Learning the Linear Dynamical System with ASOS ("Approximated Second-Order Statistics")

James Martens

University of Toronto

June 24, 2010
The Linear Dynamical System model

- model of vector-valued time-series \( \{y_t \in \mathbb{R}^{N_y}\}_{t=1}^T \)
The Linear Dynamical System model

- model of vector-valued time-series \( \{y_t \in \mathbb{R}^{N_y}\}_{t=1}^{T} \)
- widely-applied due to predictable behavior, easy inference, etc
The Linear Dynamical System model

- model of vector-valued time-series \( \{y_t \in \mathbb{R}^{N_y}\}_{t=1}^T \)
- widely-applied due to predictable behavior, easy inference, etc
- vector-valued hidden states \( \{x_t \in \mathbb{R}^{N_x}\}_{t=1}^T \) evolve via linear dynamics,

\[
x_{t+1} = Ax_t + \epsilon_t \quad A \in \mathbb{R}^{N_x \times N_x} \quad \epsilon_t \sim \mathcal{N}(0, Q)
\]
The Linear Dynamical System model

- model of vector-valued time-series \( \{ y_t \in \mathbb{R}^{N_y} \}_{t=1}^T \)
- widely-applied due to predictable behavior, easy inference, etc
- vector-valued hidden states \( \{ x_t \in \mathbb{R}^{N_x} \}_{t=1}^T \) evolve via linear dynamics,
  \[
  x_{t+1} = Ax_t + \epsilon_t \\
  A \in \mathbb{R}^{N_x \times N_x} \\
  \epsilon_t \sim N(0, Q)
  \]
- linearly generated observations:
  \[
  y_t = Cx_t + \delta_t \\
  C \in \mathbb{R}^{N_y \times N_x} \\
  \delta_t \sim N(0, R)
  \]
Learning the LDS

Expectation Maximization (EM)

- finds local optimum of log-likelihood
- pretty slow - convergence requires lots of iterations and E-step is expensive
Learning the LDS

Expectation Maximization (EM)
- finds local optimum of log-likelihood
- pretty slow - convergence requires lots of iterations and E-step is expensive

Subspace identification
- hidden states estimated directly from the data, and the parameters from these
- asymptotically unbiased / consistent
- non-iterative algorithm, but solution not optimal in any objective
- good way to initialize EM or other iterative optimizers
Our contribution

- accelerate the EM algorithm by reducing its per-iteration cost to be constant time w.r.t. $T$ (length of the time-series)
Our contribution

- accelerate the EM algorithm by reducing its per-iteration cost to be constant time w.r.t. $T$ (length of the time-series)
- key idea: approximate the inference done in the E-step
accelerate the EM algorithm by reducing its per-iteration cost to be constant time w.r.t. $T$ (length of the time-series)

key idea: approximate the inference done in the E-step

E-step approximation is unbiased and asymptotically consistent

also convergences exponentially with $L$, where $L$ is a meta-parameter that trades off approximation quality with speed

(notation change: $L$ is “$k_{lim}$” from the paper)
E.M. Objective Function

At each iteration we maximize the following objective where $\theta_n$ is the current parameter estimate:

$$Q_n(\theta) = E_{\theta_n}[\log p(x, y)|y] = \int_x p(x|y, \theta_n) \log p(x, y|\theta)$$
E.M. Objective Function
At each iteration we maximize the following objective where $\theta_n$ is the current parameter estimate:

$$Q_n(\theta) = E_{\theta_n}[\log p(x, y)|y] = \int_x p(x|y, \theta_n) \log p(x, y|\theta)$$

E-Step
- E-Step computes expectation of $\log p(x, y|\theta)$ under $p(x|y, \theta_n)$
- uses the classical Kalman filtering/smoothing algorithm
M-Step

- maximize objective $Q_n(\theta)$ w.r.t. to $\theta$, producing a new estimate $\theta_{n+1}$
  
  $\theta_{n+1} = \arg\max\limits_{\theta} Q_n(\theta)$

- very easy - similar to linear-regression
Learning via E.M. the Algorithm (cont.)

M-Step

- maximize objective $Q_n(\theta)$ w.r.t. to $\theta$, producing a new estimate $\theta_{n+1}$

$$\theta_{n+1} = \arg \max_{\theta} Q_n(\theta)$$

- very easy - similar to linear-regression

Problem

- EM can get very slow for when we have lots of data
- mainly due to call to expensive Kalman filter/smoother in the E-step
  - $O(N_x^3 T)$ where $T =$ length of the training time-series, $N_x =$ hidden state dim.
the Kalman filter/smoother estimates hidden-state means and covariances:

\[ x^k_t \equiv E_{\theta_n}[x_t | y_{\leq k}] \]
\[ V^k_{t,s} \equiv \text{Cov}_{\theta_n}[x_t, x_s | y_{\leq k}] \]

for each \( t = \{1, \ldots, T\} \) and \( s = t, t + 1 \).
the Kalman filter/smoother estimates hidden-state means and covariances:

\[ x_k^t \equiv E_{\theta_n}[ x_t | y_{\leq k} ] \]
\[ V_{t,s}^k \equiv \text{Cov}_{\theta_n}[ x_t, x_s | y_{\leq k} ] \]

for each \( t = \{1, \ldots, T\} \) and \( s = t, t + 1 \).

these are summed over time to obtain the statistics required for M-step, e.g.:

\[ E_{\theta_n}[ x_{t+1}x_t' | y_{\leq k} ] = (x^T, x^T)_1 + \sum_{t=1}^{T-1} V_{t+1,t}^T \]

where \( (a, b)_k \equiv \sum_{t=1}^{T-k} a_{t+k} b'_t \)
the Kalman filter/smoother estimates hidden-state means and covariances:

\[ x_t^k \equiv \mathbb{E}_{\theta_n}[x_t \mid y_{\leq k}] \]

\[ V_{t,s}^k \equiv \text{Cov}_{\theta_n}[x_t, x_s \mid y_{\leq k}] \]

for each \( t = \{1, \ldots, T\} \) and \( s = t, t + 1 \).

these are summed over time to obtain the statistics required for M-step, e.g.:

\[ \mathbb{E}_{\theta_n}[x_{t+1}x'_t \mid y_{\leq k}] = (x^T, x^T)_1 + \sum_{t=1}^{T-1} V_{t+1,t}^T \]

where \( (a, b)_k \equiv \sum_{t=1}^{T-k} a_{t+k} b'_t \)

but we only care about the M-statistics, not the individual inferences for each time-step → so let’s estimate these directly!
Steady-state

- first we need a basic tool from linear systems/control theory: “steady-state”
Steady-state

- first we need a basic tool from linear systems/control theory: “steady-state”

- the covariance terms, and the “filtering and smoothing matrices” (denoted $K_t$ and $J_t$) do not depend on the data $y$ - only the current parameters
Steady-state

- first we need a basic tool from linear systems/control theory: “steady-state”

- the covariance terms, and the “filtering and smoothing matrices” (denoted $K_t$ and $J_t$) do not depend on the data $y$ - only the current parameters

- and they rapidly converge to “steady-state” values:

$$V^T_{t,t}, V^T_{t,t-1}, J_t, K_t \rightarrow \Lambda_0, \Lambda_1, J, K \text{ as } \min(t, T - t) \rightarrow \infty$$
we can approximate the Kalman filter/smoother equations using the steady-state matrices
we can approximate the Kalman filter/smoother equations using the steady-state matrices

this gives the highly simplified recurrences

\[ x_t^* = Hx_{t-1}^* + Ky_t \]
\[ x_t^T = Jx_{t+1}^T + Px_t^* \]

where \( x_t^* \equiv x_t^t \equiv E_{\theta_n}[x_t | y_{\leq t}] \), \( H \equiv A - KCA \) and \( P \equiv I - JA \)
we can approximate the Kalman filter/smoother equations using the steady-state matrices

this gives the highly simplified recurrences

\[
\begin{align*}
    x_t^* &= Hx_{t-1}^* + Ky_t \\
    x_t^T &= Jx_{t+1}^T + Px_t^*
\end{align*}
\]

where \( x_t^* \equiv x_t^T \equiv \mathbb{E}_{\theta_n} [ x_t | y_{\leq t} ] \), \( H \equiv A - KCA \) and \( P \equiv I - JA \)

these don’t require any matrix multiplications or inversions
we can approximate the Kalman filter/smoother equations using the steady-state matrices

this gives the highly simplified recurrences

\[ x_t^* = H x_{t-1}^* + K y_t \]
\[ x_t^T = J x_{t+1}^T + P x_t^* \]

where \( x_t^* \equiv x_t^t \equiv E_{\theta_n}[x_t | y_{\leq t}] \), \( H \equiv A - KCA \) and \( P \equiv I - JA \)

these don’t require any matrix multiplications or inversions

we apply the approximate filter/smoother everywhere except first and last \( i \) time-steps

yields a run-time of \( O(N_x^2 T + N_x^3 i) \).
steady-state makes covariance terms easy to estimate in time independent of $T$
steady-state makes covariance terms easy to estimate in time independent of $T$

we want something similar for sum-of-products of means terms like $(x^T, x^T)_0 \equiv \sum_t x_t^T (x_t^T)'$
steady-state makes covariance terms easy to estimate in time independent of $T$

we want something similar for sum-of-products of means terms like
\[(x^T, x^T)_0 \equiv \sum_t x_t^T (x_t^T)'
\]
such sums we will call “2\textsuperscript{nd}-order statistics”. The ones-needed for the M-step are the “M-statistics”
steady-state makes covariance terms easy to estimate in time independent of $T$

we want something similar for sum-of-products of means terms like $(x^T, x^T)_0 \equiv \sum_t x^T_t (x^T_t)'$

such sums we will call “2\textsuperscript{nd}-order statistics”. The ones-needed for the M-step are the “M-statistics”

idea #1: derive recursions and equations that relate the 2\textsuperscript{nd}-order statistics of different “time-lags”

“time-lag” refers to the value of $k$ in $(a, b)_k \equiv \sum_{t=1}^{T-k} a_{t+k} b'_t$
steady-state makes covariance terms easy to estimate in time independent of $T$

we want something similar for sum-of-products of means terms like $(x^T, x^T)_0 \equiv \sum_t x_t^T (x_t^T)'$

such sums we will call “2nd-order statistics”. The ones-needed for the M-step are the “M-statistics”

idea #1: derive recursions and equations that relate the 2nd-order statistics of different “time-lags”

“time-lag” refers to the value of $k$ in $(a, b)_k \equiv \sum_{t=1}^{T-k} a_{t+k} b'_t$

idea #2: evaluate these efficiently using approximations
Deriving the 2nd-order recursions/equations: An example

- suppose we wish to find the recursion for \((x^*, y)_k\)
suppose we wish to find the recursion for \((x^*, y)_k\)

steady-state Kalman recursion for \(x^*_{t+k}\) is: \(x^*_{t+k} = Hx^*_{t+k-1} + Ky_{t+k}\)

right-multiply both sides by \(y'_t\) and sum over \(t\)

\[
(x^*, y)_k \equiv \sum_{t=1}^{T-k} x^*_{t+k} y'_t = \sum_{t=1}^{T-k} (Hx^*_{t+k-1} y'_t + Ky_{t+k} y'_t)
\]
Deriving the 2nd-order recursions/equations: An example

- Suppose we wish to find the recursion for \((x^*, y)_k\)
- Steady-state Kalman recursion for \(x_{t+k}^*\) is: \(x_{t+k}^* = Hx_{t+k-1}^* + Ky_{t+k}\)
- Right-multiply both sides by \(y'_t\) and sum over \(t\)
- Factor out matrices \(H\) and \(K\)

\[
(x^*, y)_k \equiv \sum_{t=1}^{T-k} x_{t+k}^* y'_t = \sum_{t=1}^{T-k} (Hx_{t+k-1}^* y'_t + Ky_{t+k} y'_t)
\]

\[
= H \sum_{t=1}^{T-k} x_{t+k-1}^* y'_t + K \sum_{t=1}^{T-k} y_{t+k} y'_t
\]
Deriving the 2\textsuperscript{nd}-order recursions/equations: An example

- suppose we wish to find the recursion for \((x^*, y)_k\)
- steady-state Kalman recursion for \(x^*_{t+k}\) is: \(x^*_{t+k} = Hx^*_{t+k-1} + Ky_{t+k}\)
- right-multiply both sides by \(y_t'\) and sum over \(t\)
- factor out matrices \(H\) and \(K\)
- finally, re-write everything using our special notation for 2\textsuperscript{nd}-order statistics: \((a, b)_k \equiv \sum_{t=1}^{T-k} a_{t+k} b_t'\)

\[
(x^*, y)_k \equiv \sum_{t=1}^{T-k} x^*_{t+k} y_t' = \sum_{t=1}^{T-k} (Hx^*_{t+k-1} y_t' + Ky_{t+k} y_t')
\]

\[
= H \sum_{t=1}^{T-k} x^*_{t+k-1} y_t' + K \sum_{t=1}^{T-k} y_{t+k} y_t'
\]

\[
= H((x^*, y)_{k-1} - x^*_{T} y_{T-k+1}') + K (y, y)_k
\]
The complete list (don’t bother to memorize this)

The recursions:

\[(y, x^*)_k = (y, x^*)_{k+1} H' + ((y, y)_k - y_{1+k} y'_1) K' + y_{1+k} x^*_1\]

\[(x^*, y)_k = H((x^*, y)_{k-1} - x^*_T y'_T_{-k+1}) + K(y, y)_k\]

\[(x^*, x^*)_k = (x^*, x^*)_{k+1} H' + ((x^*, y)_k - x^*_{1+k} y'_1) K' + x^*_{1+k} x^*_1\]

\[(x^*, x^*)_k = H((x^*, x^*)_{k-1} - x^*_T x^*_{T-k+1}') + K(y, x^*)_k\]

\[\left(\begin{array}{c}
x^T, y \end{array}\right)_k = J\left(\begin{array}{c}
x^T, y \end{array}\right)_{k+1} + P((x^*, y)_k - x^*_T y_{T-1}') + x^T T y_{T-1}'\]

\[\left(\begin{array}{c}
x^T, x^* \end{array}\right)_k = J\left(\begin{array}{c}
x^T, x^* \end{array}\right)_{k+1} + P((x^*, x^*)_k - x^*_T x^*_{T-1}') + x^T T x^*_{T-1}'\]

\[\left(\begin{array}{c}
x^T, x^T \end{array}\right)_k = J\left(\begin{array}{c}
x^T, x^T \end{array}\right)_{k+1} + P((x^*, x^T)_k - x^*_T x^*_{T-1}') + x^T T x^*_{T-1}'\]

The equations:

\[(x^*, x^*)_k = H(x^*, x^*)_k H' + ((x^*, y)_k - x^*_{1+k} y'_1) K' - H x^*_T x^*_{T-k}' H' + K(y, x^*)_k+1 H' + x^*_{1+k} x^*_1\]

\[\left(\begin{array}{c}
x^T, x^T \end{array}\right)_k = J\left(\begin{array}{c}
x^T, x^T \end{array}\right)_{k+1} J' + P((x^*, x^T)_k - x^*_T x^*_{T-k}') - J x^T_{k+1} x^T_{1} J' + J(x^T, x^*)_k+1 P' + x^T T x^*_{T-k}'\]
noting that statistics of time-lag $T + 1$ are 0 by definition we can start the 2$^{nd}$-order recursions at $t = T$

but this doesn’t get us anywhere - would be even more expensive than the usual Kalman recursions on the 1$^{st}$-order terms
noting that statistics of time-lag $T + 1$ are 0 by definition we can start the 2nd-order recursions at $t = T$

but this doesn’t get us anywhere - would be even more expensive than the usual Kalman recursions on the 1st-order terms

instead, start the recursions at time-lag $\sim L$ with unbiased approximations (“ASOS approximations”)

$$(y, x^*)_{L+1} \approx CA \left( (x^*, x^*)_L - x^* x^*_{T-L} \right), \quad (x^T, x^*)_L \approx (x^*, x^*)_L, \quad (x^T, y)_L \approx (x^*, y)_L$$
noting that statistics of time-lag $T + 1$ are 0 by definition we can start the 2nd-order recursions at $t = T$

but this doesn’t get us anywhere - would be even more expensive than the usual Kalman recursions on the 1st-order terms

instead, start the recursions at time-lag $\sim L$ with unbiased approximations (“ASOS approximations”)

\[(y, x^*)_{L+1} \approx CA \left( (x^*, x^*)_L - x^*_T x^*_T \right), \quad (x^T, x^*)_L \approx (x^*, x^*)_L, \quad (x^T, y)_L \approx (x^*, y)_L\]

we also need $x^T_t$ for $t \in \{1, 2, \ldots, L\} \cup \{T-L, T-L+1, \ldots, T\}$ but these can be approximated by a separate procedure (see paper)
Why might this be reasonable?

- 2\textsuperscript{nd}-order statistics with large time lag quantify relationships between variables that are far apart in time
  - weaker and less important than relationships between variables that are close in time
Why might this be reasonable?

- 2nd-order statistics with large time lag quantify relationships between variables that are far apart in time
  - weaker and less important than relationships between variables that are close in time
- in steady-state Kalman recursions, information is propagated via multiplication by $H$ and $J$:
  \[ x_t^* = Hx_{t-1}^* + Ky_t \]
  \[ x_t^T = Jx_{t+1}^T + Px_t^* \]
- both of these have spectral radius (denoted $\sigma(\cdot)$) less than 1, and so they decay the signal exponentially

\[ \sigma(H) = \sigma(J) < 1 \]
\[ \sigma(H^L) = \sigma(H)^L \ll 1 \]
Procedure for estimating the M-statistics

- how do we compute an estimate of the M-statistics consistent with the 2nd-order recursions/equations and the approximations?

essentially it is just a large linear system of dimension $O(N^2 \times L)$ but using a general solver would be far too expensive: $O(N^6 \times L^3)$

fortunately, using the special structure of this system, we have developed a (non-trivial) algorithm which is much more efficient

equations can be solved using an efficient iterative algorithm we developed for a generalization of the Sylvester equation

evaluating recursions is then straightforward

the cost is then just $O(N^3 \times L)$ after $(y, y')_k \equiv \sum_t y_t + ky'_t$ has been pre-computed for $k = 0, \ldots, L$.
Procedure for estimating the M-statistics

- how do we compute an estimate of the M-statistics consistent with the 2nd-order recursions/equations and the approximations?

- essentially it is just a large linear system of dimension $O(N^2L)$
Procedure for estimating the M-statistics

- how do we compute an estimate of the M-statistics consistent with the 2nd-order recursions/equations and the approximations?

- essentially it is just a large linear system of dimension $O(N_x^2 L)$

- but using a general solver would be far too expensive: $O(N_x^6 L^3)$
Procedure for estimating the M-statistics

- how do we compute an estimate of the M-statistics consistent with the 2nd-order recursions/equations and the approximations?

- essentially it is just a large linear system of dimension $O(N^2 L)$

- but using a general solver would be far too expensive: $O(N^6 L^3)$

- fortunately, using the special structure of this system, we have developed a (non-trivial) algorithm which is much more efficient
  - equations can be solved using an efficient iterative algorithm we developed for a generalization of the Sylvester equation
  - evaluating recursions is then straightforward
Procedure for estimating the M-statistics

- how do we compute an estimate of the M-statistics consistent with the 2nd-order recursions/equations and the approximations?

- essentially it is just a large linear system of dimension $O(N^2_\times L)$

- but using a general solver would be far too expensive: $O(N^6_\times L^3)$

- fortunately, using the special structure of this system, we have developed a (non-trivial) algorithm which is much more efficient
  - equations can be solved using an efficient iterative algorithm we developed for a generalization of the Sylvester equation
  - evaluating recursions is then straightforward

- the cost is then just $O(N^3_\times L)$ after $(y, y)_k \equiv \sum_t y_{t+k} y'_t$ has been pre-computed for $k = 0, \ldots, L$
First convergence result: our intuition confirmed

- **First result:** For a fixed $\theta$ the $\ell_2$-error in the M-statistics is bounded by a quantity proportional to $L^2 \lambda^{L-1}$, where $\lambda = \sigma(H) = \sigma(J) < 1$
  - $(\sigma(\cdot))$ denotes the spectral radius)
First convergence result: our intuition confirmed

- **First result:** For a fixed $\theta$ the $\ell_2$-error in the M-statistics is bounded by a quantity proportional to $L^2 \lambda^{L-1}$, where $\lambda = \sigma(H) = \sigma(J) < 1$
  - $(\sigma(\cdot))$ denotes the spectral radius

- so as $L$ grows, the estimation error for the M-statistics will decay exponentially
First convergence result: our intuition confirmed

- **First result:** For a fixed $\theta$ the $\ell_2$-error in the M-statistics is bounded by a quantity proportional to $L^2\lambda^{L-1}$, where $\lambda = \sigma(H) = \sigma(J) < 1$
  - ($\sigma(\cdot)$ denotes the spectral radius)

- so as $L$ grows, the estimation error for the M-statistics will decay exponentially

- *but*, $\lambda$ might be close enough to 1 so that we need to make $L$ too big
First convergence result: our intuition confirmed

- **First result**: For a fixed $\theta$ the $\ell_2$-error in the M-statistics is bounded by a quantity proportional to $L^2 \lambda^{L-1}$, where $\lambda = \sigma(H) = \sigma(J) < 1$
  - $(\sigma(\cdot))$ denotes the spectral radius)

so as $L$ grows, the estimation error for the M-statistics will decay exponentially

- *but*, $\lambda$ might be close enough to 1 so that we need to make $L$ too big

- fortunately we have a 2nd result which provides a very different type of guarantee
Second convergence result

- **Second result:** updates produced by ASOS procedure are asymptotically consistent with the usual EM updates in the limit as $T \to \infty$
  - assumes data is generated from the model
Second convergence result

- **Second result:** updates produced by ASOS procedure are asymptotically consistent with the usual EM updates in the limit as $T \to \infty$
  - assumes data is generated from the model

- first result didn’t make strong use any property of the approx.
  - (could use 0 for each and result would still hold)
Second result: updates produced by ASOS procedure are asymptotically consistent with the usual EM updates in the limit as $T \rightarrow \infty$

- assumes data is generated from the model

First result didn’t make strong use any property of the approx.

- (could use 0 for each and result would still hold)

This second one is justified in the opposite way

- strong use of the approximation
- follows from convergence of $\frac{1}{T}$-scaled expected $\ell_2$ error of approx. towards zero
- result holds for any value of $L$ value
Experiments

- we considered 3 real datasets of varying sizes and dimensionality
- each algorithm initialized from same random parameters
- latent dimension $N_x$ determined by trial-and-error

**Experimental parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>length ($T$)</th>
<th>$N_y$</th>
<th>$N_x$</th>
</tr>
</thead>
<tbody>
<tr>
<td>evaporator</td>
<td>6305</td>
<td>3</td>
<td>15</td>
</tr>
<tr>
<td>motion capture</td>
<td>15300</td>
<td>10</td>
<td>40</td>
</tr>
<tr>
<td>warship sounds</td>
<td>750000</td>
<td>1</td>
<td>20</td>
</tr>
</tbody>
</table>
Experimental results (cont.)

<table>
<thead>
<tr>
<th>Name</th>
<th>length (T)</th>
<th>$N_y$</th>
<th>$N_x$</th>
</tr>
</thead>
<tbody>
<tr>
<td>evaporator</td>
<td>6305</td>
<td>3</td>
<td>15</td>
</tr>
<tr>
<td>motion capture</td>
<td>15300</td>
<td>10</td>
<td>40</td>
</tr>
<tr>
<td>warship sounds</td>
<td>750000</td>
<td>1</td>
<td>20</td>
</tr>
</tbody>
</table>

![Graph showing iterations vs. negative log likelihood for different methods and names](attachment:graph.png)
### Experimental results (cont.)

<table>
<thead>
<tr>
<th>Name</th>
<th>length (T)</th>
<th>$N_y$</th>
<th>$N_x$</th>
</tr>
</thead>
<tbody>
<tr>
<td>evaporator</td>
<td>6305</td>
<td>3</td>
<td>15</td>
</tr>
<tr>
<td>motion capture</td>
<td>15300</td>
<td>10</td>
<td>40</td>
</tr>
<tr>
<td>warship sounds</td>
<td>750000</td>
<td>1</td>
<td>20</td>
</tr>
</tbody>
</table>

![Graph showing Negative Log Likelihood vs. Iterations](image-url)
### Experimental results (cont.)

<table>
<thead>
<tr>
<th>Name</th>
<th>length (T)</th>
<th>$N_y$</th>
<th>$N_x$</th>
</tr>
</thead>
<tbody>
<tr>
<td>evaporator</td>
<td>6305</td>
<td>3</td>
<td>15</td>
</tr>
<tr>
<td>motion capture</td>
<td>15300</td>
<td>10</td>
<td>40</td>
</tr>
<tr>
<td>warship sounds</td>
<td>750000</td>
<td>1</td>
<td>20</td>
</tr>
</tbody>
</table>

![Graph showing negative log likelihood against iterations for different sets of EM parameters.]
we applied steady-state approximations to derive a set of “2nd-order recursions and equations”
Conclusion

- we applied steady-state approximations to derive a set of “2nd-order recursions and equations”
- approximated statistics of time-lag $L$
we applied steady-state approximations to derive a set of “2nd-order recursions and equations”

approximated statistics of time-lag $L$

produced an efficient algorithm for solving the resulting system

**Per-iteration run-times:**

<table>
<thead>
<tr>
<th></th>
<th>EM</th>
<th>SS-EM</th>
<th>ASOS-EM</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$O(N_x^3 T)$</td>
<td>$O(N_x^2 T + N_x^3 i)$</td>
<td>$O(N_x^3 k_{lim})$</td>
</tr>
</tbody>
</table>
Conclusion

- we applied steady-state approximations to derive a set of “2nd-order recursions and equations”
- approximated statistics of time-lag $L$
- produced an efficient algorithm for solving the resulting system

\[
\begin{align*}
\text{Per-iteration run-times:} & \quad \text{EM} & \quad \text{SS-EM} & \quad \text{ASOS-EM} \\
& \mathcal{O}(N_x^3 T) & \mathcal{O}(N_x^2 T + N_x^3 i) & \mathcal{O}(N_x^3 k_{lim})
\end{align*}
\]

- gave 2 formal convergence results
we applied steady-state approximations to derive a set of “2nd-order recursions and equations”

approximated statistics of time-lag $L$

produced an efficient algorithm for solving the resulting system

Per-iteration run-times:

<table>
<thead>
<tr>
<th></th>
<th>EM</th>
<th>SS-EM</th>
<th>ASOS-EM</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$O(N^3_x T)$</td>
<td>$O(N^2_x T + N^3_x i)$</td>
<td>$O(N^3_x k_{lim})$</td>
</tr>
</tbody>
</table>

gave 2 formal convergence results

demonstrated significant performance benefits for learning with long time-series
Thank you for your attention