Computer Science and Artificial Intelligence Laboratory
Technical Report

MIT-CSAIL-TR-2005-041
AIM-2005-021
June 6, 2005

Nonlinear Latent Variable Models for Video Sequences

ali rahimi, ben recht, and trevor darrell

massachusetts institute of technology, cambridge, ma 02139 usa — www.csail.mit.edu

Abstract

Many high-dimensional time-varying signals can be modeled as a sequence of noisy nonlinear observations of a low-dimensional dynamical process. Given high-dimensional observations and a distribution describing the dynamical process, we present a computationally inexpensive approximate algorithm for estimating the inverse of this mapping. Once this mapping is learned, we can invert it to construct a generative model for the signals. Our algorithm can be thought of as learning a manifold of images by taking into account the dynamics underlying the low-dimensional representation of these images. It also serves as a nonlinear system identification procedure that estimates the inverse of the observation function in nonlinear dynamic system. Our algorithm reduces to a generalized eigenvalue problem, so it does not suffer from the computational or local minimum issues traditionally associated with nonlinear system identification, allowing us to apply it to the problem of learning generative models for video sequences.

1 Introduction

Video is a very high-dimensional signal, yet a few smoothly varying degrees of freedom govern the appearance of many dynamic scenes. These latent degrees of freedom may correspond to the pose of objects, illumination conditions, or other physical states of the scene. Given a video sequence, we would like to learn a generative model of the appearance of the scene that takes into account these low-dimensional processes. Once such a model is learned, it becomes straightforward to perform traditional time series tasks, such as predicting future frames, denoising, classification, anomaly detection, and dimensionality reduction. We describe an efficient approximate algorithm for learning such a latent variable generative model for high-dimensional time series such as video.

A common generative model for time series assumes each observation $y_t \in \mathcal{R}^D$ is generated by an unknown nonlinear function $g : \mathcal{R}^d \rightarrow \mathcal{R}^D$ of a corresponding state x_t:

$$y_t = g(x_t) + \nu_t, \tag{1}$$

where the ν_t model observation noise and are iid. Assuming the appearance manifold of images in the sequence does not intersect itself, g is one-to-one [18, 1, 3]. The low-dimensional states x_t are not observed, but are presumed to evolve according to a first order linear Gaussian autoregressive process:

$$x_{t+1} = \mathbf{A}x_t + \omega_t, \tag{2}$$

where $\mathbf{A}$ is a state transition matrix, and ω_t are iid zero-mean Gaussian variables with variance Σ. Together, equations (1) and (2) define a generative model $p(\mathbf{X}, \mathbf{Y}|g)$ over states and observations, with equation (1) defining $p(y_t|x_t, g)$ and equation (2) defining a prior, $p(X)$, over states.

We would like to estimate g given a video sequence and the prior $p(\mathbf{X})$. But with high-dimensional observations such as video frames, even when the dynamics of the low-dimensional process are known a priori, it is impractical to estimate g with existing nonlinear system identification techniques [4, 20]. This is because traditional representations of nonlinear functions such as multilayer perceptrons (MLP) [20] and radial basis functions (RBF) [4] require many parameters to represent a mapping to a high-dimensional space. Further, even though they make various simplifying approximations, existing techniques are susceptible to local minima. Hence, as far as we are aware, only linear system identification techniques have been used to learn state-space models of video [2, 16]. Learning a k-step-ahead predictor [7] for the sequence directly instead of a latent variable model also requires a large number of parameters to output high-dimensional outputs. Further, because these models do not build a latent variable model, many of the tasks we mentioned above become intractable.

We subvert the problems that occur in searching for g by searching for a function f that maps observations to states that agree with the prior $p(\mathbf{X})$. When adopting an RBF representation for f, this estimation problem reduces to an eigenvalue problem in which high dimensional observations are summarized by entries of a kernel matrix. Finding f is not subject to local minima, and the dimensionality of the data set does not factor into the computational or storage complexity of the algorithm, except to compute the kernel matrix. The inverse of this projection can later be implicitly plugged back into the generative model, and we can evaluate $p(\mathbf{X}|\mathbf{Y})$ and $p(\mathbf{Y})$, and sample from $p(\mathbf{Y})$ without explicitly computing and storing g.

The small storage requirement and the fast and local-minimum-free computations of this algorithm make it well suited for high-dimensional data sets like video sequences. We show experimentally that the f we obtain is close to the inverse of the true g, when the true g is invertible. These experiments demonstrate that our algorithm learns a sensible low-dimensional representation governing the appearance of frames.

2 Related Work

This work is an unsupervised extension of a semi-supervised regression algorithm for learning mappings from images in a video sequence to underlying representations [11].

Our algorithm is closely related to manifold learning and nonlinear Independent Component Analysis (ICA) algorithms. Jenkins and Mataric [6] have extended Isomap [18] by explicitly assigning similar low-dimensional coordinates to temporally adjacent samples.

In addition to our use of dynamics, a notable difference between our algorithm and the general manifold learning framework laid out in [15], the nonlinear ICA algorithm of [19], or [9] is that instead of learning a mapping from states to observations, we learn mappings from observations to states, which reduces the storage and computational requirements when processing high-dimensional data.

Instead of basing our optimization problem on an information theoretic criterion, as in ICA [10], our cost function is based on a generative model and can be optimized quickly using simple linear algebra. We show that a variant of kernel PCA [14] is obtained in the limiting case where there are no dynamics in the low-dimensional representation.

3 The Identification Problem

Given a generative model $p(\mathbf{X}, \mathbf{Y})$ for the observations and the latent states, one can generate a nonlinear video texture by sampling from $p(\mathbf{Y})$ (see [2] for a linear model); one could detect the location of an anomaly in a sequence by examining the posterior state path $p(\mathbf{X}|\mathbf{Y})$ (see [5] for an example that uses [20]); given several models $p_1(\mathbf{Y}, \mathbf{X}), p_2(\mathbf{Y}, \mathbf{X}), \cdots$, one could classify a sequence $\mathbf{Y}$ by picking the model that assigns the highest probability $p_i(\mathbf{Y})$ to $\mathbf{Y}$. We wish to learn such models given an observed sequence $\mathbf{Y}_o$.

The generative model (1)-(2), along with a prior $p(g)$ on mappings, define the joint distribution

$$-\frac{1}{2}\log p(\mathbf{X}, \mathbf{Y}, g) = \sum_{t=1}^{T} \|x_{t+1} - \mathbf{A}x_t\|_{\Sigma}^2 + \frac{1}{\sigma^2}\|y_t - g(\mathbf{H}x_t)\|^2 + \kappa(\sigma, \Sigma, \mathbf{A}) - 2\log p(g). \tag{3}$$

Here, through the given permutation matrix $\mathbf{H}$, we have explicitly selected the components of x_t that are mapped to y_t. For example, when the components of x_t are the position and velocity of an object, $\mathbf{H}$ could be used to select only the position. The term κ is a normalizer that does not depend on g.

After observing a sequence $\mathbf{Y}_o$ and picking a representation for g, [4] uses EM to find the ML parameters of the model while marginalizing out the state sequence $\mathbf{X}$, and [20] uses Variational Bayes to find the posterior over $\mathbf{X}$ and the parameters of the model. These algorithms are approximate: the E-step of both algorithms relies on approximate estimation, and the Variational Bayes formulation of [20] assumes a convenient functional form for the posteriors. Despite these approximations, these algorithms are subject to local minima. Another problem with solving for g directly is that both the RBF and MLP representations require too many parameters to represent mappings to video frames. Representing g using RBFs $g(x) = \sum_{t=1}^{T} c_t k(x, x_t)$ requires as many parameters as there are pixels in the entire video frame: each coefficient vector c_t is D-dimensional (the number of pixels in the frame), and there are T are frames in the sequence. The MLP representation has a similar requirement. This storage requirement may not be acceptable for large sequences, and operating on such a representation may be too computationally intensive. These algorithms can estimate all identifiable model parameters, but the issues pointed out persist even when the task is to estimate the parameters of g only.

We show how to alleviate these problems in Section 4.1 by approximating the functional (3) by a functional over a mapping from observations to states. This optimization can then be solved exactly. For convenience, we assume the dynamics decouple a priori into identical independent chains. Under this assumption, $\mathbf{A}$ and Λ are block diagonal matrices with repeated diagonal blocks. The matrix $\mathbf{H}$ extracts the same elements from each of these blocks. Assuming identical independent chains simplifies the presentation, but the extension of our technique to an arbitrary linear Gaussian dynamics model is straightforward.

4 Approach

Our algorithm is based on a function learning interpretation of a variant of kernel PCA [14]. Consider the least squares cost function

$$\min_{f,\mathbf{X}} \sum_{t=1}^{T} \|x_t - f(y_t)\|^2 + \lambda \sum_{i=1}^{d} \|f_i\|_k^2 \tag{4}$$

$$\text{s.t. } \frac{1}{T}\mathbf{X}\mathbf{X}^\top = \mathbf{I}, \tag{5}$$

where $\|\cdot\|_k$ is a norm on a reproducing kernel Hilbert space of functions and is defined by a positive definite kernel $k(y, y')$ (see, for example, [17]). The norm of each dimension of f is penalized individually. The representer theorem [13] states that the optimal f has the form $f(y) = \sum_{t=1}^{T} c_t k(y, y_t)$, where each c_t is a d-dimensional vector. When $k(y, y')$ is a radial function, $f(y)$ is an RBF. Substituting in this form, the optimization (4) can be rewritten as:

$$\min_{\mathbf{C},\mathbf{X}} \|\mathbf{X} - \mathbf{C}\mathbf{K}\|_F^2 + \lambda \operatorname{tr} \mathbf{C}\mathbf{K}\mathbf{C}^\top \tag{6}$$

$$\text{s.t. } \frac{1}{T}\mathbf{X}\mathbf{X}^\top = \mathbf{I}, \tag{7}$$

where $\|\cdot\|_F$ is the Frobenius norm, $\mathbf{K}$ is the kernel matrix whose $t\tau$th element is $k(y_t, y_\tau)$, and $\mathbf{C}$ is the matrix $\mathbf{C} = \{c_t\}_1^T$ of coefficients of f.

Finding the optimal $\mathbf{X}^*$ reduces to extracting the d largest eigenvectors of the kernel matrix $\mathbf{K}$. The rows of the optimal $\mathbf{C}^*$ are scaled versions of the rows of $\mathbf{X}^*$ (see [12] for a more thorough derivation). Plugging $\mathbf{C}$ back into the representer form results in a function that is a scaled versions of the function recovered by the kernel PCA algorithm of Schölkopf et al.[14].

This interpretation reveals that KPCA looks for a smooth function f that projects the sequence of observations $\mathbf{Y}$ to a low-dimensional sequence $\mathbf{X}$ so that the dimensions of $\mathbf{X}$ are orthogonal to each other and have unit sample variance. By placing an additional prior on the hidden sequence $\mathbf{X}$, we can refine this algorithm to take in to account a wide range of Gaussian priors. In particular, we can constrain the $\mathbf{X}$ to have linear Gaussian dynamics.

4.1 Learning a Mapping with Dynamics

To search for a function f that maps observations to state sequences that agree with the prior $p(\mathbf{X})$, we augment the KPCA cost function with a dynamics model:

$$\min_{f,\mathbf{X}} \sum_{t=1}^{T} \|\mathbf{H}x_t - f(y_t)\|^2 + \|x_t - \mathbf{A}x_{t-1}\|_\Sigma^2 + \lambda \sum_{i=1}^{d} \|f_i\|_k^2 \tag{8}$$

$$\text{s.t. } \frac{1}{T}\mathbf{H}\left(\sum_{t=1}^{T} x_t x_t^\top\right)\mathbf{H} = \sigma_\infty^2 I \tag{9}$$

$$\frac{1}{T}\mathbf{H}\sum_{t=1}^{T} x_t = 0. \tag{10}$$

The additional term in the cost function measures the compatibility of the projection with some of the components of $\mathbf{X}$ selected by the given permutation matrix $\mathbf{H}$. The constraints ensure that the empirical distribution of x's match the steady state distribution of the states. Note that the constraint on the second moments is identical to the constraint in KPCA, except for the scale factor σ_∞^2, the steady state variance $\lim_{t\to\infty} cov(x_t)$, given by the prior dynamics model. Equation (8) is reminiscent of Equation (3), with the notable difference that observations are mapped to states, instead of the other way. Later, we show experimentally that the optimizing f is close to the inverse of the true g in the generative model (1)-(2).

Applying the representer theorem, and keeping in mind that the dynamics decouple, the optimum of the cost function can be found by solving an eigenvalue problem [12]:

$$\begin{aligned}
\mathbf{Z}^\top \mathbf{S}\mathbf{Z}\mathbf{U}^* &= T\sigma_\infty^2 \mathbf{Z}^\top \hat{\mathbf{H}}^\top \hat{\mathbf{H}}\mathbf{Z}\mathbf{U}^*\mathbf{D}^* && (11)\\
\mathbf{V} &= (\mathbf{K} + \lambda\mathbf{I})^{-1}\,\hat{\mathbf{H}} && (12)\\
\mathbf{S} &= \Omega + \hat{\mathbf{H}}^\top \hat{\mathbf{H}} - \mathbf{V}\mathbf{K}\hat{\mathbf{H}} && (13)\\
\mathbf{Z} &= \text{null}(\hat{\mathbf{H}}\mathbf{1}) && (14)\\
\mathbf{X}^* &= \left(\hat{\mathbf{H}}\mathbf{Z}\mathbf{U}^*\right)^\top && (15)\\
\mathbf{C}^* &= \mathbf{V}\mathbf{Z}\mathbf{U}^* && (16)
\end{aligned}$$

The matrix $\mathbf{X}^*$ and $\mathbf{C}^*$ are the optimal path and RBF coefficients, respectively. The d columns of $\mathbf{U}^*$ contain the eigenvectors of the generalized eigenvalue problem (11), and the diagonal matrix $\mathbf{D}$ contains the resulting generalized eigenvalues. The matrix $\hat{\mathbf{H}}$ replicates the repeating blocks of $\mathbf{H}$ T times instead of d times. The vector $\mathbf{1}$ is a column vector consisting of all 1s. The columns of $\mathbf{Z}$ span the null space of $\hat{\mathbf{H}}\mathbf{1}$, so they span the space of solutions to the mean constraint (10). The matrix Ω is the block-tridiagonal inverse covariance matrix of the independent Markov chains defined by the dynamics. The eigenvalue problem (11) is of size $(dT-1) \times (dT-1)$, and can be solved quickly since we only require the top d eigenvectors.

4.2 Substituting into the Generative Model

Once the function f^* and the latent states X^* have been estimated, the inverse $\hat{g}(x)$ of f can be computed, specifying the generative model (1)-(2). However, we wish to avoid representing $\hat{g}$ in an explicit non-parametric form such as MLP or RBF, as this would be unwieldy.

To evaluate $\hat{g}$ at a particular x, we search for the k nearest neighbors of x in X^*, and interpolate between their corresponding y's to obtain $y_0 = \hat{g}(x)$. Since $y_1 = g(x)$ minimizes the cost $||f(y) - x||$, we can further refine y_0 by using it as an initial iterate to find the minimum of $||f(y) - x||$.

Filtering, smoothing, and calculating the evidence under the generative model can then be performed by standard techniques that do not require the derivatives of $\hat{g}$, such as the Unscented Kalman Filter [8].

5 Experiments

Empirically, the mapping we recover is close to the inverse of the true observation function. We compare the mapping recovered by our algorithm against KPCA and Isomap, since these performed the best out of the manifold learning algorithms we tried, and did not impose unreasonable storage requirements. All of our experiments use a Gaussian kernel.

Figure 1 shows the embedding of a 1500 step 2D random walk into $\mathcal{R}^3$ by the function $g(x, y) = (x, y\cos(2y), y\sin(2y))$. The 2D walk is bounded, and reflects off of the boundary. In addition to this behavior, the dynamics model used to generate the walk was different from the one specified in our algorithm, to demonstrate resilience to errors in the dynamics model. Using the recovered f, the figure plots $f(y)$ where y are gridded samples from the surface of the roll. The recovered f is close the original mapping. Varying the dynamics up to an order of magnitude on the parameters of the dynamics model yields similar results, but changes the scale of the mapping. Isomap performs poorly on this data set due to the low sampling rate on the manifold and the fact that the true mapping g is not isometric. KPCA chooses a linear projection that eliminates one of the dimensions, since this projection is smooth and the symmetry of the roll about the origin satisfies KPCA's constraint (5).

In Figure 2, we show that f accurately represents the inverse of the true g when applied to an image sequence. The figure shows a few frames of a 2000 frame sequence of a synthetically generated cube undergoing 2D rotations. To show resilience to mismatch between the true dynamics and the assumed dynamics, different dynamics were again used to generate the path of rotations than those specified to our algorithm, including reflections off of the boundary. The figure plots the coordinates X^* recovered by our algorithm. Because these coordinates are close to the true low-dimensional coordinates, the recovered f is close to the inverse of the true g.

Note that the mapping between rotations and appearances is not isometric, since infinitesimal rotations produce different amounts of change in appearance depending on the instantaneous rotation. This violation of Isomap's isometric assumption explains why its recovered poses are stretched in places. Both Isomap and KPCA pull together rotations that make the top face of the cube face the camera, because the appearance of the face is similar under all such poses. Use of dynamics allows our algorithm to disambiguate between these situations. KPCA also exhibits folding, which is absent from Isomap and our algorithm's output.

For a 2000 frame sequence, the algorithm runs in about 5 minutes on a 3 Ghz P4. We have found the following guidelines to be useful in setting the parameters of our system. To avoid numerical problems, the kernel width is chosen so that the smallest element of $\mathbf{K}$ is about 0.1. We use stable third order dynamics, with $A = \begin{bmatrix} 0.9 & 0.2 & 0 \\ 0 & 0.9 & 0.2 \\ 0 & 0 & 0.9 \end{bmatrix}$, and Σ a diagonal matrix with a few orders of magnitude more noise in the acceleration components than the velocity or position components.

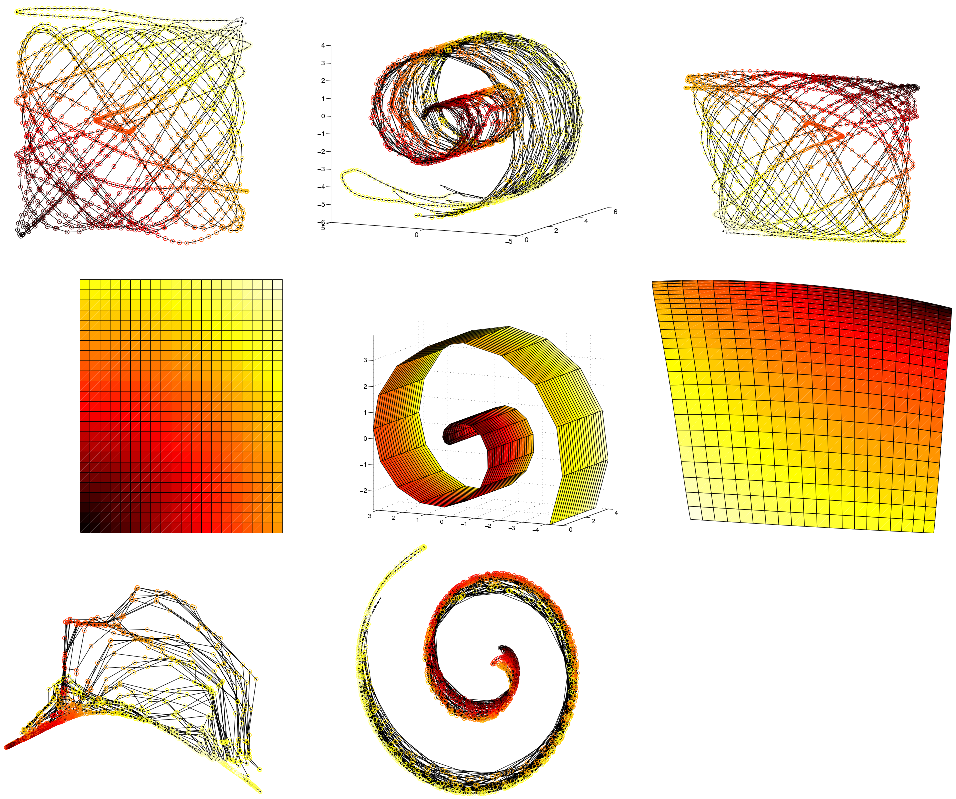
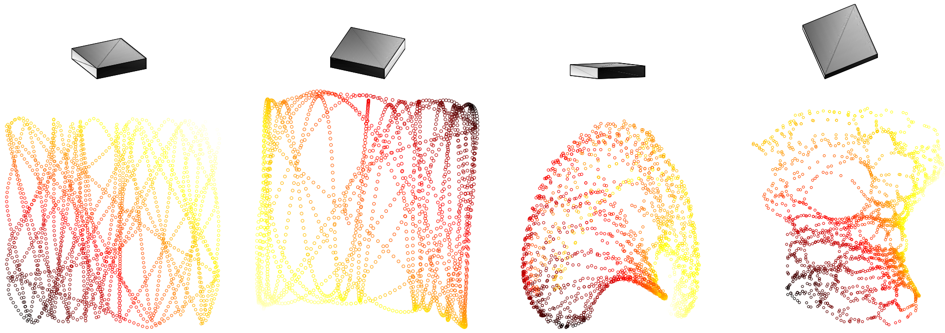


Figure 1: **(top-left)** Low-dimensional ground truth trajectory. Points are colored according to their distance from the origin in the parameter space. **(top-middle)** Embedding of the trajectory. **(top-right)** Recovered low-dimensional representation using our algorithm. The original data in **(top-left)** is correctly recovered. To further test the recovered function f, we uniformly sampled a 2D rectangle **(middle-left)**, lifted it using g **(middle-middle)**, and projected the result to 2D using the recovered f **(middle-right)**. f has correctly mapped the points near their original 2D location. Given only high-dimensional data, neither Isomap **(bottom-left)** nor KPCA **(bottom-right)** find the ground truth low-dimensional representation. These figures are best viewed in color.



(a) (b) (c) (d)

Figure 2: **(top row)** A few frames from the synthetically generated rotating cube sequence. Only the azimuth and elevation of the cube are modified. **(a)** Shows the true elevation-azimuth trajectory, **(b)** the trajectory $\mathbf{X}^*$ recovered by our algorithm, **(c)** by KPCA, and **(d)** by Isomap. Our algorithm recovers the true rotation up to a flip, but with very little distortion. Because appearance is not an isometric function of rotation, Isomap's trajectory is unevenly stretched.

6 Conclusions and Future Work

We have shown how to learn a nonlinear model for video that takes the dynamics of a latent representation for frames into account. Given a training video sequence and the approximate dynamics of the latent states, our algorithm learns a nonlinear relationship between observations and the latent variables.

To circumvent storage and computational problems, the nonlinear function maps observations to states, rather than the other way. We demonstrated that when the true underlying mapping from states to observations is one-to-one, the mapping recovered by our algorithm is close to the inverse of the true mapping.

In the future, we hope to isolate the conditions under which this convergence occurs. We intend to show how to estimate the parameters of the dynamics model as well.

The code and a version of this paper with derivations is available online on the first author's web site.

7 Acknowledgements

We would like to thank Matthew Beal and John Fisher III for helpful discussions.

References

[1] D.L. Donoho and C. Grimes. Hessian eigenmaps: new locally linear embedding techniques for highdimensional data. Technical report, TR2003-08, Dept. of Statistics, Stanford University, 2003.

[2] G. Doretto, A. Chiuso, and Y.N. Wu S. Soatto. Dynamic textures. *International Journal of Computer Vision (IJCV)*, 51(2):91–109, 2003.

[3] A. Elgammal and C.S. Lee. Inferring 3d body pose from silhouettes using activity manifold learning. In *Computer Vision and Pattern Recognition (CVPR)*, 2004.

[4] Z. Ghahramani and S. Roweis. Learning nonlinear dynamical systems using an em algorithm. In *Neural Information Processing Systems (NIPS)*, pages 431–437, 1998.

[5] A. Ilin, H. Valpola, and E. Oja. Nonlinear dynamical factor analysis for state change detection. *IEEE Transactions on Neural Networks*, 15(3):559–575, 2004.

[6] O. Jenkins and M. Mataric. A spatio-temporal extension to isomap nonlinear dimension reduction. In *International Conference on Machine Learning (ICML)*, 2004.

[7] A. Juditsky, H. Hjalmarsson, A. Benveniste, B. Delyon, L. Ljung, J. Sjöberg, and Q. Zhang. Nonlinear black-box models in system identification: Mathematical foundations. *Automatica*, 31(12):1725–1750, 1995.

[8] S. Julier and J. Uhlmann. A new extension of the kalman filter to nonlinear systems. In *Int. Symp. Aerospace/Defense Sensing, Simul. and Controls*, 1997.

[9] N. D. Lawrence. Gaussian process latent variable models for visualisation of high dimensional data. In *Advances in Neural Information Processing Systems (NIPS)*, 2004.

[10] J. Principe, D. Xu, and J. Fisher. *Unsupervised Adaptive Filtering*, chapter Information Theoretic Learning. Wiley, 1999.

[11] A. Rahimi, B. Recht, and T. Darrell. Learning appearance manifolds from video. In *Computer Vision and Pattern Recognition (CVPR)*, 2005.

[12] A. Rahimi, B. Recht, and T. Darrell. Nonlinear latent variable models for video sequences. Technical report, MIT CSAIL, 2005.

[13] B. Schölkopf, R. Herbrich, A.J. Smola, and R.C. Williamson. A generalized representer theorem. Technical Report 81, NeuroCOLT, 2000.

[14] B. Schölkopf, A. Smola, and K-R. Müller. Nonlinear component analysis as a kernel eigenvalue problem. *Neural Computation*, 10:1299–1319, 1998.

[15] A. Smola, S. Mika, B. Schoelkopf, and R. C. Williamson. Regularized principal manifolds. *Journal of Machine Learning*, 1:179–209, 2001.

[16] M. Szummer and R. W. Picard. Temporal texture modeling. In *International Conference on Image Processing (ICIP)*, pages 823–826, 1996.

[17] M. Pontil T. Evgeniou and T. Poggio. Regularization networks and support vector machines. *Advances in Computational Mathematics*, 2000.

[18] J. B. Tenenbaum, V. de Silva, and J. C. Langford. A global geometric framework for nonlinear dimensionality reduction. *Science*, 290(5500):2319–2323, 2000.

[19] H. Valpola. Nonlinear independent component analysis using ensemble learning theory. In *Proceedings of the International Workshop on Independent Component Analysis and Blind Signal Separation*, pages 351–356, 2000.

[20] H. Valpola and J. Karhunen. An unsupervised ensemble learning method for nonlinear dynamic state-space models. *Neural Computation*, 14(11):2647–2692, 2002.

A KPCA

Rewriting (6) as

$$\min_{\mathbf{C},\mathbf{X}} \sum_{i=1}^{d} \begin{bmatrix} \mathbf{X}_i \\ \mathbf{C}_i \end{bmatrix}^\top \begin{bmatrix} \mathbf{I} & -\mathbf{K} \\ -\mathbf{K} & \mathbf{K}^2 + \lambda\mathbf{K} \end{bmatrix} \begin{bmatrix} \mathbf{X}_i \\ \mathbf{C}_i \end{bmatrix} \tag{17}$$

$$\text{s.t. } \mathbf{X}\mathbf{X}^\top = \mathbf{I}, \tag{18}$$

where $\mathbf{C}_i$ is the transpose of the ith row of $\mathbf{C}$ and $\mathbf{X}_i$ is the transpose of the ith row of X, we may minimize over $\mathbf{C}$ to find

$$\mathbf{C}_i^* = (\mathbf{K} + \lambda\mathbf{I})^{-1}\mathbf{X}_i \tag{19}$$

plugging this optimal value back into the cost function yields a minimization only over X

$$\min_{\mathbf{X}} \sum_{i=1}^{d} \mathbf{X}_i^\top (\mathbf{K} + \lambda\mathbf{I})^{-1}\mathbf{X}_i \tag{20}$$

$$\text{s.t. } \mathbf{X}\mathbf{X}^\top = \mathbf{I}, \tag{21}$$

The optimal $\mathbf{X}_i^*$ are the d largest eigenvalues of the kernel $\mathbf{K}$. Let $\gamma_1, \ldots, \gamma_d$ be the largest eigenvalues of $\mathbf{K}$ and $v_1, \ldots, v_d$ be the corresponding orthonormal set eigenvectors. It follows that the optimal $\mathbf{C}_i^*$ are given by

$$\mathbf{C}_i^* = \frac{1}{\gamma_i + \lambda} v_i \tag{22}$$

Substituting this $\mathbf{C}_i^*$ into the representer form gives us a solution for f_i^*

$$f_i(y) = \frac{1}{\gamma_i + \lambda} \sum_{t=1}^{T} v_{it} k(y_t, y) \tag{23}$$

The kpca algorithm returns the functions $h_1, \ldots, h_d$ given by

$$h_i(y) = \frac{1}{\sqrt{\gamma_i}} \sum_{t=1}^{T} v_{it} k(y_t, y) \tag{24}$$

from which we see that the f_i^* are multiples of the solution to the kpca problem

$$f_i^* = \sqrt{\frac{\gamma_i}{(\gamma_i + \lambda)^2}}\ h_i\,. \tag{25}$$

B Learning with Dynamics

Applying the representer theorem, and keeping in mind that the dynamics decouple, the cost function can be rewritten as in Section **??** as

$$\min_{\mathbf{C},\mathbf{X}} \|\mathbf{HX} - \mathbf{CK}\|_F^2 + \mathbf{X}^\top \Omega \mathbf{X} + \lambda \mathbf{C}^\top \mathbf{K} \mathbf{C} \tag{26}$$

$$\text{s.t. } \mathbf{HXXH}^\top = T\sigma_\infty^2 \mathbf{I} \tag{27}$$

$$\mathbf{HX1}^\top = 0. \tag{28}$$

Letting $\mathbf{X}_i$ denote the rows of $\mathbf{X}$ corresponding to the state of the ith independent Markov chain stacked as a column vector and $\mathbf{C}_i$ denote the transpose of the ith row of C, we can rewrite this optimization as

$$\min_{\mathbf{C},\mathbf{X}} \sum_{i=1}^{d} \begin{bmatrix} \mathbf{X}_i \\ \mathbf{C}_i \end{bmatrix}^\top \begin{bmatrix} \hat{\mathbf{H}}^\top \hat{\mathbf{H}} + \Omega & -\mathbf{K} \\ -\mathbf{K} & \mathbf{K}^2 + \lambda \mathbf{K} \end{bmatrix} \begin{bmatrix} \mathbf{X}_i \\ \mathbf{C}_i \end{bmatrix} \tag{29}$$

$$\text{s.t. } \mathbf{HXX}^\top \mathbf{H}^\top = I \tag{30}$$

$$\mathbf{HX1}^\top = 0 \tag{31}$$

As before, minimizing over $\mathbf{C}$ yields

$$\mathbf{C}_i^* = (\mathbf{K} + \lambda \mathbf{I})^{-1} \hat{\mathbf{H}} \mathbf{X}_i \tag{32}$$

plugging this optimal value back into the cost function yields a minimization only over $\mathbf{X}$

$$\min_{\mathbf{X}} \sum_{i=1}^{d} \mathbf{X}_i^\top \left(\hat{\mathbf{H}}' (\mathbf{K} + \lambda \mathbf{I})^{-1} \hat{\mathbf{H}} + \Omega \right) \mathbf{X}_i \tag{33}$$

$$\text{s.t. } \mathbf{HXX}^\top \mathbf{H}^\top = T\sigma_\infty^2 \mathbf{I}, \tag{34}$$

$$\mathbf{HX1}^\top = 0 \tag{35}$$

Define a matrix $\mathbf{Z}$ with orthogonal rows to span the nullspace of $\hat{\mathbf{H}}\mathbf{1}$ so they span the space of solutions to the mean constraint (**??**). Then we may rewrite (**??**) as

$$\min_{\mathbf{X}} \sum_{i=1}^{d} \mathbf{X}_i^\top \mathbf{Z}^\top \left(\hat{\mathbf{H}}^\top (\mathbf{K} + \lambda \mathbf{I})^{-1} \hat{\mathbf{H}} + \Omega \right) \mathbf{Z} \mathbf{X}_i \tag{36}$$

$$\text{s.t. } \hat{\mathbf{H}} \mathbf{Z} \mathbf{X} \mathbf{X}^\top \mathbf{Z}^\top \hat{\mathbf{H}}^\top = T\sigma_\infty^2 \mathbf{I} \tag{37}$$

The optimal $\mathbf{C}^*$ and $\mathbf{X}^*$ pair can then be found by solving (11)-(16).