}^2 f(\lambda_{jj}) = r_i^T Q r_i = g_i^T PQP^T g_i$$

where Q is a diagonal matrix with entries $Q_{ii} = f(\lambda_{ii})$. Thus,

$$\sum_{i=1}^{n} f(g_i^T\Sigma g_i) \geq \sum_{i=1}^{n} g_i^T PQP^T g_i = Tr[G^T PQP^T G]$$

Using the identity $Tr[AB] = Tr[BA]$, we obtain

$$Tr[G^T PQP^T G] = Tr[GG^T PQP^T] = Tr[PQP^T] = Tr[P^T PQ] = Tr[Q] = \sum_{i=1}^{n} f(\lambda_{ii})$$

Furher, notice that $W(P) = \sum_{i=1}^{n} f(g_i^T\Sigma g_i) = \sum_{i=1}^{n} f(p_i^T P\Lambda P^T p_i) = \sum_{i=1}^{n} f(\lambda_{ii})$ □

First we would like to know how to solve for α, given x and G. The solution turns out to be a simple thresholding of the solution to $x = G\alpha$.

Proposition 2. *The vector c defined by*

$$c_i = \begin{cases} g_i^T x & |g_i^T x| \geq \sqrt{\lambda} \\ 0 & otherwise \end{cases}$$

is the solution to $argmin_\alpha(||x - G\alpha||_2^2 + \lambda||\alpha||_0)$.

Proof.

$$||x - G\alpha||_2^2 = ||G^T x - G^T G\alpha||_2^2 = ||G^T x - \alpha||_2^2$$

since G is orthonormal. Furthermore,

$$||x - G\alpha||_2^2 + \lambda||\alpha||_0 = ||G^T x - \alpha||_2^2 + \lambda||\alpha||_0 = \sum_{i=1}^{n} |g_i^T x - \alpha_i|^2 + \lambda I(\alpha_i \neq 0)$$

The minimization can be carried out for each α_i independently and results in the thresholding. □

Using the ideas in propositions 1 and 2, we observe that:

$$E(min_\alpha\{||x - G\alpha||_2^2 + \lambda||\alpha||_0\}) = E(\sum_{i=1}^{n} min_{\alpha_i}\{|g_i^T x - \alpha_i|^2 + \lambda I(\alpha_i \neq 0)\})$$

Since sum distributes over expectations, we obtain

$$E(min_\alpha\{||x - G\alpha||_2^2 + \lambda||\alpha||_0\}) = \sum_{i=1}^{n} E(min_{\alpha_i}\{|g_i^T x - \alpha_i|^2 + \lambda I(\alpha_i \neq 0)\})$$

If x is a zero mean gaussian random variable with covariance matrix Σ, $g_i^T x$ is a scalar gaussian random variable with variance $g_i^T \Sigma g_i$. Using this fact, an individual expectation can be evaluated.

$$E(min_{\alpha_i}\{|g_i^T x - \alpha_i|^2 + \lambda I(\alpha_i \neq 0)\}) \propto \int_0^{\sqrt{\lambda}} y e^{-y^2/2\sigma_i^2} dy + \lambda \int_{\sqrt{\lambda}}^{\infty} e^{-y^2/2\sigma_i^2} dy$$

where $\sigma_i^2 = g_i^T \Sigma g_i$. Thus, the expectation can be written as a function $f(g_i^T \Sigma g_i)$, and the objective function for SOT as a sum of form

$$\sum_{i=1}^{n} f(g_i^T \Sigma g_i)$$

It can be checked that f is a concave function as its second order derivative is always negative. Using proposition 1, P is a minimizer of the objective function and thus SOT is equivalent to KLT on a zero mean Gaussian process.

Finding the SOT

Algorithm

Finding the SOT for a sample of vectors X for given λ works via a 2 step minimizaton:
A random initial transform H is chosen. Then, until convergence,

1. Evaluate the best coefficients (constant transform)
2. Evaluate the best transforms (constant coefficients)

Which is the general trend of dictionary learning algorithms.

Proposition 2 stated how we can perform step 1. Step 2 is demonstrated :
Since the coefficents are set to be constant, we need to minimize

$$E(||x - G\alpha||_2^2)$$

We do not have knowledge of the distribution on x. All we have is a dataset X consisting of a large number of samples. We instead minimize the data driven expected value

$$\hat{E}(||x - G\alpha||_2^2) = ||X - GA||_F^2$$

for an orthogonal matrix G. This is an orthogonal procrustes problem.
Efficient solution of the problem is known and is given by

$$UV^T \text{ where } AX^T = U\Sigma V^T \text{ with orthonormal } U, V \text{ and diagonal } \Sigma$$

We expect to use this algorithm with a few additions (classification and annealing) to beat the KLT on image datasets in terms of compression properties.
For the initial transform provided as an input, we use the KLT.

Peculiarities of the Algorithm

We list a few points we find interesting about the algorithm.

The algorithm is similar to overcomplete dictionary learning using a collection of orthonormal dictionaries but it is different in the sense that reconstruction follows by chosing the best basis. This guarentees that reconstructions would be faster (as we do not solve a complex optimization problem) and unique irrespective of conditions needed in overcomplete dictionary learning.

Drawbacks of the Algorithm

We list a few drawbacks we find in the algorithm.

The algorithm does not function well for low and high values of λ. For low values, we do not observe much change in direction. For high values, accuracy is compromised significantly. Thus, hyper-parameter tuning is a concern.

Further, it is a convergence based algorithm and even though convergence is theoretically guarenteed, it will remain slow compared to KLT, which is a one step transformation. Reclassification and Annealing only slow the process further.

SOT vs KLT Results on Synthetic Test Data

Gaussian Noise

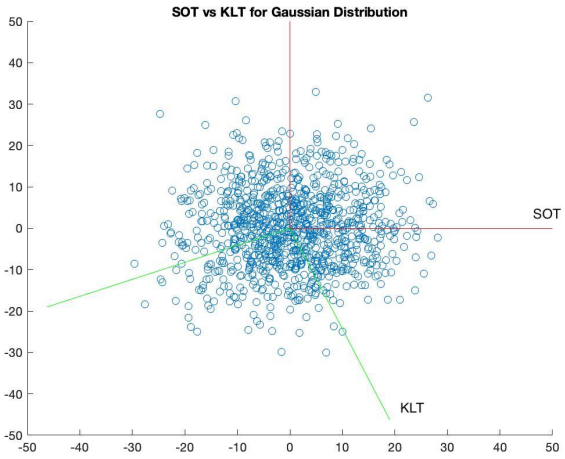

Figure 1: KLT and SOT Matrices for Gaussian Distribution $\mathcal{N}(100I)$

As can be seen, the SOT and KLT both produce optimal directions.

Non Gaussian Noise

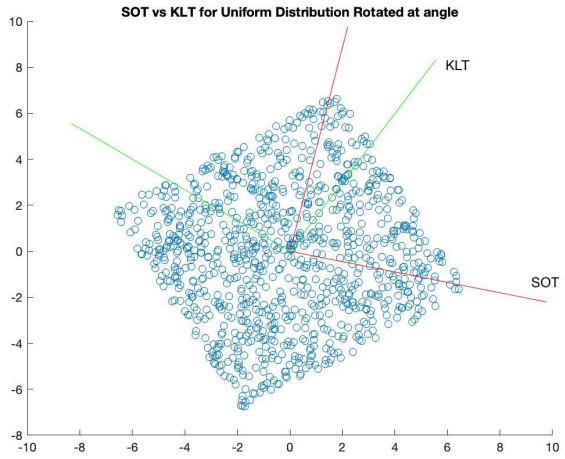

Figure 2: KLT and SOT Matrices for Uniform Distribution rotated by 30° ACW

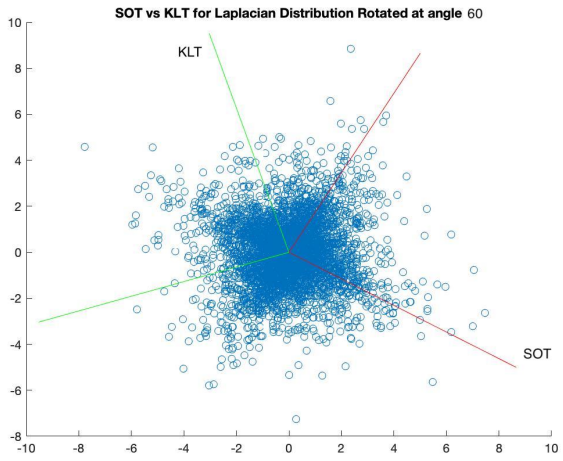

Figure 3: KLT and SOT Matrices for Laplacian Distribution rotated by 60° ACW

In both these cases, we observe that SOT finds a direction where coefficients are expected to be sparser compared to the KLT. In the case of uniform noise, the KLT is degenerate and any orthogonal basis serves as KLT.

This guarentee the initial claim, which is, the SOT and KLT are equivalent on zero mean Gaussian processes whereas the SOT has the capability to outperform KLT on other processes.

Tabulated Results

The NL-cost and relative cost error between the 2 methods is as mentioned in table

Distribution	**SOT Cost**	**KLT Cost**	**Rel. Cost**
Gaussian	1.8900e+03	1.8866e+03	0.0018
Uniform	1.6591e+03	1.6986e+03	0.0232
Laplacian	5.2907e+03	5.7982e+03	0.0875

Hence we can see we get a big difference for non gaussian data between SOT and KLT distributions.

SOT vs KLT vs DCT on Image Compression

The Experiment

Data

A dataset of 8×8 patches obtained from the first 20 images available in `code/data/` folder. The data is obtained from here and each image is converted to greyscale using standard conversion procedures in openCV.

Validation

Since hyperparameter tuning was a concern, we used a range of values of λ and kept $\delta = \lambda$ and $\lambda_{max} = 10\lambda$ for convenience. We trained dictionaries using only 5 of the 20 images obtained with different values of λ.

For every image in the validation set, i.e. images 31-33 in the folder `code/data/`, we perform a non-overlapping patchwise reconstruction using these bases. Only the largest n values (in absolute value sense) are kept in each patch and reconstruction is performed using them. We then plot PSNR between the original image and the reconstructions vs n.

All the plots obtained were of the following form : In the rather cluttered plot, the following observations can be made :

1. For very low values of λ, the PSNR is low throughout. This is because for extremely small values of λ, the cost function for SOT reduces to the squared norm difference between the image and its reconstruction, which, the KLT is known to minimize. Thus, we do not obtain the benefits we expected by using SOT over KLT.

2. For very high values of λ, the PSNR starts off high but drops down significantly as n increases. The straightforward explaination is a high value of λ promotes sparsity above reconstruction accuracy.

We observe that the plots corresponding to $\lambda \in \{1e-3, 2e-3\}$ perform well on a large range of n values. We thus set $\lambda = 2e-3$ for training the final dictionary. The code for this part can be obtained in `code/validation/` directory along with the learnt bases and more plots.

Final Training

A KLT basis is obtained and a group of $K = 8$ SOT bases are obtained by offline training on the complete 20 image dataset. The code can be obtained in `code/finaltraining/` folder.

Note : The basis learning process is time consuming (we needed 15 minutes on a state of the art system) and hence we have saved the matlab variables in a file `code/dict.mat`. We use this for demonstration purposes as the algorithm is deterministic.

Testing

Testing was done on 10 images (21-30 from the dataset). For each image in the test set, reconstructions were obtained using the learng SOT, KLT and DCT bases keeping only the top n entries. PSNR vs n plots were obtained.

Results

Consider the following reconstructions obtained using SOT, KLT and DCT keeping only $n = 10$ coefficients in a batch. SOT significantly outperforms DCT and has a better reconstruction than KLT. A few plots obtained in testing follow : It can be seen that at vert low values of n, SOT and KLT behave similarly but for interesting ranges of n (between 15-35), SOT outperforms

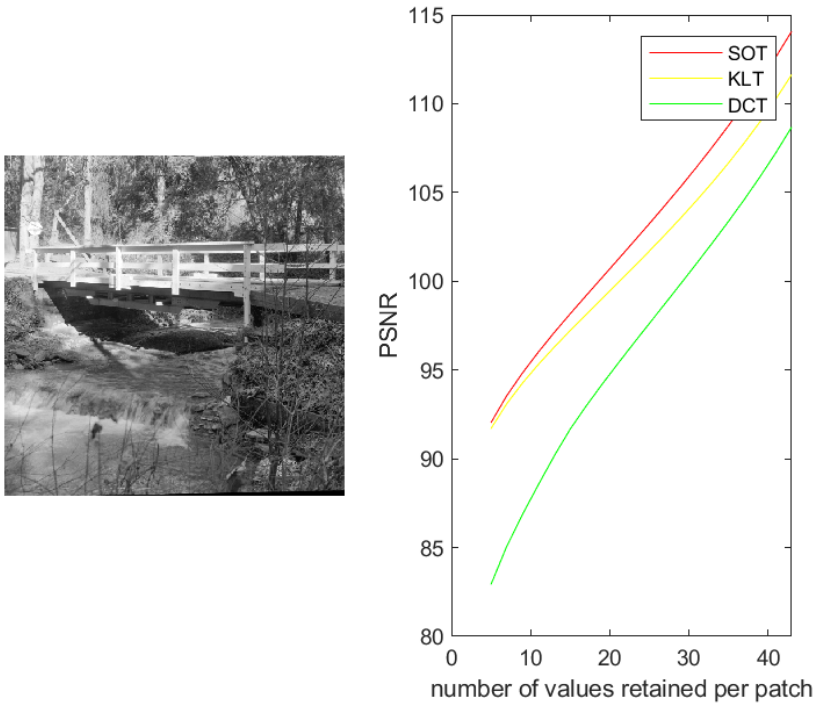

Figure 4: SOT vs DCT vs KLT demonstration 1

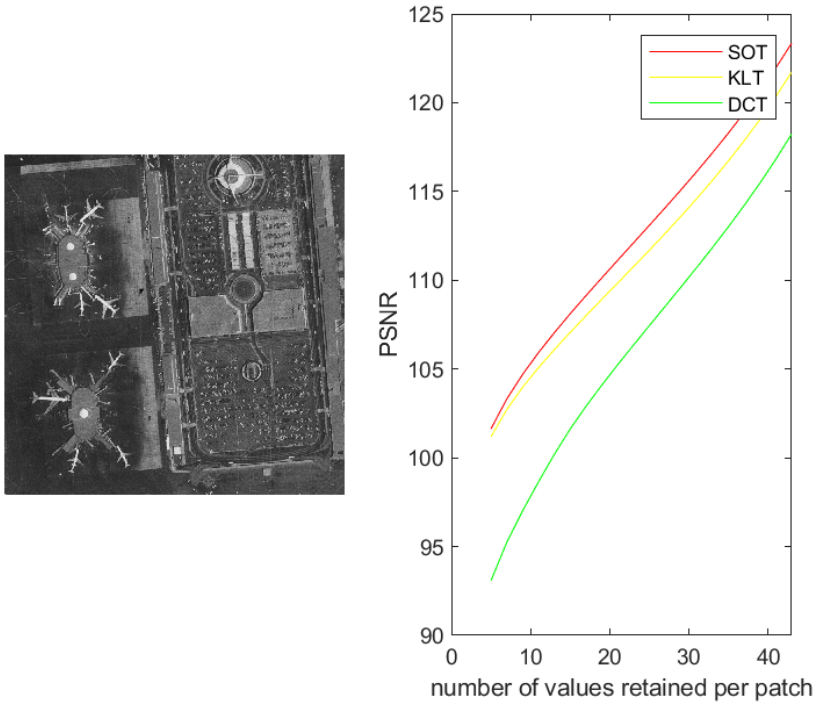

Figure 5: SOT vs DCT vs KLT demonstration 1

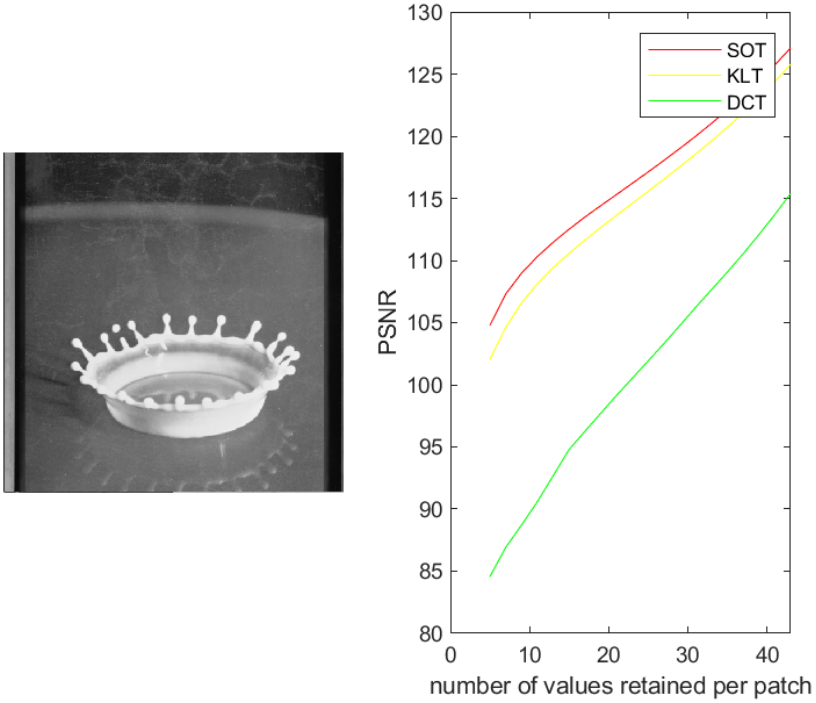

Figure 6: SOT vs DCT vs KLT demonstration 1

KLT. DCT starts off very poor but quickly picks up coming closer to KLT, DCT in the values of $n \approx 20 - 40$, justifying the popularity of the JPEG compression algorithm.

We observe that using a good value of λ, SOT can provide significant benefits over KLT which is what we sought after.

The code and more results can be obtained in `code/testing/` folder.

Comments

The following improvements can be made to the experiment :

1. The use of quad-tree based implementation mentioned in the paper. We expect a slight improvement in border pecularities.
2. Use of adaptive patch sizes along with the Quad Tree based implementation. The paper states a significant improvement in PSNR on using this method.
3. Using the SOT in Wavelet basis might improve performance and is mentioned as a specialization in the paper.
4. Better hyperparameter tuning $(\lambda, \lambda_{max}, \delta, \epsilon)$ with a larger validation set can be performed.