Sparse Coding of Linear Dynamical Systems with an Application to Dynamic Texture Recognition

Bernard Ghanem and Narendra Ahuja
Department of Electrical and Computer Engineering
University of Illinois at Urbana-Champaign, Urbana, IL 61801, USA
{bghanem2,ahuja}@vision.ai.uiuc.edu

Abstract

Given a sequence of observable features of a linear dynamical system (LDS), we propose the problem of finding a representation of the LDS which is sparse in terms of a given dictionary of LDSs. Since LDSs do not belong to Euclidean space, traditional sparse coding techniques do not apply. We propose a probabilistic framework and an efficient MAP algorithm to learn this sparse code. Since dynamic textures (DTs) can be modeled as LDSs, we validate our framework and algorithm by applying them to the problems of DT representation and DT recognition. In the case of occlusion, we show that this sparse coding scheme outperforms conventional DT recognition methods.

1. Introduction

A dynamic texture (DT) sequence captures a random spatiotemporal phenomenon. The randomness reflects in the spatial and temporal changes in the image signal. This may be caused by a variety of physical processes, e.g., involving objects that are small (smoke particles) or large (snowflakes), or rigid (grass, flag) or nonrigid (cloud, fire), moving in 2D or 3D, etc. Even though the overall global motion of a DT may be perceived by humans as being simple and coherent, the underlying local motion is governed by a complex stochastic model. For example, a scene of "translating" clouds conveys visually identifiable global dynamics; however, the implosion and explosion of the cloud segments during the motion result in very complicated local dynamics. Irrespective of the nature of the physical phenomena, the usual objective of DT modeling in computer vision and graphics is to capture the nondeterministic, spatial and temporal variation in images. DT modeling is motivated by a range of applications including recognition, synthesis, and registration of the underlying random processes.

The challenges of DT modeling arise from the need to capture the large number of objects involved, their complex motions, and their intricate interactions. A good model must accurately and efficiently capture both the appearance and global dynamics of a DT. In [8], Doretto et al. showed that the spatiotemporal variations of a DT can be represented using a linear dynamical system (LDS). In Figure 1, we depict the graphical representation of an LDS model, where the gray elements designate the observed features of the DT (e.g. pixel-wise intensities) and the white elements designate the latent state variables. Note the linear relationship between consecutive state variables and between an observation and the state variable at the same time instance. The Markovian nature of the state transitions is evident. In a general LDS, $(\mathbf{C}_t, \mathbf{A}_t)$ is a time dependent pair of matrices. However, a more feasible and widely used case is when this pair is constant over time, whereby learning $(\mathbf{C}, \mathbf{A})$ can be done in closed form [8].

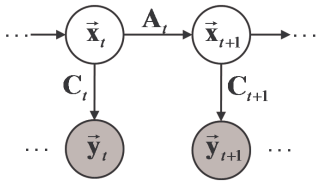

Figure 1. Graphical depiction of a LDS

Based on this LDS modeling, numerous recognition methods have been proposed using the defining LDS parameters $(\mathbf{C}, \mathbf{A})$. Since a LDS model cannot be embedded in a Euclidean space, the need arises for new formulations of distances/similarities between two LDS models. Among the most popular methods, a nearest neighbor (NN) classifier employing a subspace distance (denoted by Martin distance [5]) was proposed in [7]. In [2], a kernel function between two LDS models was

1051-4651/10 $26.00 © 2010 IEEE
DOI 10.1109/ICPR.2010.247

proposed and used in a support vector machine (SVM) framework to perform DT recognition. More recent work has addressed shift [9] and view invariant [6] DT recognition. The latter work extends the use of the popular bag-of-features model to the non-Euclidean space of LDS models.

Despite the merits of these methods, they are all sensitive to input variations due to noise. This is especially the case when the noise results from occlusion. To alleviate this drawback, we make use of the developments in other recognition problems (e.g. face recognition [10]) that utilize linear sparse coding of data based on a predefined dictionary. Using only a sparse set of training models to represent a test sample is at the heart of popular model selection methods (e.g. minimum description length) and classification methods (e.g. SVM). Using sparse representation and compressed sensing theory [4], it was shown in [10] that a high performance face classifier could outperform state-of-the-art classifiers, especially when the test samples were corrupted with noise (e.g. occlusion). Theoretical guarantees were also provided to bound the recognition performance. However, there does not appear to be a simple extension of traditional sparse coding for DT recognition. Before this can be achieved, the problem of sparsely representing a DT using a predefined set of LDS models needs to be solved. To the best of our knowledge, this paper represents the first time such a problem has been formulated and addressed.

In Section 2, we give a detailed presentation of this new sparse coding problem and the probabilistic model we use to solve it. This model is embedded in a MAP framework and an efficient model learning algorithm is described. In Section 3, we present experimental results on the representation and recognition of real world DT video sequences. These results validate the effectiveness of our model and learning algorithm.

2. Proposed Model

In this paper, we model a DT as a LDS that is represented by a pair of matrices ($\mathbf{C}$,$\mathbf{A}$) [8]. Matrix $\mathbf{C}$ describes how the observed features (e.g. intensity values) of a DT frame are produced from the hidden states. Matrix $\mathbf{A}$ describes the transition between hidden states of consecutive frames. The representation is given by the following equations, where $\vec{\mathbf{n}}_y$ and $\vec{\mathbf{n}}_x$ are noise components in the observation and state space, respectively, and F is the number of DT frames:

$$\begin{cases} \vec{\mathbf{y}}_t = \mathbf{C}\vec{\mathbf{x}}_t + \vec{\mathbf{n}}_y \\ \vec{\mathbf{x}}_{t+1} = \mathbf{A}\vec{\mathbf{x}}_t + \vec{\mathbf{n}}_x \end{cases} \quad \forall t = 1, \cdots, F$$

We are given a set of L DT sequences for training, from which we extract L LDS models (i.e. $\mathcal{M} = \{(\mathbf{C}_i, \mathbf{A}_i)\}_{i=1}^{L}$). The LDS model parameters are learned by the closed form method proposed in [8], where $\mathbf{C}_i \in \mathbb{R}^{M \times N}$ and $\mathbf{A}_i \in \mathbb{R}^{N \times N}$. M is the size of the observed feature vector per frame and N is the LDS model size. The task of learning a sparse code for a target video sequence, which can be modeled as an LDS, is equivalent to learning a sparse set of linear coefficients ($\vec{\alpha} \in \mathbb{R}^L$) that combines the LDS models of $\mathcal{M}$ to best generate the frames of the target. Here, $\vec{\alpha}$ is the sparse code of the target with respect to $\mathcal{M}$. Given $\mathcal{M}$, we model the target DT as follows. F is the number of frames in the target DT.

$$\begin{cases} \vec{\mathbf{y}}_t = \sum_{i=1}^{L} \alpha_i \mathbf{C}_i \vec{\mathbf{x}}_t + \vec{\mathbf{n}}_y \\ \vec{\mathbf{x}}_{t+1} = \sum_{i=1}^{L} \alpha_i \mathbf{A}_i \vec{\mathbf{x}}_t + \vec{\mathbf{n}}_x \end{cases} \quad \forall t = 1, \cdots, F \quad (1)$$

For a given target DT, its sparse coding with respect to $\mathcal{M}$ is equivalent to finding the sparsest $\vec{\alpha}$ and learning the hidden state variables $\{\vec{\mathbf{x}}_t\}_{t=1}^{F}$ that best fit the model of Eq. (1). Since $\vec{\mathbf{n}}_y$ and $\vec{\mathbf{n}}_x$ are i.i.d. Gaussian random vectors, this sparse coding problem can be viewed as a signal detection problem. As such, $\{\vec{\mathbf{x}}_t\}_{t=1}^{F}$ and $\vec{\alpha}$ are learned by embedding them into a maximum a posteriori (MAP) framework that fits the model in Eq. (1) to the observed data.

To do this, we form the joint probability: $p\left(\{\vec{\mathbf{y}}_t\}_{t=1}^{F}, \{\vec{\mathbf{x}}_t\}_{t=1}^{F}, \vec{\alpha}\right) = \mathcal{L}\mathcal{P}_x\mathcal{P}_\alpha$. Here, $\mathcal{L}$ represents the likelihood of the observed data, $\mathcal{P}_x$ represents the prior of the state variables (i.e. Markovian property), and $\mathcal{P}_\alpha$ represents the prior of the sparse code.

To conform to the model in Eq. (1), we decompose these three terms further. Since the DT observations are conditionally independent of each other, we decompose $\mathcal{L}$ as: $\mathcal{L} = p\left(\{\vec{\mathbf{y}}_t\}_{t=1}^{F} \mid \{\vec{\mathbf{x}}_t\}_{t=1}^{F}, \vec{\alpha}\right) = \prod_{t=1}^{F} p\left(\vec{\mathbf{y}}_t \mid \vec{\mathbf{x}}_t, \vec{\alpha}\right)$. Due to the Markovian property governing the state variables, we decompose $\mathcal{P}_x$ as: $\mathcal{P}_x = p\left(\{\vec{\mathbf{x}}_t\}_{t=1}^{F} \mid \vec{\alpha}\right) = p\left(\vec{\mathbf{x}}_1\right) \prod_{t=1}^{F-1} p\left(\vec{\mathbf{x}}_{t+1} \mid \vec{\mathbf{x}}_t, \vec{\alpha}\right)$. Assuming conditional independence between the sparse code elements, we decompose $\mathcal{P}_\alpha$ as: $\mathcal{P}_\alpha = p\left(\vec{\alpha}\right) = \prod_{i=1}^{L} p\left(\alpha_i\right)$.

To formalize this framework, we model each of the above probabilities. Both the likelihood of an observation and the state prior probability are modeled as Gaussian distributions, according to Eq. (1). In detail, we have: $p\left(\vec{\mathbf{y}}_t \mid \vec{\mathbf{x}}_t, \vec{\alpha}\right) \sim \mathcal{N}\left(\sum_{i=1}^{L} \alpha_i \mathbf{C}_i \vec{\mathbf{x}}_t, \sigma_y^2 \mathbf{I}\right)$ and $p\left(\vec{\mathbf{x}}_{t+1} \mid \vec{\mathbf{x}}_t, \vec{\alpha}\right) \sim \mathcal{N}\left(\sum_{i=1}^{L} \alpha_i \mathbf{A}_i \vec{\mathbf{x}}_t, \sigma_x^2 \mathbf{I}\right)$. In order to guarantee sparsity in $\vec{\alpha}$, we model its prior as a Laplacian distribution: $p\left(\alpha_i\right) \sim \text{Laplace}\left(0, \lambda_\alpha\right)$.

MAP Formulation: After modeling the joint distribution, we proceed to maximize it by minimizing its neg-

ative log, as in Eq. (2). As we can see, there are three main terms in this cost function: **(1)** a frame reconstruction term that evaluates how well the individual DT observations are reproduced, **(2)** a state prediction term that evaluates how well the state variables are predicted, and **(3)** a regularization term that ensures the sparsity of $\vec{\alpha}$. Note that these three terms are weighted differently, according to σ_y, σ_x, and λ_α. More importantly, these weights are data-driven, so there is no need for user-defined constants. Next, we apply the method of Iterated Conditional Modes (ICM) [1] to solve Eq. (2). In this method, we perform blockwise coordinate descent on the cost function. In each ICM iteration, we fix all the variables except one and then minimize the function with respect to that variable.

$$\min_{\vec{\alpha},\{\vec{\mathbf{x}}_t\}_{t=1}^F} FM\ln(\sigma_y) + \frac{1}{2\sigma_y^2}\sum_{t=1}^{F}\|\vec{\mathbf{y}}_t - \sum_{i=1}^{L}\alpha_i\mathbf{C}_i\vec{\mathbf{x}}_t\|_2^2 + L\ln\lambda_\alpha + \frac{\|\vec{\alpha}\|_1}{\lambda_\alpha} + \frac{1}{2\sigma_x^2}\sum_{t=1}^{F-1}\|\vec{\mathbf{x}}_{t+1} - \sum_{i=1}^{L}\alpha_i\mathbf{A}_i\vec{\mathbf{x}}_t\|_2^2 + (F-1)N\ln\sigma_x \quad (2)$$

In the $(k+1)^{\text{th}}$ ICM iteration, we have the following update rules for the sparse code, the state variables, and the distribution parameters.

• **Update $\vec{\alpha}$:** Fixing all variables except $\vec{\alpha}$, we obtain a convex quadratic program whose global minimum can be obtained using gradient descent. The update is: $\vec{\alpha}^{(k+1)} = \arg\min_{\vec{\mathbf{z}}\in\mathbb{R}^L} \frac{1}{\sigma_y^{2(k)}}\sum_{t=1}^{F}\|\vec{\mathbf{y}}_t - \sum_{i=1}^{L}\mathbf{z}_i\mathbf{C}_i\vec{\mathbf{x}}_t^{(k)}\|_2^2 + \frac{1}{\sigma_x^{2(k)}}\sum_{t=1}^{F-1}\|\vec{\mathbf{x}}_{t+1}^{(k)} - \sum_{i=1}^{L}\mathbf{z}_i\mathbf{A}_i\vec{\mathbf{x}}_t^{(k)}\|_2^2 + \frac{\|\vec{\mathbf{z}}\|_1}{\lambda_\alpha^{(k)}}$.

• **Update $\vec{\mathbf{x}}_t$:** Fixing all variables except $\vec{\mathbf{x}}_t$, we obtain a convex quadratic program whose global minimum can be obtained in closed form. The update is: $\vec{\mathbf{x}}_t^{(k+1)} = \arg\min_{\vec{\mathbf{z}}\in\mathbb{R}^N} \frac{1}{\sigma_y^{2(k)}}\|\vec{\mathbf{y}}_t - \sum_{i=1}^{L}\alpha_i^{(k)}\mathbf{C}_i\vec{\mathbf{z}}\|_2^2 + \frac{I_{\{t=F\}}}{\sigma_x^{2(k)}}\|\vec{\mathbf{x}}_{t+1}^{(k)} - \sum_{i=1}^{L}\alpha_i^{(k)}\mathbf{A}_i\vec{\mathbf{z}}\|_2^2 + \frac{I_{\{t=1\}}}{\sigma_x^{2(k)}}\|\vec{\mathbf{z}} - \sum_{i=1}^{L}\alpha_i^{(k)}\mathbf{A}_i\vec{\mathbf{x}}_{t-1}^{(k)}\|_2^2$. Here, $I_{\{t=F\}}$ and $I_{\{t=1\}}$ are the indicator functions for the first and last frames.

• **Update $\sigma_x, \sigma_y, \lambda_\alpha$:** We update these parameters to their ML estimates, as follows.

$$\begin{cases}\sigma_y^{2(k+1)} = \frac{1}{FM}\sum_{t=1}^{F}\|\vec{\mathbf{y}}_t - \sum_{i=1}^{L}\alpha_i^{(k)}\mathbf{C}_i\vec{\mathbf{x}}_t^{(k)}\|_2^2 \\ \sigma_x^{2(k+1)} = \frac{1}{(F-1)N}\sum_{t=1}^{F-1}\|\vec{\mathbf{x}}_{t+1}^{(k)} - \sum_{i=1}^{L}\alpha_i^{(k)}\mathbf{A}_i\vec{\mathbf{x}}_t^{(k)}\|_2^2 \\ \lambda_\alpha^{(k+1)} = \frac{\|\vec{\alpha}^{(k)}\|_1}{L}\end{cases}$$

• **Initialization:** Since ICM is an iterative method that only guarantees a local minimum for Eq. (2), a good initialization of the variables is essential. To initialize the sparse code, we assume a sparsity of K (e.g. $K = 10$) for $\vec{\alpha}^{(0)}$. We build a separate LDS model for the target, as described in [8]. Then, we use this learned target model to determine the K "closest" training LDS models in $\mathcal{M}$. This is done by using the Martin distance [3,5]. The nonzero elements of $\vec{\alpha}^{(0)}$ correspond to these K "closest" training models and they are set to 1. To initialize each state variable, we need to solve a separate least-squares problem, whose closed form solution is: $\vec{\mathbf{x}}_t^{(0)} = \mathbf{C}_T^\dagger\vec{\mathbf{y}}_t$. Here, $\mathbf{C}_T = \sum_{i=1}^{L}\alpha_i^{(0)}\mathbf{C}_i$.

3. Experimental Results

In this section, we will evaluate the performance of our sparse LDS coding method when applied to two DT applications: **(I)** DT representation and **(II)** DT recognition. In both applications, the sparse code ($\vec{\alpha}$) of a sample DT sequence is computed for a set of L LDS models ($\mathcal{M}$), as in Section 2. In application **(I)**, $\vec{\alpha}$ is used to reconstruct the sample DT, while in application **(II)**, it is used to recognize the class of the sample DT as one of the DT classes in $\mathcal{M}$. We use the benchmark UCLA DT dataset [8] for both applications. This dataset contains 50 classes of gray-scale DT, each of which is comprised of 4 DT sequences. Each sequence has $F = 75$ frames with $M = 48 \times 48$ pixels. In all our experiments, we use pixelwise intensities of the DT frames as the observed feature vectors.

(I) DT Representation: For a given DT sequence in the dataset, we first compute its sparse code and state variables, where all other DTs are used as the set of training LDS models $\mathcal{M}$. Given the code and state variables, we can reconstruct the DT sequence, according to Eq. (1). Figure 2 shows the learned sparse code $\vec{\alpha}$ for a sample DT sequence, at the first and last ICM iterations. The Peak-Signal-to-Noise-Ratio (PSNR) between the reconstructed DT sequence and the original one is 24dB, which demonstrates faithful DT reconstruction. Note that the code at the last ICM iteration is much sparser than the initial code ($K = 10$) and that the highest α values occur for the LDS models from $\mathcal{M}$ corresponding to the class to which the sample DT sequence belongs. This latter observation motivates why our LDS sparse code method is suitable for DT recognition, to be addressed in the next section.

(II) DT Recognition: The UCLA dataset contains 50 gray-scale DT classes, each of which is comprised of 4 DT sequences. Four-fold cross validation is performed on this dataset and average recognition performance is reported. Using sparse LDS coding, we perform recognition according to sparse representation-based classifi-

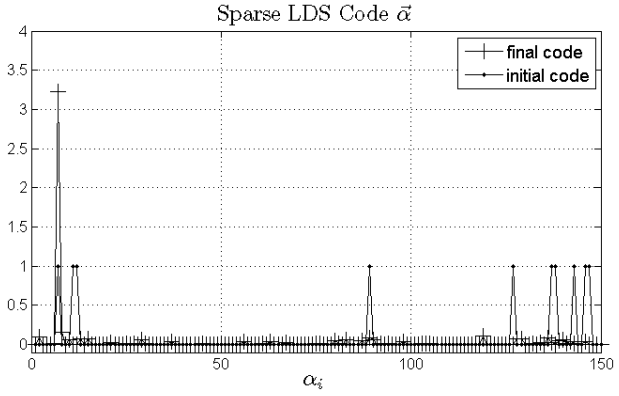

Figure 2. Example of DT sparse representation

cation (SRC) proposed in [10]. In SRC, the class of a test sequence is determined by the single class of training LDS models, which leads to the smallest reconstruction error among all classes. More formally, we define the reconstruction error with a certain sparse code $\vec{\alpha}$ as $e\left(\vec{\alpha}\right) = \sum_{t=1}^{F} \|\vec{\mathbf{y}}_t - \sum_{i=1}^{L} \alpha_i \mathbf{C}_i \vec{\mathbf{x}}_t\|_2^2$. If $\vec{\alpha}^k$ defines the sparse code of class k (i.e. all the α values are set to zero if they do not correspond to class k), then the recognized SRC class of a given test sequence is $c_T = \arg\min_k e\left(\vec{\alpha}^k\right)$. As a benchmark comparison, we perform recognition using a NN classifier (as in [8]), where the distance used is the Martin distance. In Table 1, we report the average recognition rates, where the maximum rate is achieved at $N = 25$ for both methods. Our SRC method clearly outperforms the benchmark method.

	N=10	N=25	N=40	N=55	N=70
SRC(%)	89.0	**95.5**	93.0	91.5	89.0
NN (%)	87.0	**92.5**	90.5	88.0	86.0

Table 1. DT recognition rates vs. N

Next, we evaluate the performance of our SRC method under varying conditions of structured occlusion. To do this, we occlude a varying percentage of each frame of the test sample before recognition is performed. Within each test frame, we use a single, randomly positioned square, whose intensity values are uniformly distributed in $[0, 255]$, to occlude the image. We fix $N = 25$ for this experiment. As before, we use the NN classifier based on Martin distance for comparison. Figure 3 plots the average recognition rate as a function of the percentage of occlusion. The performance of our SRC method decreases much more gradually with occlusion, as compared to the NN classifier. This is primarily due to the fact that our method explicitly incorporates noise in the proposed model. On the other hand, the NN classifier builds a separate LDS (global) model for the test sample, thus, rendering it more prone to occlusion.

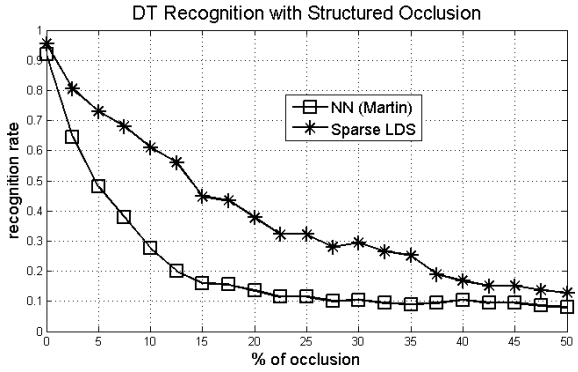

Figure 3. DT Recognition with occlusion

4. Conclusion

This paper proposes the novel problem of sparse coding for LDS models and applies it to the problem of DT recognition. Since LDS parameters lie in a non-Euclidean space, we learn the sparse code in a probabilistic framework. We present experiments showing that this method outperforms benchmark recognition methods, especially with occlusion. In this paper, we used a predefined dictionary of LDS models. In the future, we aim to extend this coding approach to learn dictionaries of LDS models.

References

[1] J. Besag. On the statistical analysis of dirty pictures. *Journal of the Royal Statistical Society*, 48(3):259–302, 1986.

[2] A. B. Chan and N. Vasconcelos. Probabilistic kernels for the classification of auto-regressive visual processes. In *CVPR*, volume 1, pages 846–851, 2005.

[3] K. DeCock and B. De Moor. Subspace angles between ARMA models. In *Systems and Control Letters*, volume 46, pages 265–270, 2002.

[4] D. Donoho. For most large underdetermined systems of linear equations the minimal l1-norm solution is also the sparsest solution. *Comm. on Pure and Applied Math*, 59:797–829, 2006.

[5] R. J. Martin. A metric for arma processes. *IEEE Trans. on Signal Processing*, 48:1164–1170, 2000.

[6] A. Ravichandran, R. Chaudhry, and R. Vidal. View-invariant dynamic texture recognition using a bag of dynamical systems. In *CVPR*, pages 1651–1657, 2009.

[7] P. Saisan, G. Doretto, Y. N. Wu, and S. Soatto. Dynamic texture recognition. In *CVPR*, pages 58–63, 2001.

[8] S. Soatto, G. Doretto, and Y. N. Wu. Dynamic textures. *IJCV*, 51:91–109, 2003.

[9] F. Woolfe and A. W. Fitzgibbon. Shift-invariant dynamic texture recognition. In *ECCV*, pages 549–562, 2006.

[10] J. Wright, A. Y. Yang, A. Ganesh, S. S. Sastry, and Y. Ma. Robust face recognition via sparse representation. *PAMI*, 31:210–227, 2009.