Learning Stable Linear Dynamical Systems

Byron Boots BEB@CS.CMU.EDU
Machine Learning Department
Carnegie Mellon University
Pittsburgh, PA 15217, USA

Abstract

Stability is a desirable characteristic for linear dynamical systems, but it is often ignored by algorithms that learn these systems from data. We propose a novel method for learning stable linear dynamical systems: we formulate an approximation of the problem as a convex program, start with a solution to a relaxed version of the program, and incrementally add constraints to improve stability. Rather than continuing to generate constraints until we reach a feasible solution, we test stability at each step; because the convex program is only an approximation of the desired problem, this early stopping rule can yield a higher-quality solution. We employ both maximum likelihood and subspace ID methods to the problem of learning dynamical systems with exogenous inputs directly from data. Our algorithm is applied to a variety of problems including the tasks of learning dynamic textures from image sequences, learning a model of laser and vision sensor data from a mobile robot, learning stable baseline models for drug-sales data in the biosurveillance domain, and learning a model to predict sunspot data over time. We compare the constraint generation approach to learning stable dynamical systems to the best previous stable algorithms (Lacy and Bernstein, 2002, 2003), with positive results in terms of prediction accuracy, quality of simulated sequences, and computational efficiency. Source code and video results of stable dynamic textures are available online at http://www.select.cs.cmu.edu/projects/stableLDS.

Keywords: Linear Dynamical Systems, Stability, Constraint Generation

1. Introduction

Many problems in machine learning involve sequences of real-valued multivariate observations. To model the statistical properties of such data, it is often sensible to assume each observation to be correlated to the value of an underlying latent variable, or *state*, that is evolving over the course of the sequence. In the case where the state is real-valued and the noise terms are assumed to be Gaussian, the resulting model is called a *linear dynamical system* (LDS), also known as a Kalman Filter (Kalman, 1960). LDSs are an important tool for modeling time series in engineering, controls and economics as well as the physical and social sciences.

Let $\{\lambda_i(M)\}_{i=1}^n$ denote the eigenvalues of an $n \times n$ matrix M in decreasing order of magnitude, $\{\nu_i(M)\}_{i=1}^n$ the corresponding unit-length eigenvectors, and define its *spectral radius* $\rho(M) \equiv |\lambda_1(M)|$. An LDS with dynamics matrix A is *internally stable* if all of A's eigenvalues have magnitude at most 1, i.e., $\rho(A) \leq 1$. Standard algorithms for learning LDS parameters do not enforce this stability criterion, learning locally optimal values for LDS parameters by gradient descent (Ljung, 1999), Expectation Maximization (EM) (Ghahra-

mani and Hinton, 1996) or least squares on a state sequence estimate obtained by subspace identification methods. However, when learning from finite data samples, all of these solutions may be unstable even if the system being modeled is stable (Chui and Maciejowski, 1996). The drawback of ignoring stability is most apparent when simulating or predicting long sequences from the system in order to generate representative data or infer stretches of missing values.

We propose a convex optimization algorithm for learning the dynamics matrix while guaranteeing stability when the estimate of the dynamical system is first obtained using either EM or subspace identification. We then formulate the least-squares problem for the dynamics matrix as a quadratic program (QP) (Boyd and Vandenberghe, 2004), initially without constraints. When this QP is solved, the estimate $\hat{A}$ obtained may be unstable. However, any unstable solution allows us to derive a linear constraint which we then add to our original QP and re-solve. The above two steps are iterated until we reach a stable solution, which is then refined by a simple interpolation to obtain the best possible stable estimate.

Our method can be viewed as *constraint generation* (Horst and Pardalos, 1995) for an underlying convex program with a feasible set of all matrices with singular values at most 1, similar to work in control systems (Lacy and Bernstein, 2002). However, we terminate *before* reaching feasibility in the convex program, by checking for matrix stability after each new constraint. This makes our algorithm less conservative than previous methods for enforcing stability since it chooses the best of a larger set of stable dynamics matrices. The difference in the resulting stable systems is apparent when simulating and predicting data. The constraint generation approach also achieves much greater efficiency than previous methods in our experiments.

One application of LDSs in computer vision is learning *dynamic textures* from video data (Soatto et al., 2001). An advantage of learning dynamic textures is the ability to play back a realistic-looking generated sequence of any desired duration. In practice, however, videos synthesized from dynamic texture models can quickly degenerate because of instability in the underlying LDS. In contrast, sequences generated from dynamic textures learned by our method remain "sane" even after arbitrarily long durations. We also apply our algorithm to learning models of robot sense data conditioned on odometry information, baseline dynamic models of over-the-counter (OTC) drug sales for biosurveillance, and sunspot data from the UCR archive (Keogh and Folias, 2002). Comparison to the best alternative methods (Lacy and Bernstein, 2002, 2003) on these problems consistently yields positive results.

2. Linear Dynamical Systems

The evolution of a stochastic linear time-invariant dynamical system (LDS) can be described by the following two equations:

$$x_{t+1} = Ax_t + w_t \qquad w_t \sim \mathcal{N}(0, Q) \tag{1a}$$

$$y_t = Cx_t + v_t \qquad v_t \sim \mathcal{N}(0, R) \tag{1b}$$

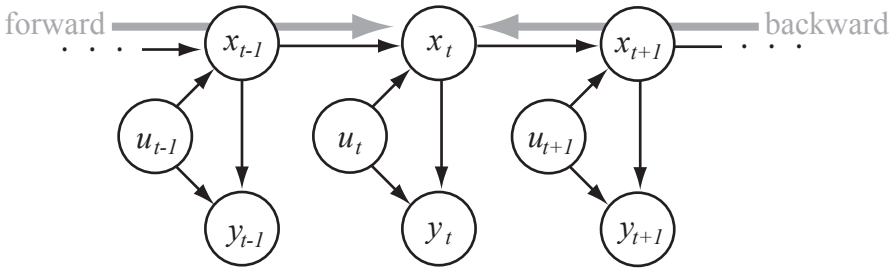

Figure 1: Graphical representation of the deterministic-stochastic linear dynamical system. The larger grey arrows indicate the forward and backward messages passed during inference. See text for details.

Time is indexed by the discrete[1] variable t. Here x_t denotes the hidden states in $\mathbb{R}^n$, y_t the observations in $\mathbb{R}^m$, and the parameters of the system: the dynamics matrix $A \in \mathbb{R}^{n \times n}$ and the observation model $C \in \mathbb{R}^{m \times n}$. The variables w_t and v_t describe zero-mean normally distributed process and observation noise respectively, with covariance and cross-covariance matrices

$$\mathbb{E}\left\{ \begin{bmatrix} w_t \\ v_t \end{bmatrix} \begin{bmatrix} w_s^{\mathrm{T}} & v_s^{\mathrm{T}} \end{bmatrix} \right\} = \begin{bmatrix} Q & S \\ S^{\mathrm{T}} & R \end{bmatrix} \delta_{ts} \tag{2}$$

where δ_{ts} is the Kronecker delta, $Q \in \mathbb{R}^{n \times n}$ is non-negative definite, and $R \in \mathbb{R}^{m \times m}$ is positive definite, and $S \in \mathbb{R}^{n \times m}$ is the cross-covariance. Inputs can be incorporated into the LDS model via a modification of Equations 1 resulting in a deterministic-stochastic LDS

$$x_{t+1} = Ax_t + Bu_t + w_t \qquad w_t \sim \mathcal{N}(0, Q) \tag{3a}$$
$$y_t = Cx_t + Du_t + v_t \qquad v_t \sim \mathcal{N}(0, R) \tag{3b}$$

where u_t denotes an exogenous input in $\mathbb{R}^l$ at time t and $B \in \mathbb{R}^{n \times l}$ and $D \in \mathbb{R}^{m \times l}$ are parameters that govern the effect of the inputs on the dynamical system. Thus, a stochastic LDS (Equations 1a-b) models the distribution of outputs $P(y_{1:T})$, while the deterministic-stochastic LDS (Equations 3a-b and Figure 1) models the *conditional* distribution of outputs given deterministic inputs $P(y_{1:T}|u_{1:T})$. For the remainder of this paper we will consider LDSs with exogenous inputs (also called deterministic-stochastic LDSs), a model that includes stochastic LDSs as a special case (when the inputs are constant over time).

1. In *continuous-time* dynamical systems, the derivatives are specified as functions of the current state. They can be converted to discrete-time systems. If we could be guaranteed that the system we're trying to recover is *positive real*, we could use an exact method due to (Hoagg et al., 2004), but there's no reason to expect positive realness in many cases of interest.

2.1 Inference

In this section we describe the forwards and backwards inference algorithms for LDS. More details can be found in several sources (Ljung, 1999; Van Overschee and De Moor, 1996; Katayama, 2005).

The distribution over state at time t, $P(X_t|y_{1:T}, u_{1:T})$, can be exactly computed in two parts: a forward and a backward recursive pass. The forward pass which is dependent on the initial state x_0 and the observations $y_{1:t}$, is known as the *Kalman filter*, and the backward pass which uses the observations from y_T to y_{t+1} is known as the *Rauch-Tung-Striebel* (RTS) equations. The combined forward and backward passes are together called the *Kalman smoother*. It is worth noting that the standard LDS filtering and smoothing inference algorithms (Kalman, 1960; Rauch, 1963) are instantiations of the junction tree algorithm for Bayesian Networks on the dynamic Baysian network described in Figure 1 (see, for example, Murphy (2002)).

2.1.1 The Forward Pass (Kalman Filter)

Let the mean and covariance of the belief state estimate $P(X_t|y_{1:t}, u_{1:t})$ at time t be denoted by $\hat{x}_t$ and $\hat{P}_t$ respectively. The estimates $\hat{x}_t$ and $\hat{P}_t$ can be predicted from the previous time step, the exogenous input, and the previous observation. Let $\hat{x}_{t_1|t_2}$ denote an estimate of variable x at time t_1 given data $y_1, \ldots, y_{t_2}$. We then have the following recursive equations:

$$x_{t|t-1} = Ax_{t-1|t-1} + Bu_t \tag{4a}$$

$$P_{t|t-1} = AP_{t-1|t-1}A^{\mathrm{T}} + Q \tag{4b}$$

Equation 4a can be thought of as applying the system matrices A and B and exogenous input u_{t-1} to the mean to form an initial prediction of $\hat{x}_t$. Similarly, Equation 4b can be interpreted as using the dynamics matrix A and error covariance Q to form an initial estimate of the belief covariance $\hat{P}_t$. The estimates are then adjusted:

$$x_{t|t} = x_{t|t-1} + K_t e_t \tag{4c}$$

$$P_{t|t} = P_{t|t-1} - K_t C P_{t|t-1} \tag{4d}$$

where the error in prediction at the previous time step (the innovation) e_{t-1} and the Kalman gain matrix K_{t-1} are computed as follows:

$$e_{t-1} = y_{t-1} - (C\hat{x}_{t-1|t-1} + Du_{t-1}) \tag{4e}$$

$$K_{t-1} = P_{t-1|t-1}C^{\mathrm{T}}(C\hat{P}_{t-1|t-1}C^{\mathrm{T}} + R)^{-1} \tag{4f}$$

The weighted error in Equation 4c corrects the predicted mean given an observation, and Equation 4d reduces the variance of the belief by an amount proportional to the observation covariance. Taken together, Equations 4a-f define a specific form of the Kalman filter known as the *forward innovation model*.

2.1.2 The Backward Pass (RTS Equations)

The forward pass finds the mean and variance of the states x_t, conditioned on *past* observations. The backward pass corrects the results of the forward pass by evaluating the influence

of *future* observations on these estimates. Once the forward recursion has completed and the final values of the mean and variance $x_{T|T}$ and $P_{T|T}$ have been calculated, the backward pass proceeds in reverse by evaluating the influence of future observations on the states in the past:

$$x_{t|T} = x_{t|t} + G_t(x_{t+1|T} - x_{t+1|t}) \tag{5a}$$
$$P_{t|T} = P_{t|t} + G_t(P_{t+1|T} - P_{t+1|t})G_t^{\mathrm{T}} \tag{5b}$$

where $x_{t+1|t}$ and $P_{t+1|t}$ are 1-step predictions

$$x_{t+1|t} = Ax_{t|t} + Bu_{t+1} \tag{5c}$$
$$P_{t+1|t} = AP_{t|t}A^{\mathrm{T}} + Q \tag{5d}$$

and the smoother gain matrix G is computed as:

$$G_t = P_{t|t}A^{\mathrm{T}}P_{t+1|t}^{-1} \tag{5e}$$

The cross variance $P_{t,t-1|T} = \mathrm{Cov}[X_{t-1}, X_t|y_{1:T}]$, a useful quantity for parameter estimation (section 3.1), may also be computed at this point:

$$P_{t-1,t|T} = G_{t-1}P_{t|T} \tag{5f}$$

3. Learning Linear Dynamical Systems

Learning a dynamical system from data (system identification) involves finding the parameters $\theta = \{A, B, C, D, Q, R\}$ and the distribution over hidden variables $\mathcal{Q} = P(X|Y, \theta)$ that maximize the likelihood of the observed data. The maximum likelihood solution for these parameters can be found through iterative techniques such as expectation maximization (EM). An alternative approach is to use *subspace identification* methods to compute an asymptotically unbiased solution in closed form. In practice, a good approach is to use subspace identification to find an initial solution and then refine the solution with EM. The EM algorithm for system identification is presented in section 3.1 and subspace identification is presented in section 3.2.

3.1 Expectation Maximization

The EM algorithm is an iterative procedure for finding parameters that maximize the likelihood of observed data $P(Y|\theta)$ in the presence of latent variables x. In practice, instead of maximizing the likelihood directly, a lower bound on the log-likelihood

$$\mathcal{L}(\theta) = \log P(Y|\theta) = \log \int_X P(X, Y|\theta)dX \tag{6}$$

is maximized by coordinate ascent [2]. Using any distribution over the hidden variables $\mathcal{Q}$, a lower bound on the log-likelihood $\mathcal{F}(\mathcal{Q}, \theta) \leq \mathcal{L}(\theta)$ can be obtained by utilizing Jensen's

2. For LDSs, this lower bound is tight and EM maximizes the likelihood. See Section 3.1.1

inequality (at Equation 7b, below):

$$\mathcal{L}(\theta) = \log P(Y|\theta) = \log \int_X P(X, Y|\theta) dX \tag{7a}$$

$$= \log \int_X \mathcal{Q}(X) \frac{P(X, Y|\theta)}{\mathcal{Q}(X)} dX \geq \int_X \mathcal{Q}(X) \log \frac{P(X, Y|\theta)}{\mathcal{Q}(X)} dx \tag{7b}$$

$$= \int_X \mathcal{Q}(X) \log P(X, Y|\theta) dX - \int_X \mathcal{Q}(X) \log \mathcal{Q}(X) dx \tag{7c}$$

$$= \mathcal{F}(\mathcal{Q}, \theta) \tag{7d}$$

The EM algorithm alternates between maximizing the lower-bound on the log-likelihood $\mathcal{F}$ with respect to the parameters θ and with respect to the distribution $\mathcal{Q}$, holding the other quantity fixed. Thus, starting from an initial estimate of the parameters θ_0, we alternately apply:

$$\text{Expectation-step (E-step): } \mathcal{Q}_{k+1} \leftarrow \arg\max_{\mathcal{Q}} \mathcal{F}(\mathcal{Q}, \theta_k) \tag{8a}$$

$$\text{Maximization-step (M-step): } \theta_{k+1} \leftarrow \arg\max_{\theta} \mathcal{F}(\mathcal{Q}_{k+1}, \theta) \tag{8b}$$

where k indexes an iteration, until the parameter estimate θ_k converges to a local maximum.

3.1.1 The E-Step

The E-step (Equation 8a) is maximized when $\mathcal{Q}$ is exactly the conditional distribution of X, that is $\mathcal{Q}_{k+1}(X) = P(X|Y, \theta_k)$, at which point the bound becomes an equality: $\mathcal{F}(\mathcal{Q}_{k+1}, \theta_k) = \mathcal{L}(\theta)$. Fortunately, we have already seen how the maximum value of $P(X|Y, \theta_k)$ can be computed exactly by solving the LDS inference (Kalman smoothing) problem: estimating the hidden state trajectory given the inputs, the outputs, and the parameter values. This algorithm is outlined in section 2.1.

3.1.2 The M-step

As noted in Equation 8b, the M-step is maximized by finding the maximum of $\mathcal{F}(\mathcal{Q}_{k+1}, \theta) = \int_X \mathcal{Q}_{k+1}(X) \log P(X, Y|\theta) dX - \int_X \mathcal{Q}_{k+1}(X) \log \mathcal{Q}_{k+1}(X) dx$ with respect to θ. The parameters of the system $\theta_{k+1} = \{\hat{A}, \hat{B}, \hat{C}, \hat{D}, \hat{Q}, \hat{R}\}$ are estimated by taking the corresponding partial derivative of the expected log-likelihood, setting to zero and solving, resulting in the

following update equations:

$$[\ \hat{C} \quad \hat{D}\ ] = \left(\sum_{t=1}^{T} y_t \mathbb{E}\{w_t^{\mathrm{T}} | y_{1:T}\}\right)\left(\sum_{t=1}^{T} \mathbb{E}\{w_t w_t^{\mathrm{T}} | y_{1:T}\}\right)^{-1} \tag{9a}$$

$$\hat{R} = \frac{1}{T}\left(\sum_{t=1}^{T} y_t y_t^{\mathrm{T}} - [\ \hat{C} \quad \hat{D}\ ] \sum_{t=1}^{T} \mathbb{E}\{w_t | y_{1:T}\} y_t^{\mathrm{T}}\right) \tag{9b}$$

$$[\ \hat{A} \quad \hat{B}\ ] = \left(\sum_{t=2}^{T} \mathbb{E}\{x_t w_{t-1}^{\mathrm{T}} | y_{1:T}\}\right)\left(\sum_{t=2}^{T} \mathbb{E}\{w_{t-1} w_{t-1}^{\mathrm{T}} | y_{1:T}\}\right)^{-1} \tag{9c}$$

$$\hat{Q} = \frac{1}{T-1}\left(\sum_{t=2}^{T} \mathbb{E}\{x_t x_t^{\mathrm{T}} | y_{1:T}\} - [\ \hat{A} \quad \hat{B}\ ] \sum_{t=2}^{T} \mathbb{E}\{w_{t-1} x_t^{\mathrm{T}} | y_{1:T}\}\right) \tag{9d}$$

where $w_t = \begin{bmatrix} x_t \\ u_t \end{bmatrix}$.

3.2 Subspace Identification

Subspace methods calculate the parameters of an LDS by using tools from linear algebra including the oblique projection (explained below) and the *singular value decomposition* (SVD) (Horn and Johnson, 1985) to find Kalman filter estimates of the underlying state sequence in closed form. See Van Overschee and De Moor (1996) for variations. This approach is often advantageous with respect to EM in practice when the state space is high dimensional.

Let $U_{0|i-1}$ and $Y_{0|i-1}$ be defined as:

$$U_{0|i-1} = \begin{bmatrix} u_0 & u_1 & \cdots & u_{j-1} \\ u_1 & u_2 & \cdots & u_j \\ \vdots & \vdots & \ddots & \vdots \\ u_{i-1} & u_i & \cdots & u_{i+j-2} \end{bmatrix}_{li \times j} \qquad Y_{0|i-1} = \begin{bmatrix} y_0 & y_1 & \cdots & y_{j-1} \\ y_1 & y_2 & \cdots & y_j \\ \vdots & \vdots & \ddots & \vdots \\ y_{i-1} & y_i & \cdots & y_{i+j-2} \end{bmatrix}_{mi \times j} \tag{10}$$

We will use U_p and Y_p to denote certain matrices of "past" inputs and observations respectively, U_p^+, Y_p^+ to denote one-timestep *extensions* of these matrices, and i to denote the "present." Specifically, we define

$$U_p \equiv U_{0|i-1} \quad U_p^+ \equiv U_{0|i}$$
$$Y_p \equiv Y_{0|i-1} \quad Y_p^+ \equiv Y_{0|i}$$

These will be useful in the subsequent discussion. Similarly, let U_f, Y_f, U_f^-, Y_f^- denote matrices of "future" inputs and observations and their one-step *contractions*. These are defined as

$$U_f \equiv U_{i|2i-1} \quad U_f^- \equiv U_{i+1|2i-1}$$
$$Y_f \equiv Y_{i|2i-1} \quad Y_f^- \equiv Y_{i+1|2i-1}$$

Column t of Y_f or U_f represents the future observation or exogenous input at time $t+i-1$, and the corresponding column of Y_p or U_p represents the past observation or exogenous input at time $t+i-1$. Matrices of the above form, with each block of rows equal to the previous block but shifted by a constant number of columns, are called *block Hankel* matrices (Ljung, 1999). Finally, let

$$\hat{X}_i = [\hat{x}_i \ \hat{x}_{i+1} \ \dots \ \hat{x}_{i+j}] \in \mathbb{R}^{n \times j} \tag{11}$$

be a set of Kalman filter state estimates at time i derived from the same set of observations. From Equations 10 and 11 and the equation for the Kalman filter (Equation 4a-f), the equations for updating the predictive state estimate for a set of Kalman filters in parallel are:

$$\hat{X}_{i+1} = A\hat{X}_i + BU_{i|i} + K_i E_i \tag{12a}$$

$$Y_{i|i} = C\hat{X}_i + DU_{i|i} + E_i \tag{12b}$$

$$E_i = Y_{i|i} - C\hat{X}_i - DU_{i|i} \tag{12c}$$

where $E_i \in \mathbb{R}^{n \times j}$ contains the Kalman filter innovations. The subspace identification algorithm assumes that the innovations are uncorrelated with the predictive state estimates $\hat{X}_i$, the past inputs U_p and past outputs Y_p. Thus, if the observations truly arise from an LDS, then $\mathbb{E}\{E_i | Y_p, U_p, U_f, \hat{X}_i\} = 0$ and

$$\begin{aligned}
\mathbb{E}\{Y_f | \hat{X}_i, U_f\} &= \begin{bmatrix} C\hat{x}_i + Du_i & C\hat{x}_{i+1} + Du_{i+1} & \cdots & C\hat{x}_{j-1} + Du_{j-1} \\ C\hat{x}_{i+1} + Du_{i+1} & C\hat{x}_{i+2} + Du_{i+2} & \cdots & C\hat{x}_j + Du_j \\ C\hat{x}_{i+2} + Du_{i+2} & C\hat{x}_{i+3} + Du_{i+3} & \cdots & C\hat{x}_{j+1} + Du_{j+1} \\ \vdots & \vdots & \ddots & \vdots \\ C\hat{x}_{2i-1} + Du_{2i-1} & C\hat{x}_{2i} + Du_{2i} & \dots & C\hat{x}_{2i+j-2} + Du_{2i+j-2} \end{bmatrix}_{mi \times j} \\
&= \begin{bmatrix} C\hat{x}_i + Du_i & \cdots \\ CA\hat{x}_i + CBu_i + Du_{i+1} & \cdots \\ CA^2\hat{x}_i + CABu_i + CBu_{i+1} + Du_{i+2} & \cdots \\ CA^3\hat{x}_i + CA^2Bu_i + CABu_{i+1} + CBu_{i+2} + Du_{i+3} & \cdots \\ \vdots & \ddots \end{bmatrix}_{mi \times j}
\end{aligned} \tag{13}$$

Let Γ_i (the extended observability matrix) and the lower block triangular Toeplitz matrix H_i be defined as:

$$\Gamma_i = \begin{bmatrix} C \\ CA \\ CA^2 \\ \vdots \\ CA^{i-1} \end{bmatrix}_{mi \times n} \qquad H_i = \begin{bmatrix} D & 0 & \cdots & 0 \\ CB & D & \cdots & 0 \\ CAB & CB & \cdots & 0 \\ \vdots & \vdots & & \vdots \\ CA^{i-2}B & CA^{i-3}B & \cdots & D \end{bmatrix}_{mi \times li} \tag{14}$$

Note that Γ_i and Γ_{i-1} are related by the expression

$$\Gamma_i = \begin{bmatrix} \Gamma_{i-1} \\ CA^{i-1} \end{bmatrix} \tag{15}$$

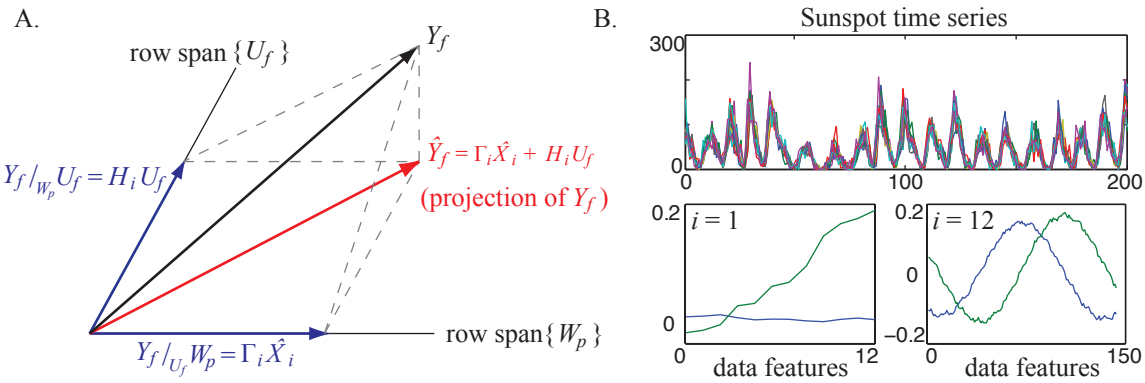

Figure 2: (A): Oblique projection. Y_f is the Hankel matrix of future observations, U_f is the Hankel matrix of future exogenous inputs, W_p is the matrix of past observations and inputs, and $\hat{Y}_f$ is the projection of Y_f onto W_p and Uf. The oblique projection of Y_f *along* U_f and *onto* W_p is the vector along the row span of W_p. (B): Sunspot data, sampled monthly for 200 years. Each curve is a month, the x-axis is over years. Below the graph are the first two principal components of $\mathcal{O}_i$ where Y_f and Y_p each consist of 1-observation Hankel matrices and 12-observation Hankel matrices. The 1-observation Hankel matrices do not contain enough observations to recover a state which accurately reflects the temporal patterns in the data, while the 12-observation Hankel matrices do.

By substituting Γ_i and H_i from Equation 14 and U_f into Equation 13, we see that $\mathbb{E}\{Y_f|\hat{X}_i, U_f\}$ is decomposed into a sum of two products of low-rank matrices:

$$\mathbb{E}\{Y_f|\hat{X}_i, U_f\} = \Gamma_i \hat{X}_i + H_i U_f \tag{16}$$

where $H_i U_f$ is a linear function of future inputs that lies in the row span of U_f and $\Gamma_i \hat{X}_i$ is a rank n linear function of state that lies in the row span of W_p, where $W_p = \begin{bmatrix} Y_p \\ U_p \end{bmatrix}$ [3].

Given the model in Equation 16, the *oblique projection* (Van Overschee and De Moor, 1996) of Y_f *along* U_f and *onto* W_p, denoted $Y_f/_{U_f} W_p$, may be used to find $\Gamma_i \hat{X}$ directly from data matrices Y_f, U_f and W_p (Figure 2). Here, $\hat{X}_i$ is the *true* Kaman filter state estimate starting from state $\hat{X}_0 = 0$, incorporating inputs U_p, and filtering on observations Y_p [4]. The oblique projection is calculated

$$Y_f/_{U_f} W_p = Y_f \begin{bmatrix} W_p \\ U_f \end{bmatrix}^\dagger \begin{bmatrix} W_p \\ 0_{li \times j} \end{bmatrix} \tag{17}$$

3. The Kalman filter state estimates $\hat{X}_i$ can be computed exactly in closed form as a linear function of Y_p and U_p. The proof can be found in Van Overschee and De Moor (1996)
4. The proof of this fact can be found in Van Overschee and De Moor (1996)

where † denotes the Moore-Penrose pseudo-inverse and $0_{li\times j}$ is a matrix of zeros $\in \mathbb{R}^{li\times j}$. When there are no exogenous inputs, the oblique projection reduces to a regular orthogonal projection. Let

$$\mathcal{O}_i = Y_f /_{U_f} W_p = \Gamma_i \hat{X}_i \tag{18a}$$

$$\mathcal{O}_{i+1} = Y_f^- /_{U_f^-} W_p^+ = \Gamma_{i-1} \hat{X}_{i+1} \tag{18b}$$

where $W_p^+ = \begin{bmatrix} Y_p^+ \\ U_p^+ \end{bmatrix}$. The rank of $\mathcal{O}_i$ is the dimensionality of the state space, the row space of $\mathcal{O}_i$ is equal to the row space of Γ_i and the column space of $\mathcal{O}_i$ is equal to the column space of $\hat{X}_i$ (Van Overschee and De Moor, 1996). These properties can be exploited to find estimates of $\hat{X}_i$ and $\hat{X}_{i+1}$. Let

$$\mathcal{O}_i = \mathcal{U}\Sigma\mathcal{V}^{\mathrm{T}} \tag{19}$$

be the singular value decomposition of the oblique projection. The order of the system can be determined by inspecting the singular values in Σ. Near-zero singular values are deleted from Σ, as are the corresponding singular vectors in $\mathcal{U}$ and $\mathcal{V}^{\mathrm{T}}$. The SVD will choose the columns of $\mathcal{U}$ to be an optimal basis for compressing and reconstructing sequences of expected future observations. Thus, Γ_i and Γ_{i-1} and the Kalman filter state sequences are estimated, up to a linear transform, as:

$$\Gamma_i = \mathcal{U}\Sigma^{1/2} \tag{20}$$

which allows us to estimate Γ_{i-1} using Eq (15), and

$$\tilde{X}_i = \Gamma_i^\dagger \mathcal{O}_i \tag{21a}$$

$$\tilde{X}_{i+1} = \Gamma_{i-1}^\dagger \mathcal{O}_{i+1} \tag{21b}$$

An important result is that, if the number of observations, i, in Y_f is large enough, as the number of columns in our Hankel matrices $j \to \infty$ the state estimates recovered by SVD converge to the *true* Kalman filter state estimates $\tilde{X}_i \to \hat{X}_i$ and $\tilde{X}_{i+1} \to \hat{X}_{i+1}$ up to a linear transform (Van Overschee and De Moor, 1996).

Having multiple observations per column in Y_f is particularly important when the underlying dynamical system is not completely observable. For example, Figure 2(B) shows 200 years of sunspot numbers, with each month modeled as a separate variable. Sunspots are known to have two periods, the longer of which is 11 years. When subspace ID is performed using Hankel matrices with $i = 12$, the first two principal components of $\mathcal{O}_i$ resemble the sine and cosine bases, and the corresponding state variables are the coefficients needed to combine these bases so as to predict 12 years of data. This is in contrast to the bases obtained by SVD on a 1-observation Y_f and Y_p, which reconstruct just the variation within a single year. Thus, with $i = 1$ the state estimate *will not* converge to the true Kalman filter state estimate even if $j \to \infty$.

Once $\tilde{X}_i$ and $\tilde{X}_{i+1}$ have been determined, the parameters can be found by solving the following set of linear equations for A, B, C and D:

$$\begin{bmatrix} \hat{X}_{i+1} \\ Y_{i|i} \end{bmatrix} = \begin{bmatrix} A & B \\ C & D \end{bmatrix} \begin{bmatrix} \hat{X}_i \\ U_{i|i} \end{bmatrix} + \begin{bmatrix} \rho_w \\ \rho_v \end{bmatrix} \tag{22a}$$

The covariance matrices $\hat{Q}$ and $\hat{R}$ can be estimated directly from residuals:

$$\begin{bmatrix} \hat{Q} & \hat{S} \\ \hat{S}^{\mathrm{T}} & \hat{R} \end{bmatrix} = \frac{1}{T-1} \left(\begin{bmatrix} \rho_w \\ \rho_v \end{bmatrix} \begin{bmatrix} \rho_w^{\mathrm{T}} \rho_v^{\mathrm{T}} \end{bmatrix} \right) \tag{22b}$$

Due to the fact that $\tilde{X}_i \to \hat{X}_i$ and $\tilde{X}_{i+1} \to \hat{X}_{i+1}$, the parameter estimates $\theta = \{\hat{A}, \hat{B}, \hat{C}, \hat{D}, \hat{Q}, \hat{R}\}$ are consistent as $j \to \infty$ (Van Overschee and De Moor, 1996).

The maximum likelihood solution found by EM might provide more sensible parameter estimates than subspace ID, especially when the amount of training data is small, but is subject to local optima. One reasonable approach to parameter estimation is to first estimate the parameters with subspace ID and then refine the solution by using the parameter estimates as a starting point for the EM algorithm. This approach is explored in Section 6.

4. Stability

Stability is a property of dynamical systems defined in terms of *equilibrium points*. If all solutions of a dynamical system that start out near an equilibrium state x_e stay near or converge to x_e, then the state x_e is stable or asymptotically stable respectively (see Figure 4). A linear system $x_t = Ax_t + Bu_t$ is *internally stable* if the internal states are stable, i.e. if the matrix A obeys the Lyapunov stability criterion (see below). Internal stability is sufficient, though not strictly necessary, for the stability of a dynamical system with exogenous inputs (Halanay and Rasvan, 2000). The standard algorithms for learning linear Gaussian systems described in Section 3 *do not enforce stability*; when learning from finite data samples, the maximum likelihood or subspace ID solution may be unstable even if the true system is stable due to the sampling constraints, modeling errors, and measurement noise.

A dynamics matrix A is said to be asymptotically stable in the sense of Lyapunov if, for any given positive semi-definite symmetric matrix Q (interpreted as an observation covariance), there exists a positive-definite symmetric matrix P (interpreted as a steady-state belief covariance) that satises the following Lyapunov criterion:

$$P - APA^{\mathrm{T}} = Q \tag{23}$$

An LDS is said to be asymptotically stable in the sense of Lyapunov if its dynamics matrix A is. Thus, the Lyapunov criterion can be interpreted as holding for an LDS if, for a given covariance matrix, there exists a belief distribution where the predicted belief over state is equivalent to the previous belief over state, that is, if there exists an equilibrium point.

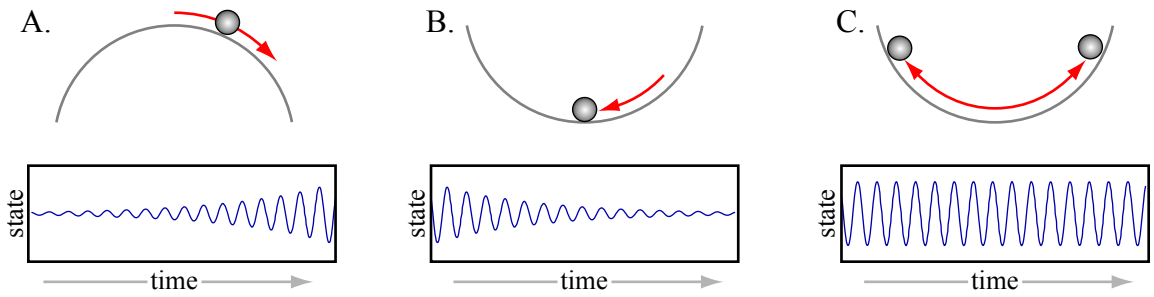

Figure 3: System equilibria. (A) Unstable equilibrium. The state vector will rapidly move away from the equilibrium point when perturbed. (B) Asymptotically stable equilibrium. The state vector will return to the original equilibrium point when perturbed. (C.) Stable equilibrium. No "resistance;" a perturbed state vector will oscillate forever around the equilibrium point. Note that the notion of *asymptotic stability* is stronger than *stability*.

It is interesting to note that the Lyapunov criterion holds *iff* the spectral radius $\rho(A) \leq 1$. Recall that a matrix M is positive semi-definite *iff* $zMz^{\mathrm{T}} \geq 0$ for all non-zero vectors z. Let λ be an left eigenvalue of A and ν be a corresponding eigenvector, giving us $\nu A = \nu\lambda$, then

$$\nu Q \nu^{\mathrm{T}} = \nu(P - A^{\mathrm{T}} P A)\nu^{\mathrm{T}} = \nu P \nu^{\mathrm{T}} - \nu\lambda P \lambda \nu^{\mathrm{T}} = \nu P \nu^{\mathrm{T}}(1 - |\lambda|^2) \tag{24}$$

since $\nu P \nu^{\mathrm{T}} \geq 0$, it follows that $|\lambda| \leq 1$ is equivalent to $\nu Q \nu^{T} \geq 0$, and therefore to Equation 24. When $\rho(A) < 1$, the system is asymptotically stable. To see this, suppose $D\Lambda D^{-1}$ is the eigen-decomposition of A, where Λ has the eigenvalues of A along the diagonal and D contains the eigenvectors. Then,

$$\lim_{k\to\infty} A^k = \lim_{k\to\infty} D\Lambda^k D^{-1} = D\left(\lim_{k\to\infty} \Lambda^k\right) D^{-1} = 0 \tag{25}$$

since it is clear that $\lim_{k\to\infty} \Lambda^k = 0$. If $\Lambda = I$, then A is stable but not asymptotically stable and the state oscillates around x_e indefinitely.

5. Learning Stable Linear Dynamical Systems

The estimation procedures in Section 3 does not enforce stability in $\hat{A}$ which can cause problems when predicting and simulating from an LDS learned from data. To account for stability, we first formulate the dynamics matrix learning problem as a quadratic program with a feasible set that includes the set of stable dynamics matrices. Then we demonstrate how instability in its solutions can be used to generate constraints that restrict this feasible set appropriately. As a final step, the solution is refined to be as close as possible to the least-squares estimate while remaining stable. The overall algorithm is illustrated in Figure 4(A). We now explain the algorithm in more detail.

5.1 Formulating the Objective

In subspace ID as well as in the M-step of an iteration of EM, it is possible to write the objective function for $\hat{A}$ as a quadratic function. For subspace ID we define a quadratic objective function using the residuals: $X_R = \tilde{X}_{i+1} - \hat{B}\tilde{U}_{i|i}$.

$$\begin{aligned}
\hat{A} &= \arg\min_A \left\| A\tilde{X}_i - X_R \right\|_F^2 \\
&= \arg\min_A \left\{ \mathrm{tr}\left[\left(A\tilde{X}_i - X_R \right)^{\mathrm{T}} \left(A\tilde{X}_i - X_R \right) \right] \right\} \\
&= \arg\min_A \left\{ \mathrm{tr}\left(A\tilde{X}_i\tilde{X}_i^{\mathrm{T}}A^{\mathrm{T}} \right) - 2\mathrm{tr}\left(\tilde{X}_i X_R^{\mathrm{T}} A \right) + \mathrm{tr}\left(X_R^{\mathrm{T}} X_R \right) \right\} \\
&= \arg\min_a \left\{ a^{\mathrm{T}} P a - 2\, q^{\mathrm{T}} a + r \right\} \qquad (26a)
\end{aligned}$$

where $a \in \mathbb{R}^{n^2\times 1}$, $q \in \mathbb{R}^{n^2\times 1}$, $P \in \mathbb{R}^{n^2\times n^2}$ and $r \in \mathbb{R}$ are defined as:

$$a = \mathrm{vec}(A) = [A_{11}\ A_{21}\ A_{31}\ \cdots\ A_{nn}]^{\mathrm{T}} \qquad (26b)$$

$$P = I_n \otimes \left(\tilde{X}_i \tilde{X}_i^{\mathrm{T}} \right) \qquad (26c)$$

$$q = \mathrm{vec}(X_i X_R^{\mathrm{T}}) \qquad (26d)$$

$$r = \mathrm{tr}\left(X_R^{\mathrm{T}} X_R \right) \qquad (26e)$$

I_n is the $n \times n$ identity matrix and $\otimes$ denotes the Kronecker product. Note that P (which is defined differently from the P in Section 4) is a symmetric positive semi-definite matrix and the objective function in Equation 27a is a quadratic function of a.

For EM, we use the same function form (Equation 27a), but with:

$$\hat{A} = \arg\min_a \left\{ a^{\mathrm{T}} P a - 2\, q^{\mathrm{T}} a \right\} \qquad (27a)$$

where $a \in \mathbb{R}^{n^2\times 1}$, $q \in \mathbb{R}^{n^2\times 1}$ and $P \in \mathbb{R}^{n^2\times n^2}$ are defined as:

$$a = \mathrm{vec}(A) = [A_{11}\ A_{21}\ A_{31}\ \cdots\ A_{nn}]^{\mathrm{T}} \qquad (27b)$$

$$P = I_n \otimes \left(\sum_{t=2}^{T} P_t \right) \qquad (27c)$$

$$q = \mathrm{vec}\left(\sum_{t=2}^{T} P_{t-1,t} \right) \qquad (27d)$$

$$r = 0 \qquad (27e)$$

Here, P_t and $P_{t-1,t}$ are taken directly from the E-step of EM.

5.2 Generating Constraints

The feasible set of the quadratic objective function is the space of all $n \times n$ matrices, regardless of their stability. When its solution yields an unstable matrix, the spectral

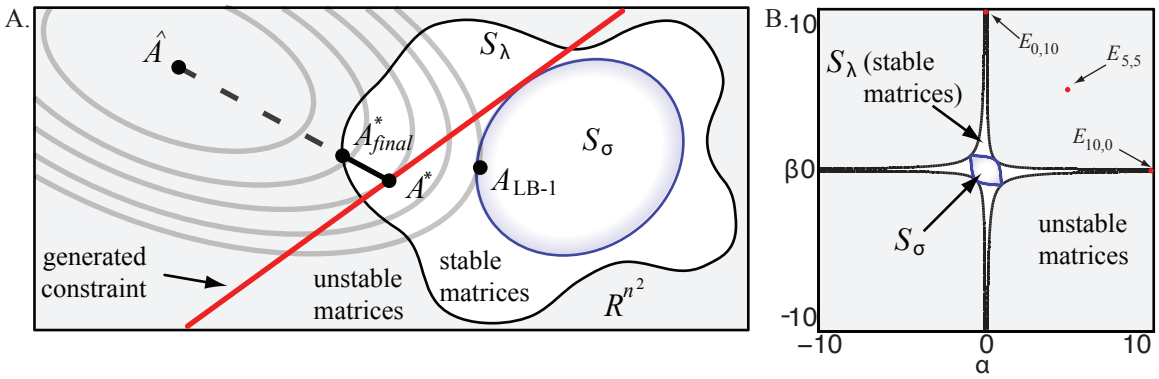

Figure 4: (A): Conceptual depiction of the space of $n \times n$ matrices. The region of stability (S_λ) is non-convex while the smaller region of matrices with $\sigma_1 \leq 1$ (S_σ) is convex. The elliptical contours indicate level sets of the quadratic objective function of the QP. $\hat{A}$ is the unconstrained least-squares solution to this objective. $A_{\text{LB-1}}$ is the solution found by LB-1 (Lacy and Bernstein, 2002). One iteration of constraint generation yields the constraint indicated by the line labeled 'generated constraint', and (in this case) leads to a stable solution A^*. The final step of our algorithm improves on this solution by interpolating A^* with the previous solution (in this case, $\hat{A}$) to obtain A^*_{final}. (B): The actual stable and unstable regions for the space of 2×2 matrices $E_{\alpha,\beta} = [\ 0.3 \ \ \alpha\ ;\beta \ \ 0.3\ ]$, with $\alpha, \beta \in [-10, 10]$. Constraint generation is able to learn a nearly optimal model from a noisy state sequence of length 7 simulated from $E_{0,10}$, with better state reconstruction error than either LB-1 or LB-2. The matrices $E_{10,0}$ and $E_{0,10}$ are stable, but their convex combination $E_{5,5} = 0.5E_{10,0} + (1 - 0.5)E_{0,10}$ is unstable.

radius of $\hat{A}$ (i.e. $|\lambda_1(\hat{A})|$) is greater than 1. Ideally we would like to use $\hat{A}$ to calculate a convex constraint on the spectral radius. However, consider the class of 2×2 matrices: $E_{\alpha,\beta} = [\ 0.3 \ \ \alpha\ ;\beta \ \ 0.3\ ]$ (Ng and Kim, 2004). The matrices $E_{10,0}$ and $E_{0,10}$ are stable with $\lambda_1 = 0.3$, but their convex combination $\gamma E_{10,0} + (1 - \gamma)E_{0,10}$ is unstable for (e.g.) $\gamma = 0.5$ (Figure 4(B)). This shows that the set of stable matrices is non-convex for $n = 2$, and in fact this is true for all $n > 1$. We turn instead to the largest *singular value*, which is a closely related quantity since

$$\sigma_{min}(\hat{A}) \leq |\lambda_i(\hat{A})| \leq \sigma_{max}(\hat{A}) \quad \forall i = 1, \ldots, n \qquad \text{(Horn and Johnson, 1985)}$$

Therefore every unstable matrix has a singular value greater than one, but the converse is not necessarily true. Moreover, the set of matrices with $\sigma_1 \leq 1$ *is* convex[5]. Figure 4(A) conceptually depicts the non-convex region of stability S_λ and the convex region S_σ with $\sigma_1 \leq 1$ in the space of all $n \times n$ matrices for some fixed n. The difference between S_σ and S_λ can be significant. Figure 4(B) depicts these regions for $E_{\alpha,\beta}$ with $\alpha, \beta \in [-10, 10]$. The

5. Since $\sigma_1(M) \equiv \max_{u,v:\|u\|_2=1,\|v\|_2=1} u^{\mathrm{T}} M v$, so if $\sigma_1(M_1) \leq 1$ and $\sigma_1(M_2) \leq 1$, then for all convex combinations, $\sigma_1(\gamma M_1 + (1-\gamma)M_2) = \max_{u,v:\|u\|_2=1,\|v\|_2=1} \gamma u^{\mathrm{T}} M_1 v + (1-\gamma) u^{\mathrm{T}} M_2 v \leq 1$.

stable matrices $E_{10,0}$ and $E_{0,10}$ reside at the edges of the figure. While results for this class of matrices vary based on the instance used, the constraint generation algorithm described below is able to learn a nearly optimal model from a noisy state sequence of $\tau = 7$ simulated from $E_{0,10}$, with better state reconstruction error than LB-1 and LB-2.

Let $\hat{A} = \tilde{U}\tilde{\Sigma}\tilde{V}^{\mathrm{T}}$ by SVD, where $\tilde{U} = [\tilde{u}_i]_{i=1}^n$ and $\tilde{V} = [\tilde{v}_i]_{i=1}^n$ and $\tilde{\Sigma} = \mathrm{diag}\{\tilde{\sigma}_1, \ldots, \tilde{\sigma}_n\}$. Then:

$$\hat{A} = \tilde{U}\tilde{\Sigma}\tilde{V}^{\mathrm{T}} \Rightarrow \qquad \tilde{\Sigma} = \tilde{U}^{\mathrm{T}}\hat{A}\tilde{V} \Rightarrow \qquad \tilde{\sigma}_1(\hat{A}) = \tilde{u}_1^{\mathrm{T}}\hat{A}\tilde{v}_1 = \mathrm{tr}(\tilde{u}_1^{\mathrm{T}}\hat{A}\tilde{v}_1) \tag{28}$$

Therefore, instability of $\hat{A}$ implies that:

$$\tilde{\sigma}_1 > 1 \Rightarrow \quad \mathrm{tr}\left(\tilde{u}_1^{\mathrm{T}}\hat{A}\tilde{v}_1\right) > 1 \Rightarrow \quad \mathrm{tr}\left(\tilde{v}_1\tilde{u}_1^{\mathrm{T}}\hat{A}\right) > 1 \Rightarrow \quad g^{\mathrm{T}}\hat{a} > 1 \tag{29}$$

Here $g = \mathrm{vec}(\tilde{u}_1\tilde{v}_1^{\mathrm{T}})$. Since Eq. (29) arose from an unstable solution of Eq. (27a), g is a hyperplane separating $\hat{a}$ from the space of matrices with $\sigma_1 \leq 1$. We use the negation of Eq. (29) as a constraint:

$$g^{\mathrm{T}}\hat{a} \leq 1 \tag{30}$$

5.3 Computing the Solution

The overall quadratic program can be stated as:

$$\begin{array}{ll} \text{minimize} & a^{\mathrm{T}}Pa - 2\, q^T a + r \\ \text{subject to} & Ga \leq h \end{array} \tag{31}$$

with a, P, q and r as defined in Eqs. (26e). $\{G, h\}$ define the set of constraints, and are initially empty. The QP is invoked repeatedly until the stable region, i.e. S_λ, is reached. At each iteration, we calculate a linear constraint of the form in Eq. (30), add the corresponding g^{T} as a row in G, and augment h with 1. Note that we will almost always stop *before* reaching the feasible region S_σ.

5.4 Refinement

Once a stable matrix is obtained, it is possible to refine this solution. We know that the last constraint caused our solution to cross the boundary of S_λ, so we interpolate between the last solution and the previous iteration's solution using binary search to look for a boundary of the stable region, in order to obtain a better objective value while remaining stable. This results in a stable matrix with top eigenvalue slightly less than 1. In principle, such an interpolation could be attempted between the least squares solution and any stable solution. However, the stable region can be highly complex, and there may be several folds and boundaries of the stable region in the interpolated area. In our experiments (not shown), interpolating from the Lacy-Bernstein solution yielded worse results.

6. Experiments

For learning the dynamics matrix, we implemented EM, subspace identification, constraint generation (using `quadprog`), LB-1 (Lacy and Bernstein, 2002) and LB-2 (Lacy and Bernstein, 2003) (using `CVX` with `SeDuMi`) in Matlab on a 3.2 GHz Pentium with 2 GB RAM.

	CG	LB-1	LB-1*	LB-2	CG	LB-1	LB-1*	LB-2
	`steam` ($n = 10$)				`fountain` ($n = 10$)			
$\lvert\lambda_1\rvert$	1.000	0.993	0.993	1.000	0.999	0.987	0.987	0.997
σ_1	1.036	1.000	1.000	1.034	1.051	1.000	1.000	1.054
$e_x(\%)$	**45.2**	103.3	103.3	546.9	**0.1**	4.1	4.1	3.0
time	**0.45**	95.87	3.77	0.50	**0.15**	15.43	1.09	0.49
	`steam` ($n = 20$)				`fountain` ($n = 20$)			
$\lvert\lambda_1\rvert$	0.999	—	0.990	0.999	0.999	—	0.988	0.996
σ_1	1.037	—	1.000	1.062	1.054	—	1.000	1.056
$e_x(\%)$	**58.4**	—	154.7	294.8	**1.2**	—	5.0	22.3
time	**2.37**	—	1259.6	33.55	**1.63**	—	159.85	5.13
	`steam` ($n = 30$)				`fountain` ($n = 30$)			
$\lvert\lambda_1\rvert$	1.000	—	0.988	1.000	1.000	—	0.993	0.998
σ_1	1.054	—	1.000	1.130	1.030	—	1.000	1.179
$e_x(\%)$	**63.0**	—	341.3	631.5	**13.3**	—	14.9	104.8
time	**8.72**	—	23978.9	62.44	**12.68**	—	5038.94	48.55
	`steam` ($n = 40$)				`fountain` ($n = 40$)			
$\lvert\lambda_1\rvert$	1.000	—	0.989	1.000	1.000	—	0.991	1.000
σ_1	1.120	—	1.000	1.128	1.034	—	1.000	1.172
$e_x(\%)$	**20.24**	—	282.7	768.5	**3.3**	—	4.8	21.5
time	**5.85**	—	79516.98	289.79	**61.9**	—	43457.77	239.53

Table 1: Quantitative results on the dynamic textures data for different numbers of states n. CG is our algorithm, LB-1and LB-2 are competing algorithms, and LB-1* is a simulation of LB-1 using our algorithm by generating constraints until we reach S_σ, since LB-1 failed for $n > 10$ due to memory limits. e_x is percent difference in squared reconstruction error. Constraint generation, in all cases, has lower error and faster runtime.

Note that the algorithms that constrain the solution to be stable give a different result from the basic EM and and subspace ID algorithms only in situations when the learned $\hat{A}$ is unstable. However, LDSs learned in scarce-data scenarios are unstable for almost any domain, and some domains lead to unstable models up to the limit of available data (e.g. the `steam` dynamic textures in Section 6.1). The goals of our experiments are to: (1) compare learning LDSs with EM to learning LDs with subspace ID; (2) examine the state evolution and simulated observations of models learned using constraint generation, and compare them to previous work on learning stable dynamical systems; and (3) compare the algorithms in terms of predictive accuracy and computational efficiency. We apply these algorithms to learning dynamic textures from the vision domain (Section 6.1), learning models of robot sensor data (Section 6.2) as well as OTC drug sales counts (Section 6.3) and sunspot numbers (Section 6.4).

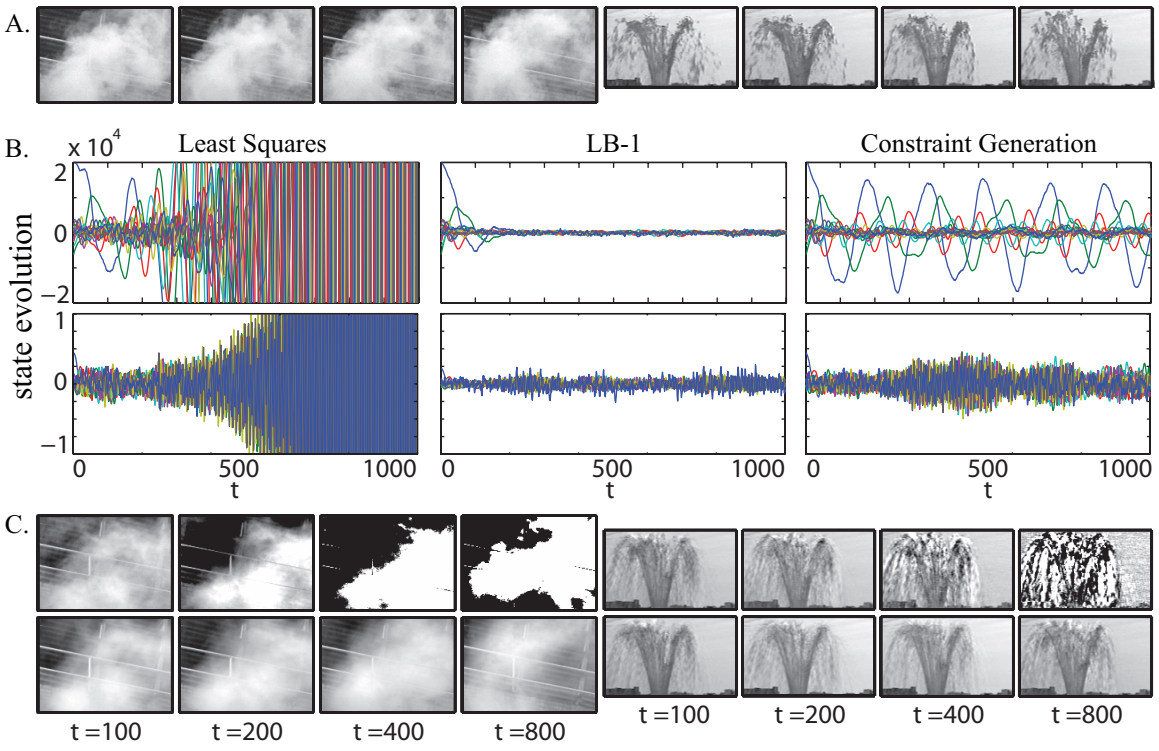

Figure 5: Dynamic textures. A. Samples from the original `steam` sequence and the `fountain` sequence. B. State evolution of synthesized sequences over 1000 frames (`steam` top, `fountain` bottom). The least squares solutions display instability as time progresses. The solutions obtained using LB-1 remain stable for the full 1000 frame image sequence. The constraint generation solutions, however, yield state sequences that are stable over the full 1000 frame image sequence without significant damping. C. Samples drawn from a least squares synthesized sequences (top), and samples drawn from a constraint generation synthesized sequence (bottom). Images for LB-1 are not shown. The constraint generation synthesized `steam` sequence is qualitatively better looking than the `steam` sequence generated by LB-1, although there is little qualitative difference between the two synthesized `fountain` sequences.

6.1 Stable Dynamic Textures

Dynamic textures in vision can intuitively be described as models for sequences of images that exhibit some form of low-dimensional structure and recurrent (though not necessarily repeating) characteristics, e.g., fixed-background videos of rising smoke or flowing water. Treating each frame of a video as an observation vector of pixel values y_t, we learned dynamic texture models of two video sequences by subspace identification: the `steam` sequence, composed of 120×170 pixel images, and the `fountain` sequence, composed of 150×90 pixel images, both of which originated from the MIT temporal texture database (Figure 5(A)).

We use the following parameters: training data size $\tau = 80$, number of latent state dimensions $n = 15$, and number of past and future observations in the Hankel matrix $i = 5$. Note that, while the observations are the raw pixel values, the underlying state sequence we learn has no *a priori* interpretation.

An LDS model of a dynamic texture may *synthesize* an arbitrarily long sequence of images by driving the model with zero mean Gaussian noise. Each of our two models uses an 80 frame training sequence to generate 1000 sequential images in this way. To better visualize the difference between image sequences generated by least-squares, LB-1, and constraint generation, the evolution of each method's state is plotted over the course of the synthesized sequences (Figure 5(B)). Sequences generated by the least squares models appear to be unstable, and this was in fact the case; both the `steam` and the `fountain` sequences resulted in unstable dynamics matrices. Conversely, the constrained subspace identification algorithms all produced well-behaved sequences of states and stable dynamics matrices (Table 1), although constraint generation demonstrates the fastest runtime, best scalability, and lowest error of any stability-enforcing approach.

A qualitative comparison of images generated by constraint generation and least squares (Figure 5(C)) indicates the effect of instability in synthesized sequences generated from dynamic texture models. While the unstable least-squares model demonstrates a dramatic increase in image contrast over time, the constraint generation model continues to generate qualitatively reasonable images. Qualitative comparisons between constraint generation and LB-1 indicate that constraint generation learns models that generate more natural-looking video sequences[6] than LB-1.

Given the paucity of data available when modeling dynamic textures, it is not possible to test the long-range predictive power of the learned dynamical systems quantitatively (such results are illustrated in Section 6.2 on robot sensor data). Instead, the error metric used for the quantitative experiments when evaluating matrix A^* is

$$e_x(A^*) = 100\% \times \left(J^2(A^*) - J^2(\hat{A}) \right) / J^2(\hat{A}) \tag{32}$$

i.e. percent increase in squared reconstruction error compared to least squares, with $J(\cdot)$ as defined in Eq. (27a). Table 1 demonstrates that constraint generation always has the lowest error as well as the fastest runtime. The running time of constraint generation depends on the number of constraints needed to reach a stable solution. Note that LB-1 is more efficient and scalable when simulated using constraint generation (by adding constraints until S_σ is reached) than it is in its original SDP formulation.

6.2 Stable Models of Robot Sensor Data

We investigate the problem of learning a dynamical model of sensory input from a mobile robot in an indoor environment. Video and associated laser range scans consisting of 2000 frames each, as well as the estimated change in pose (x, y position and orientation θ) derived from odometry, were collected at 6 frames-per-second from a Point Grey Bumblebee2

6. See videos at http://www.select.cs.cmu.edu/projects/stableLDS

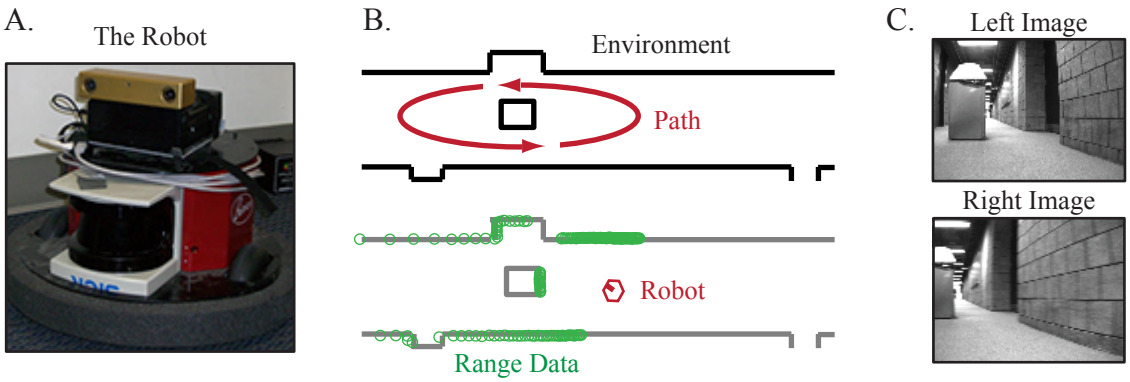

Figure 6: (A): The mobile robotic platform used in experiments. The robot is outfitted with two sensors, a Point Grey stereo camera and a SICK laser rangefinder. (B): The robot in its environment. The upper figure depicts the hallway environment with a central obstacle (black) and the path that the robot took through the environment while collecting data (the red counter-clockwise ellipse). The lower figure shows an example of the robot's laser range scan at a single point in time. The range readings are represented by circles (green) and plotted relative to the robot position (the red hexagon). The range scan consists of 180 measurements, one per degree, indicating the distance to the closest surfaces along each degree. (C):Two graysacle images (left and right) captured by the stereo camera from the robot in the position indicated in (B). Major features of the environmental geometry including walls and the central obstacle are visible.

stereo camera and a SICK laser rangefinder mounted on a Botrics O-bot d100 mobile robot platform (Figure 6(A)) circling an obstacle (Figure 6(B)). The goal was to learn a linear dynamical system with exogenous inputs that jointly modeled the probability distribution over video (Figure 6(C)) and range data conditioned on change in pose. Given the high dimensionality of the sensor data, images and range scans were preprocessed as follows. The raw sensor data consisting of pixels and range readings were vectorized at each time-step and concatenated to form high dimensional observation vectors. These vectors were centered, the total variance between the laser readings and images was normalized, and reduced to 10 dimensions via a singular value decomposition. Once the sequence of 2000 10-dimensional processed observations were formed, two sets of experiments were conducted.

First we compared the predictive power of models learned by subspace ID, EM, and EM initialized by subspace ID. From the initial set of 2000 observations, 15 stable models with 10 dimensional state were learned by each of the three approaches from different sequences of 750 observations. For subspace ID, we used Hankel matrices of 50 stacked observations in Y_p and Y_f. The models were evaluated by comparing the log likelihood of observations while filtering and predicting (Figure 7(A)) on 1500 subsequences of data. The results indicate that, while filtering, EM initialized with the parameters estimated from subspace

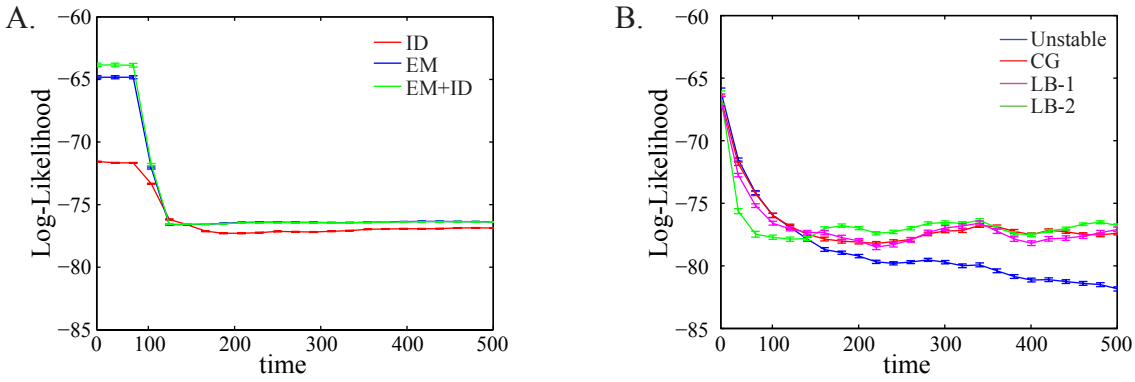

Figure 7: (A): The log-likelihood of observations while filtering (the first 100 frames) and predicting (the next 400 frames) using stable models learned by subspace ID (ID), expectation maximization (EM), and EM initialized by subspace ID (EM+ID) (see text for details). Data points were plotted with error bars every 20 frames. Note that EM initialized by subspace ID gives the best results in terms of log-likelihood. All three models quickly converge to the stationary distribution of the time series when predicting. (B): The log-likelihood of observations while predicting over 500 frames using the various stabilization approaches (see text for details): the unstable model (Unstable), constraint generation (CG), and the two models previous models (LB-1) and (LB-2). Data points were plotted with error bars every 20 frames. Note that CG provides the best short term predictions nearly mirroring the unstable model, while all three stabilized models do better than the unstable model in the long term.

ID provides the best results. While predicting, both EM with random restarts and EM initialized with subspace ID did slightly better than subspace ID alone.

Next we looked at the problem of instability in learned models of robot sense data. First, 15 sequences of 200 frames were used to learn models of the environment via EM initialized by subspace ID; of these models 10 were unstable. Next, we compared the original unstable model against models stabilized by each of the three possible stabilization approaches: our constraint generation approach, LB-1, and LB-2. The stabilization was performed during the last M-step of EM. [7] After a model was learned for a given sequence, the predictive power of the model was tested in the following way: observations were filtered for 20 frames using the original model, and, starting from the same distribution, 500 frames were predicted from each of the four models (Figure 7(B)). We repeated the experiment on 1500 separate subsequences of data. The results indicate that in the short term, constraint generation very closely approximates the unstable model, while in the long term each of the three stabilized models outpreform the unstable model.

7. While it is possible to apply CG during *each* M-step of EM, this computationally intensive alternative did not yield better results than simply applying CG during the last step of EM.

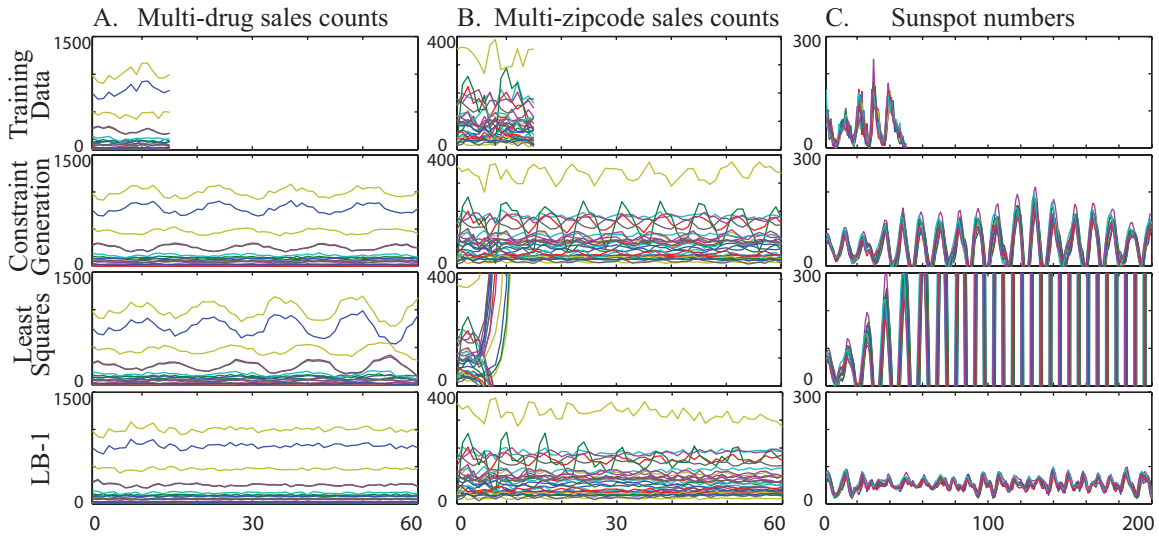

Figure 8: (A): 60 days of data for 22 drug categories aggregated over all zipcodes in the city. (B): 60 days of data for a single drug category (cough/cold) for all 29 zipcodes in the city. (C): Sunspot numbers for 200 years separately for each of the 12 months. The training data (top), simulated output from constraint generation, output from the unstable least squares model, and output from the over-damped LB-1 model (bottom).

6.3 Stable Baseline Models for Biosurveillance

We examine daily counts of OTC drug sales in pharmacies, obtained from the National Data Retail Monitor (NDRM) collection (Wagner, 2004). The counts are divided into 23 different categories and are tracked separately for each zipcode in the country. We focus on zipcodes from a particular American city. The data exhibits 7-day periodicity due to differential buying patterns during weekdays and weekends. We isolate a 60-day subsequence where the data dynamics remain relatively stationary, and attempt to learn LDS parameters to be able to simulate sequences of baseline values for use in detecting anomalies.

We perform two experiments on different aggregations of the OTC data, with parameter values We use the following parameters: number of latent state dimensions $n = 7$, number of past and future observations in the Hankel matrix $i = 4$, and training data size $\tau = 14$. Figure 8(A) plots 22 different drug categories aggregated over all zipcodes, and Figure 8(B) plots a single drug category (cough/cold) in 29 different zipcodes separately. In both cases, constraint generation is able to use very little training data to learn a stable model that captures the periodicity in the data, while the least squares model is unstable and its predictions diverge over time. LB-1 learns a model that is stable but overconstrained, and the simulated observations quickly drift from the correct magnitudes. Further details be found in Siddiqi et al. (2007).

6.4 Modeling Sunspot Numbers

We compared least squares and constraint generation on learning LDS models for the sunspot data discussed earlier in Section 3.2. We use the following parameters: number of latent state dimensions $n = 7$, number of past and future observations in the Hankel matrix $i = 9$, and training data size $\tau = 50$. Figure 8(C) represents a data-poor training scenario where we train a least-squares model on 18 timesteps, yielding an unstable model whose simulated observations increase in amplitude steadily over time. Again, constraint generation is able to use very little training data to learn a stable model that seems to capture the periodicity in the data if not the magnitude, while the least squares model is unstable. The model learned by LB-1 attenuates more noticeably, capturing the periodicity to a smaller extent. Quantitative results on both these domains exhibit similar trends as those in Table 1.

7. Related Work

Linear system identification is a well-studied subject (Ljung, 1999). Within this area, subspace identification methods (Van Overschee and De Moor (1996), Section 3.2 above) have been very successful. These techniques first estimate the model dimensionality and the underlying state sequence, and then derive parameter estimates using least squares. Within subspace methods, techniques have been developed to enforce stability by augmenting the extended observability matrix with zeros (Chui and Maciejowski, 1996) or adding a regularization term to the least squares objective (Van Gestel et al., 2001).

All previous methods were outperformed by LB-1 (Lacy and Bernstein, 2002). They formulate the problem as a semidefinite program (SDP) whose objective minimizes the state sequence reconstruction error, and whose constraint bounds the largest singular value by 1. This convex constraint is obtained by rewriting the nonlinear matrix inequality $I_n - AA^{\mathrm{T}} \succeq 0$ as a linear matrix inequality[8], where I_n is the $n \times n$ identity matrix. Here, $\succ 0$ ($\succeq 0$) denotes positive (semi-) definiteness. The existence of this constraint also proves the convexity of the $\sigma_1 \leq 1$ region. This condition is *sufficient* but not *necessary*, since a matrix that violates this condition may still be stable.

A follow-up to this work by the same authors (Lacy and Bernstein, 2003), which we call LB-2, attempts to overcome the conservativeness of LB-1 by approximating the Lyapunov inequalities $P - APA^{\mathrm{T}} \succ 0$, $P \succ 0$ with the inequalities $P - APA^{\mathrm{T}} - \delta I_n \succeq 0$, $P - \delta I_n \succeq 0$, $\delta > 0$. These inequalities hold *iff* the spectral radius is less than 1.[9] However, the approximation is achieved only at the cost of inducing a nonlinear distortion of the objective function by a problem-dependent reweighting matrix involving P, which is a variable to be optimized. In our experiments, this causes LB-2 to perform worse than LB-1 (for any δ) in terms of the state sequence reconstruction error (dynamic textures) and predictive log-likelihood (robot sensor data), even while obtaining solutions outside the feasible region

8. This bounds the top singular value by 1 since it implies $\forall x\ x^T(I_n - AA^T)x \geq 0 \Rightarrow \forall x\ x^T AA^T x \leq x^T x \Rightarrow$ for $\nu = \nu_1(AA^{\mathrm{T}})$ and $\lambda = \lambda_1(AA^{\mathrm{T}})$, $\nu^T AA^T \nu \leq \nu^T \nu \Rightarrow \nu^{\mathrm{T}} \lambda \nu \leq 1 \Rightarrow \sigma_1^2(A) \leq 1$ since $\nu^{\mathrm{T}}\nu = 1$ and $\sigma_1^2(M) = \lambda_1(MM^T)$ for any square matrix M.
9. For a proof sketch, see Horn and Johnson (1985) pg. 410.

of LB-1. Consequently, we focus on LB-1 in our conceptual and qualitative comparisons as it is the strongest baseline available. However, LB-2 is more scalable than LB-1, so quantitative results are presented for both.

To summarize the distinction between constraint generation, LB-1 and LB-2: it is hard to have both the right objective function (reconstruction error) and the right feasible region (the set of stable matrices). LB-1 optimizes the right objective but over the wrong feasible region (the set of matrices with $\sigma_1 \leq 1$). LB-2 has a feasible region close to the right one, but at the cost of distorting its objective function to an extent that it fares worse than LB-1 in nearly all cases. In contrast, our method optimizes the right objective over a less conservative feasible region than that of any previous algorithm with the right objective, and this combination is shown to work the best in practice.

8. Discussion

We have introduced a novel method for learning stable linear dynamical systems. Our constraint generation algorithm is more powerful than previous methods in the sense of optimizing over a larger set of stable matrices with a suitable objective function. In practice, the benefits of stability guarantees are readily noticeable, especially when the training data is limited. This connection between stability and amount of training data is important in practice, since time series data is often expensive to collect in large quantities, especially for phenomena with long periods in domains like physics or astronomy. The constraint generation approach also has the benefit of being faster than previous methods in nearly all of our experiments.

References

Stephen Boyd and Lieven Vandenberghe. *Convex Optimization*. Cambridge University Press, 2004.

N. L. C. Chui and J. M. Maciejowski. Realization of stable models with subspace methods. *Automatica*, 32(100):1587–1595, 1996.

Zoubin Ghahramani and Geoffrey E. Hinton. Parameter estimation for Linear Dynamical Systems. Technical Report CRG-TR-96-2, U. of Toronto, Department of Comp. Sci., 1996.

Aristide Halanay and Vladimir Rasvan. *Stability and Stable Oscillations in Discrete Time Systems*. CRC, 2000.

J. B. Hoagg, Seth L. Lacy, , R. S. Erwin, and Dennis S. Bernstein. First-order-hold sampling of positive real systems and subspace identification of positive real models. In *Proceedings of the American Control Conference*, 2004.

Roger Horn and Charles R. Johnson. *Matrix Analysis*. Cambridge University Press, 1985.

R. Horst and P. M. Pardalos, editors. *Handbook of Global Optimization*. Kluwer, 1995.

R.E. Kalman. A new approach to linear filtering and prediction problems. *Transactions of the ASME–Journal of Basic Engineering*, 1960.

Tohru Katayama. *Subspace Methods for System Identification: A Realization Approach.* Springer, 2005.

E. Keogh and T. Folias. The UCR Time Series Data Mining Archive, 2002. URL `http://www.cs.ucr.edu/ eamonn/TSDMA/index.html`.

Seth L. Lacy and Dennis S. Bernstein. Subspace identification with guaranteed stability using constrained optimization. In *Proc. American Control Conference*, 2002.

Seth L. Lacy and Dennis S. Bernstein. Subspace identification with guaranteed stability using constrained optimization. *IEEE Transactions on Automatic Control*, 48(7):1259–1263, July 2003.

L. Ljung. *System Identification: Theory for the user.* Prentice Hall, 2nd edition, 1999.

Kevin Murphy. *Dynamic Bayesian Networks: Representation, Inference and Learning.* PhD thesis, UC Berkeley, 2002.

Andrew Y. Ng and H. Jin Kim. Stable adaptive control with online learning. In *Proc. NIPS*, 2004.

H. Rauch. Solutions to the linear smoothing problem. In *IEEE Transactions on Automatic Control*, 1963.

Sajid Siddiqi, Byron Boots, Geoffrey J. Gordon, and Artur W. Dubrawski. Learning stable multivariate baseline models for outbreak detection. *Advances in Disease Surveillance*, 4:266, 2007.

S. Soatto, G. Doretto, and Y. Wu. Dynamic Textures. *Intl. Conf. on Computer Vision*, 2001.

T. Van Gestel, J. A. K. Suykens, P. Van Dooren, and B. De Moor. Identification of stable models in subspace identification by using regularization. *IEEE Transactions on Automatic Control*, 2001.

P. Van Overschee and B. De Moor. *Subspace Identification for Linear Systems: Theory, Implementation, Applications.* Kluwer, 1996.

M. Wagner. A national retail data monitor for public health surveillance. *Morbidity and Mortality Weekly Report*, 53:40–42, 2004.