Outline

• Sequence Models / Hidden Markov Models

• Linear Dynamical Systems

• LDS Extensions
Outline

• Sequence Models

• Linear Dynamical Systems

• LDS Extensions
Sequential data has order, and the order matters.

What has happened, informs what will happen.

Sequential data is everywhere.

Examples:
- spoken language (word production)
- written language (sentence level statistics)
- weather
- human movement
- stock market data
- genomes
Sequential data

Graphical models for such data?

The complexity of the representation seems to increase with time.

Observations over time tend to depend on the past.

We can simply life by assuming that the distant past does not matter.

If we assume that history does not matter other than the immediate previous entity, we have a first order Markov model.

If what happens now depends on two previous entities, we have a second order Markov model.
Markov chains

Zeroth order

First order

Second order
Notice that this plan has arrows from data-this complicates model specification.
Hidden Markov Model (HMM)

Introducing latent state simplifies modeling data likelihood…

Intuition Temporal extension of mixture model
• Data clusters into hidden “states” $Z$ at each time
• Hidden state encodes important part of history
• Markov chain models transitions among clusters
Markovian assumptions

The basic HMM is like a mixture model, with the mixture component being used for the current observations depends on the last previous component.

\[ z_{n+1} \perp z_{n-1} \mid z_n \]

Observation likelihood \( p(x_n \mid z_n) \) models how data from a component are generated (things are easy if you know the cluster)
Transition Dynamics

State Transition Diagram
(Not a PGM)

\[ A_{jk} \equiv p\left(z_{nk} = 1 \mid z_{n-1,j} = 1\right) \]

\[ 0 \leq A_{jk} \leq 1 \quad \text{and} \quad \sum_k A_{jk} = 1 \]

\[ A_{jk} : \text{Probability of going from state } j=4 \text{ to state } k=3 \]
Starting state

Our HMM will be a pure* generative model, so we need to know how to start.

\[ \pi_k \equiv p(z_{1k} = 1) \]

with \( 0 \leq \pi_k \leq 1 \) and \( \sum_k \pi_k = 1 \)

*By pure, I mean that we can do ancestral sampling, i.e., a Bayes net.
HMM parameter summary

$\Theta = \{ \pi, A, \phi \}$

$\pi$ is probability over initial states

$A$ is transition matrix

$\phi$ are the parameters for the data emission probabilities (i.e., for $p(x_n | z_n)$, e.g., means of Gaussians)
Data distribution from an HMM

Let $X = \{x_n\}$ be the observed sequence

$$p(X|\theta) = ?$$
Data distribution from an HMM

Let $X = \{x_n\}$ be the observed sequence

$p(X|\theta)$ is a marginalization over $Z$.

$p(X,Z|\theta) = ?$
An HMM is specified by: $\theta = \{ \pi, A, \phi \}$

$$p(X, Z|\theta) = p(z_1|\pi) \prod_{n=2}^{N} p(z_n|z_{n-1}, A) \prod_{n=1}^{N} p(x_n|z_n, \phi)$$

(complete data, i.e., we can generate from this).
Data distribution from an HMM

Gaussian likelihood model $p(x_n|z_n)$

Transition probability to another state is 5% (from Bishop—the short visits in green seem a bit anomalous).
Example: Matching slides to video frames
Our state sequence corresponds to what slide is being shown.

We assume that only the jump matters. IE, going from slide 6 to 8 has the same chance of going from 10 to 12.
Matching slides to video frames

Our state sequence corresponds to what slide is being shown.

\[ p(X, Z | \theta) = p(z_1 | \pi) \left[ \prod_{n=2}^{N} p(z_n | z_{n-1}, A) \right] \prod_{m=1}^{N} p(x_m | z_m, \phi) \]
Classic HMM computational problems

1. Given unlabeled* data, what is the HMM (parameter learning).

2. Given an HMM and observed data, what is the probability distribution of states for each time point ($z_n$ in our notation).

3. Given an HMM and observed data, what is the most probable state sequence?

*We could have data with labels (annotated) which means this step becomes trivial, much like training naive Bayes versus fitting a GMM using EM. This is what we did in the matching slides to video project.
Classic HMM computational problems

1. Given unlabeled* data, what is the HMM (parameter learning).

2. Given an HMM and observed data, what is the probability distribution of states for each time point ($z_n$ in our notation).

3. Given an HMM and observed data, what is the most probable state sequence?

#2 and #3 seem similar, but to understand the difference consider a three state system about doing HW problems A, B, and C in order, with B being very easy. So you will spend most of your time in state A and C. State B may be the least likely state for every time point. But the most likely state sequence must include it.
Classic HMM computational problems

1. Given unlabeled data, what is the HMM (learning).
   This is a missing value problem, which we can tackle using EM, but we will need to solve #2 as sub-problem.

2. Given an HMM and data, what is the probability distribution of states for each time point (\(z_n\) in our notation).
   These are marginals in a Bayes net, and so we use the sum-product algorithm (in HMM often called alpha-beta or forwards backwards).

3. Given an HMM and data, what is the most likely state sequence?
   This is a maximal configuration of a Bayes net, and so we use max-sum (in HMM this is Viterbi).
Classic HMM computational problems

1. Given unlabeled data, what is the HMM (learning).
   This is a missing value problem, which we can tackle using EM, but we will need to solve #2 as sub-problem.

2. Given an HMM and data, what is the probability distribution of states for each time point ($z_n$ in our notation).
   These are marginals in a Bayes net, and so we use the sum-product algorithm (in HMM often called alpha-beta or forwards backwards).

3. Given an HMM and data, what is the most likely state sequence?
   This is a maximal configuration of a Bayes net, and so we use max-sum (in HMM this is Viterbi).
Learning HMM Parameters

**Learn These** E.g. via maximum likelihood:

\[ \theta = \{ \pi, A, \phi \} \]

**Problem** Don’t know latent states

**Observations** e.g. training data

Need to compute *marginal likelihood*:

\[
\max_\theta \mathcal{L}(\theta) = \int p(z)p(x \mid z) \, dz
\]

Natural approach is to use Expectation Maximization
**Learning the HMM with EM (sketch)**

If we know the state distributions, **and the successive state pair distributions** (needed for the transition matrix, A), **for each training sequence**, we can compute the parameters.

If we know the parameters, we can compute the state distributions, **for each training sequence** (this is HMM computation problem #2, which we need to solve as a subproblem if we use EM).

**Blue** text highlights differences from the mixture model for one sequence.

**Green** text reminds us that we need a bit of book-keeping when we train on multiple sequences.
Recall the General EM algorithm

1. Choose initial values for $\theta^{(s=1)}$

2. E step: Evaluate $p(Z \mid X, \theta^{(s)})$

3. M step: Evaluate $\theta^{(s+1)} = \arg \max_{\theta} \left\{ Q\left( \theta^{(s+1)}, \theta^{(s)} \right) \right\}$
   
   where $Q\left( \theta^{(s+1)}, \theta^{(s)} \right) = \sum_{Z} p(Z \mid X, \theta^{(s)}) \log \left( p(X, Z \mid \theta^{(s+1)}) \right)$

4. Check for convergence; If not done, goto 2.

★ At each step, our objective function increases unless it is at a local maximum. It is important to check this is happening for debugging!
HMM complete data likelihood (one sequence)

\[ p(X, Z | \theta) = p(z_1 | \pi) \left[ \prod_{n=2}^{N} p(z_n | z_{n-1}, A) \right] \prod_{n=1}^{N} p(x_n | z_n, \phi) \]
Remember our “indicator variable” notation. $Z$ is a particular assignment of the missing values (i.e., which cluster the HMM was in at each time). For each time point, $n$, one of the values of $z_n$ is one, and the others are zero. So, it “selects” the factor for the particular state at that time.
HMM complete data likelihood (one sequence)

\[ p(X,Z|\theta) = p(z_1 | \pi) \prod_{n=2}^{N} p(z_n | z_{n-1}, A) \prod_{n=1}^{N} p(x_n | z_n, \phi) \]

\[ = \prod_{k=1}^{K} \pi_{1,k}^{z_{1,k}} \prod_{n=2}^{N} \prod_{j=1}^{K} \prod_{k=1}^{K} A_{j,k}^{z_{n-1,j}^{*}z_{n,k}} \prod_{n=1}^{N} \prod_{k=1}^{K} (p(x_n | \phi_k))^ {z_{n,k}} \]

\[ \log(p(X,Z|\theta)) = \sum_{k=1}^{K} z_{1,k} \log(\pi_k) + \sum_{n=2}^{N} \sum_{j=1}^{K} \sum_{k=1}^{K} z_{n-1,j}^{*}z_{n,k} \log(A_{j,k}) + \sum_{n=1}^{N} \sum_{k=1}^{K} z_{n,k} \log(p(x_n | \phi_k)) \]

(complete data log-likelihood)
HMM complete data likelihood (E training sequences)

\[
p(X, Z | \theta) = \prod_{e=1}^{E} p(z_{e,1} | \pi) \left[ \prod_{e=1}^{E} \prod_{n=2}^{N_e} p(z_{e,n} | z_{e,n-1}, A) \right] \prod_{e=1}^{E} \prod_{n=1}^{N_e} p(x_{e,n} | z_{e,n}, \phi) \\
= \prod_{e=1}^{E} \prod_{k=1}^{K} \pi_{k} z_{s,1,k} \prod_{e=1}^{E} \prod_{n=2}^{N_e} \prod_{j=1}^{K} A_{j,k}^{z_{e,n-1,j} \ast z_{n,k}} \prod_{e=1}^{E} \prod_{n=1}^{N_e} \prod_{k=1}^{K} p(x_{e,n} | \phi_k)
\]

\[
\log(p(X, Z | \theta)) = \sum_{e=1}^{E} \sum_{k=1}^{K} z_{s,1,k} \log(\pi_k) + \sum_{e=1}^{E} \sum_{n=2}^{N_e} \sum_{j=1}^{K} z_{e,n-1,j} z_{e,n,k} \log(A_{j,k}) + \sum_{e=1}^{E} \sum_{n=1}^{N_e} \sum_{k=1}^{K} z_{e,n,k} \log(p(x_{e,n} | \phi_k))
\]

(complete data log likelihood)
Learning the HMM with EM (sketch)

In the simple clustering case (e.g., GMM), the E step was simple. For HMM it is a bit more involved.

The M step works a lot like the GMM, except we need to deal with successive states. Consider the M step first.
E-Step for HMM

*Provides two distributions (responsibilities)*...

The degree each state explains each data point (analogous to GMM responsibilities):
\[ \gamma (z_{e,n,k}) = p(z_{e,n,k} \mid X_e, \theta^{(s)}) \]

The degree that the combination of a state, and a previous one explain two data points:
\[ \xi (e, z_{n-1,j}, z_{n,k}) = p(z_{e,n-1,j}, z_{e,n,k} \mid X_e, \theta^{(s)}) \]

**Q:** How can we compute one / two stage-marginals in HMM?
Forward-Backward Algorithm

Forward message:
\[ \alpha_{n-1}(z_n) = \psi(z_n, x_n) \sum_{z_{n-1}} \alpha_{n-2}(z_{n-1}) \psi(z_{n-1}, z_n) \]

Forward message:
\[ \beta_{n+1}(z_n) = \sum_{z_{n+1}} \beta_{n+2}(z_{n+1}) \psi(z_n, z_{n+1}) \psi(z_{n+1}, x_{n+1}) \]
Forward-Backward Algorithm

Node Responsibility (a.k.a. marginal):
\[ \gamma(z_n) = p(z_n \mid \mathcal{X}) \propto \alpha_n(z_n) \beta_n(z_n) \]

Two-Node Responsibility (a.k.a. pairwise marginal):
\[ \xi(z_n, z_{n+1}) = p(z_n, z_{n+1} \mid \mathcal{X}) \propto \alpha_n(z_n) \psi(z_n, z_{n+1}) \beta_{n+1}(z_{n+1}) \]
Recall the complete data log-likelihood:

\[
\log(p(X, Z|\theta)) = \sum_{e=1}^{E} \sum_{k=1}^{K} z_{e,1k} \log(\pi_k) + \sum_{e=1}^{E} \sum_{n=2}^{N_e} \sum_{j=1}^{K} \sum_{k=1}^{K} z_{e,n-1,j} z_{e,n,k} \log(A_{j,k}) + \sum_{e=1}^{E} \sum_{n=1}^{N_e} \sum_{k=1}^{K} z_{e,n,k} \log(p(x_{e,n}|\phi_k))
\]
M-Step for HMM

Recall the complete data log-likelihood:

$$\log(p(X,Z|\theta)) = \sum_{e=1}^{E} \sum_{k=1}^{K} z_{e,1,k} \log(\pi_k) + \sum_{e=1}^{E} \sum_{n=2}^{N_e} \sum_{j=1}^{K} z_{e,n-1,j} z_{e,n,k} \log(A_{j,k}) + \sum_{e=1}^{E} \sum_{n=1}^{N_e} \sum_{k} z_{e,n,k} \log(p(x_{e,n}|\phi_k))$$

Expected complete data log-likelihood:

$$Q(\theta^{(s+1)},\theta^{(s)}) = \sum_{Z} p(Z|\theta^{(s)}) \log(p(X,Z|\theta^{(s+1)}))$$

$$= \sum_{e=1}^{E} \sum_{k=1}^{K} \gamma(z_{e,1,k}) \log(\pi_k) + \sum_{e=1}^{E} \sum_{n=2}^{N_e} \sum_{j=1}^{K} \sum_{k} \xi(e,z_{e,n-1,j},z_{e,n,k}) \log(A_{j,k}) + \sum_{e=1}^{E} \sum_{n=1}^{N_e} \sum_{k} \gamma(z_{e,n,k}) \log(p(x_{e,n}|z_{e,n,\phi}))$$
M-Step for HMM

Doing the maximization using Lagrange multipliers (or intuition) gives us

\[
\pi_k = \frac{\sum_{e=1}^{E} \sum_{k'} \gamma(z_{e,1,k})}{\sum_{e=1}^{E} \sum_{k'} \gamma(z_{e,1,k'})}
\]

Much like the GMM. Taking the partial derivative for \( \pi_k \) kills second and third terms.

\[
A_{jk} = \frac{\sum_{e=1}^{E} \sum_{n=2}^{E} \zeta(e, z_{n-1,j}, z_{n,k})}{\sum_{e=1}^{E} \sum_{k'} \sum_{n=2}^{E} \zeta(z_{n-1,j}, z_{n,k'})}
\]
M-Step for HMM

The maximization of $p(x_{e,n} | \phi)$ is exactly the same as the mixture model.

For example, if we have Gaussian emissions, then

$$
\mu_k = \frac{\sum_{e=1}^{E} \sum_{n} x_{e,n} \gamma(z_{e,n,k})}{\sum_{e=1}^{E} \sum_{n} \gamma(z_{e,n,k})}
$$
Classic HMM computational problems

Given data, what is the HMM (learning).

Given an HMM, what is the **distribution over the state** variables. Also, **how likely** are the observations, given the model.

Given an HMM, what is the most probable **state sequence** for some data?
Viterbi algorithm (special case of max-sum)

Recall max-sum

Forward direction is like sum-product, except
  Factor nodes take the max over logs instead of sum
  Variable nodes use sum of logs instead of product
  We remember incoming variable values* that give max

Backwards direction is simply backtracking on (*).
Recall sum-product for HMM

If we identify $\mu_{f_n \rightarrow f_{n+1}}(z_n) = \alpha(z_n)$

$$\alpha(z_1) = p(z_1)p(x_1|z_1)$$

$$\alpha(z_n) = \sum_{z_{n-1}} f_n(z_{n-1}, z_n) \alpha(z_{n-1})$$

$$= \sum_{z_{n-1}} p(z_n|z_{n-1}) p(x_n|z_n) \alpha(z_{n-1})$$

$$= p(x_n|z_n) \sum_{z_{n-1}} p(z_n|z_{n-1}) \alpha(z_{n-1})$$
By analogy we get max sum

Left to right messages for max-sum

\[
\omega(z_n) = \log(p(x_n|z_n)) + \max_{z_{n-1}} \left\{ \log(p(z_n|z_{n-1})) + \omega(z_{n-1}) \right\}
\]

\[
\omega(z_1) = \log(p(z_1)) + \log(p(x_1|z_1))
\]
Squares represent being in each of the three states at a given time.

We store the log of the probability of the maximal likely way to get there.

And the particular previous state that gave the max (orange)

Store $\omega_{n-1,k}$ and $k' = \arg \max(\bullet)$ for $k$
Consider all possible paths to each of the $K$ states for time $n$.

The message encodes the probabilities for the maximum probability path for each of the $K$ states.

I.E., for a given time, for each state $k$, it records is the probability of being there by via the maximal probably sequence.

That value is (recursively defined by)

$$\omega(z_n) = \log(p(x_n | z_n)) + \max_{z_{n-1}} \left\{ \log(p(z_n | z_{n-1})) + \omega(z_{n-1}) \right\}$$
The incoming message is the vector of probabilities for the maximum probability path for each of the $K$ states at the previous time.

For each state $k$

$$\omega(z_n) = \log(p(x_n|z_n)) + \max_{z_{n-1}} \left\{ \log(p(z_n|z_{n-1})) + \omega(z_{n-1}) \right\}$$

Consider getting there from each previous state $k'$

This gives the maximal probability way to be in each of $K$ states, $k$, at time $n$. 

**Story (continued)**
For Viterbi, we remember the previous state, $k'$, leading to the max for each $k$. (This is simpler than the general case because no branches).

Once we know the end state of the maximal probability path, we can find the maximal probability path by back-tracking.

You might also recognize this as dynamic programming (think minimum cost path).
Intuitive understanding

Store $\omega_{n-1,k}$ and $k' = \arg \max(\bullet)$ for $k$
If this is the end, we now know the max, and what the ending state is.
The max path is dark (but we only know it when we get to the end).

To find the path, we need to chase the back pointers.

Suppose the max is attained at $k=3$. 

**Intuitive understanding**
Classic HMM computational problems

Given data, what is the HMM (learning).

Given an HMM, what is the distribution over the state variables. Also, how likely are the observations, given the model.

Given an HMM, what is the most probable state sequence for some data?
• Sequence Models

• Linear Dynamical Systems

• LDS Extensions
Dynamical System

Models of latent states evolving over time/space

Human Pose Tracking  Stock Market Prediction  Visual Object Tracking

Move away from discrete HMM states to continuous ones…
Probabilistic Principal Component Analysis (PPCA)

Latent: $x \in \mathbb{R}^p$  Data: $y \in \mathbb{R}^q$

Typically $p < q$ for dimension reduction

Data are exchangeable linear Gaussian projections of latent quantities

$x \sim \mathcal{N}(0, I)$

$y \mid x \sim \mathcal{N}(\Lambda x + \mu, \sigma^2 I)$

[ Source: M. I. Jordan ]
Gaussian Linear Dynamical System (LDS)

Temporal extension of probabilistic PCA…

Data are time-dependent and non-exchangeable
Consider the state vector:

\[ x_t = \begin{pmatrix} x(t) \\ \dot{x}(t) \end{pmatrix} \]

where \( x(t) \): Position \( \dot{x}(t) \triangleq \frac{d}{dt}x(t) \): Velocity

Differential equations for constant velocity dynamics:

\[ x(t) = x(t - 1) + \dot{x}(t - 1) \]
\[ \dot{x}(t) = \dot{x}(t - 1) \]

Linear Gaussian state-space model

\[ x_t = F x_{t-1} + \epsilon \]

where \( F = \begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix} \) and \( \epsilon \sim \mathcal{N}(0, \Sigma) \)
Simple Linear Gaussian Dynamics

Random Walk (Brownian Motion)

Constant Velocity (a.k.a. zero acceleration)

Acceleration can be included in higher-order models as well
Define shorthand notation: \( y_{1}^{t-1} \triangleq \{y_1, \ldots, y_{t-1}\} \)

**Filtering**

Compute \( p(x_t \mid y_{1}^{t}) \) at each time \( t \)

**Smoothing**

Compute full posterior marginal \( p(x_t \mid y_{1}^{T}) \) at each time \( t \)
Suppose we have jointly Gaussian model,

\[ x \sim \mathcal{N}(\mu, \Sigma) \quad y \mid x \sim \mathcal{N}(Ax + b, R) \]

Key quantities of inference:

- Marginal: \( p(y) \)
- Posterior: \( p(x \mid y) \)

\( \text{Both are Gaussian distributions!} \)

**Gaussians closed under marginalization / conditioning**

**Marginal**

\[ p(y) = \mathcal{N}(A\mu + b, R + A\Sigma A^T) \]

**Posterior**

\[ p(x \mid y) = \mathcal{N}(\mu_{x\mid y}, \Sigma_{x\mid y}) \]

Where,

\[ \Sigma_{x\mid y}^{-1} = \Sigma^{-1} + A^T R^{-1} A \quad \text{and} \quad \mu_{x\mid y} = \Sigma_{x\mid y} \left[ A^T R^{-1} (y - b) + \Sigma^{-1} \mu \right] \]
Deriving marginal and posterior Gaussian is straightforward
• …but takes too long in lecture… (See Murphy Sec. 4.3)
• Those who did HW2 Extra Credit have seen this already!

Basic Approach
• Marginal and Posterior are closed-form Gaussians
• Use final formulas as building blocks for linear dynamical systems
Gaussian Linear Dynamical System (LDS) Inference

At time $t$ assume we have posterior,

$$p(x_t \mid y_1, \ldots, y_t) = p(x_t \mid y_t^t)$$

shorthand

**Key inference steps**

1) *Predict* state at next time

$$p(x_{t+1} \mid y_t^t)$$

Data only up to previous time

2) Update posterior with measurement

$$p(x_{t+1} \mid y_t^{t+1})$$

Data at time $t+1$

All distributions remain Gaussian because of closure properties
Suppose we have a Gaussian posterior at time $t-1$:

$$p(x_{t-1} \mid y_1^{t-1}) = \mathcal{N}(\mu_{t-1}, \Sigma_{t-1}) \quad \text{where} \quad y_1^{t-1} \triangleq \{y_1, \ldots, y_{t-1}\}$$

Forward prediction at time $t$:

$$p(x_t \mid y_1^{t-1}) = \int p(x_t \mid x_{t-1})p(x_{t-1} \mid y_1^{t-1}) \, dx_{t-1}$$

$$= \int \mathcal{N}(x_t \mid Fx_{t-1}, \Sigma)\mathcal{N}(x_{t-1} \mid \mu_{t-1}, \Sigma_{t-1}) \, dx_{t-1}$$

$$= \mathcal{N}(x_t \mid F\mu_{t-1}, \Sigma + F\Sigma_{t-1}F^T) \int \mathcal{N}(x_{t-1} \mid \cdot, \cdot) \, dx_{t-1}$$

Integrates to 1

Same form as marginal likelihood on previous slide
Gaussian LDS Filtering

- **Forward prediction at time** $t$:
  \[
  p(x_t \mid y_{1}^{t-1}) = \mathcal{N}(x_t \mid \mu_{t|t-1}, \Sigma_{t|t-1})
  \]
  
  where \( \mu_{t|t-1} \triangleq F \mu_{t-1} \) and \( \Sigma_{t|t-1} = \Sigma + F \Sigma_{t-1} F^T \)

- **State Prediction**: \( \mu_{t|t-1} \)
- **Predicted Covariance**: \( \Sigma_{t|t-1} \)

- **Posterior at time** $t$ is also Gaussian:
  \[
  p(x_t \mid y_1^t) \propto p(x_t \mid y_1^{t-1})p(y_t \mid x_t)
  = \mathcal{N}(x_t \mid \mu_{t|t-1}, \Sigma_{t|t-1})\mathcal{N}(y_t \mid Hx_t, R) \propto \mathcal{N}(x_t \mid \mu_{t|t}, \Sigma_{t|t})
  \]

  **Gain Matrix**: \( K_t = \Sigma_{t|t-1} H^T (H\Sigma_{t|t-1} H^T + R)^{-1} \)

  **Filter Covariance**: \( \Sigma_{t|t} = \Sigma_{t|t-1} - K_t H \Sigma_{t|t-1} \)

  **Filter Mean**: \( \mu_{t|t} = \mu_{t|t-1} + K_t(y_t - H \mu_{t|t-1}) \)

Can be derived from Gaussian conditional formulas and Woodbury matrix identity
Kalman Filter

Prediction Step:

State Prediction: \( \mu_{t|t-1} = F \mu_{t-1|t-1} \)

Covariance Prediction:
\[
\Sigma_{t|t-1} = \Sigma + F \Sigma_{t-1|t-1} F^T
\]

Measurement Update Step:

Gain Matrix: \( K_t = \Sigma_{t|t-1} H^T (H \Sigma_{t|t-1} H^T + R)^{-1} \)

Filter Covariance: \( \Sigma_{t|t} = \Sigma_{t|t-1} - K_t H \Sigma_{t|t-1} \)

Filter Mean: \( \mu_{t|t} = \mu_{t|t-1} + K_t (y_t - H \mu_{t|t-1}) \)
Kalman Filter

Prediction Step:

State Prediction: \( \mu_{t|t-1} = F \mu_{t-1|t-1} \)

Covariance Prediction: \( \Sigma_{t|t-1} = \Sigma_{t-1|t-1} + F \Sigma_{t-1|t-1} F^T \)

Measurement Update:

Gain Matrix: \( K_t = \Sigma_{t|t-1} H^T (H \Sigma_{t|t-1} H^T + R)^{-1} \)

Filter Covariance: \( \Sigma_{t|t} = \Sigma_{t|t-1} - K_t H \Sigma_{t|t-1} \)

Filter Mean: \( \mu_{t|t} = \mu_{t|t-1} + K_t (y_t - H \mu_{t|t-1}) \)

What algorithm does this look like?
Kalman Filter

Prediction Step:

State Prediction: \( \mu_{t|t-1} = F \mu_{t-1|t-1} \)

Covariance Prediction: \( \Sigma_{t|t-1} = \Sigma + F \Sigma_{t-1|t-1} F^T \)

Measurement Update:

Gain Matrix: \( K_t = \frac{\Sigma_{t|t-1} H^T (H \Sigma_{t|t-1} H^T + R)^{-1}}{\Sigma_{t|t-1} H^T (H \Sigma_{t|t-1} H^T + R)^{-1} + \Sigma} \)

Filter Covariance: \( \Sigma_t = \Sigma_{t|t-1} - K_t H \Sigma_{t|t-1} \)

Filter Mean: \( \mu_t = \mu_{t|t-1} + K_t (y_t - H \mu_{t|t-1}) \)

What algorithm does this look like?

Sum-Product BP
Mean Parameterization:

\[ \mathcal{N}(x \mid \mu, \Sigma) = \frac{1}{(2\pi)^{N/2} |\Sigma|^{1/2}} \exp \left\{ -\frac{1}{2} (x - \mu)^T \Sigma^{-1} (x - \mu) \right\} \]

Canonical Parameterization

\[ \mathcal{N}^{-1}(x \mid \vartheta, \Lambda) \propto \exp \left\{ -\frac{1}{2} x^T \Lambda x + \vartheta^T x \right\} \]

where \( \Lambda = \Sigma^{-1} \quad \vartheta = \Sigma^{-1} \mu \)

Also called \textit{natural parameters and information parameters in some texts...}
Gaussian Belief Propagation

\[
p_t(x_t) \propto \prod_{s \in \Gamma(t)} m_{st}(x_t) \quad m_{ts}(x_s) = \int_{x_t} \psi_{st}(x_s, x_t) \prod_{u \in \Gamma(t) \setminus s} m_{ut}(x_t) \, dx_t
\]

\[
p_t(x_t) = \mathcal{N}^{-1}(x_t \mid \vartheta_t, \Lambda_t)
\]

Computing *marginal mean and covariance* from messages:

\[
\Lambda_t = \sum_{s \in \Gamma(t)} \Lambda_{st} \quad \vartheta_t = \sum_{s \in \Gamma(t)} \vartheta_{st}
\]
Gaussian Belief Propagation

- Compute message mean & covar as algebraic function of incoming message mean & covar (generalizes Kalman)
- For tree of $N$ nodes of dimension $d$, cost is $O(Nd^3)$

Computing *marginal mean and covariance* from messages:

\[ \Lambda_t = \sum_{s \in \Gamma(t)} \Lambda_{st} \quad \nu_t = \sum_{s \in \Gamma(t)} \nu_{st} \]
Learning : Gaussian LDS

Probability Model

\[ x_t \mid x_{t-1} \sim \mathcal{N}(Fx_{t-1}, \Sigma) \]
\[ y_t \mid x_t \sim \mathcal{N}(Hx_t, R) \]

Learning Model Parameters

Given observations \( \{y_1, y_2, \ldots, y_T\} \) across all timesteps, maximize log-marginal likelihood:

\[
\max_{F, H, \Sigma, R} \log p(y_1, \ldots, y_T \mid F, H, \Sigma, R)
\]

Problem We do not know the latent states \( \{x_1, x_2, \ldots, x_T\} \). How to find maximum likelihood estimates?

Kalman Filter
The Kalman filter *exactly* marginalizes latent state, e.g. at time $t=1$:

\[
\begin{align*}
\text{( Law of total Probability )} & \quad p(y_1) = \int p(x_1)p(y_1 | x_1) \, dx_1 \\
\text{( LDS Model )} & \quad = \int \mathcal{N}(x_1 | \mu_0, P_0)\mathcal{N}(y_1 | Hx_1, R) \, dx_1 \\
\text{( Gaussian Marginal )} & \quad = \mathcal{N}(y_1 | H\mu_0, R + HP_0H^T)
\end{align*}
\]

For 2 timesteps we have:

\[
\text{( Probability Chain Rule )} \quad p(y_1, y_2) = p(y_1)p(y_2 | y_1)
\]

Just did this \hspace{2cm} Let's compute this
Learning : Gaussian LDS

Conditional likelihood at time $t=2$:

$$p(y_2 \mid y_1) = \int_{\mathcal{X}_2} \int_{\mathcal{X}_1} p(x_1, x_2 \mid y_1) p(y_2 \mid x_2) \, dx_1 \, dx_2$$

(Chain rule and $x_2 \perp y_1 \mid x_1$)

$$= \int_{\mathcal{X}_2} \int_{\mathcal{X}_1} p(x_1 \mid y_1) p(x_2 \mid x_1) p(y_2 \mid x_2) \, dx_1 \, dx_2$$

(LDS Model)

$$= \int_{\mathcal{X}_2} \int_{\mathcal{X}_1} p(x_1 \mid y_1) \mathcal{N}(x_2 \mid F x_1, \Sigma) \mathcal{N}(y_2 \mid H x_2, R) \, dx_1 \, dx_2$$

Kalman filter distribution at time $t=1$ (Gaussian)

We can compute these integrals using Gaussian formulas

**Surprise**, $p(y_2 \mid y_1)$ is Gaussian
Using probability chain rule we can write log-marginal likelihood as,

$$\max_{\theta} \log p(y_1^T \mid \theta) = \max_{\theta} \log p(y_1 \mid \theta) + \sum_{t=1}^{T} \log p(y_t \mid y_1^{t-1}, \theta)$$

Where $\theta = \{F, H, \Sigma, R\}$ and $y_1^T = \{y_1, \ldots, y_T\}$

• Every term is Gaussian
• We can compute every term in closed-form using Kalman updates
• Directly maximizing above w.r.t. parameters is cumbersome in this form

Using Expectation Maximization turns is much easier
Expectation Maximization: Gaussian LDS

Recall the EM lower bound of the log-marginal likelihood:

$$\max_{\theta} \log p(y_1^T \mid \theta) \geq \max_{q, \theta} \mathbb{E}_q \left[ \log \frac{p(x_1^T, y_1^T \mid \theta)}{q(x_1^T)} \right] \equiv \mathcal{L}(q, \theta)$$

Initialize Parameters: $\theta^{(0)}$

At iteration $t$ do:

**E-Step:** $q^{(t)}(z) = p(z \mid y, \theta^{(t-1)})$

**M-Step:** $\theta^{(t)} = \arg \max_{\theta} \mathcal{L}(q^{(t)}, \theta)$

Until convergence
EM : Gaussian LDS

**E-Step** Compute *expected complete data log-likelihood*,

\[
\mathbb{E} \left[ \log p(x_1^T, y_1^T \mid \theta) \right] = \text{Expectation taken w.r.t. posterior } p(x_1^T, y_1^T \mid \theta^{\text{old}})
\]

\[
= \mathbb{E} \left[ \log p(x_1) + p(y_1 \mid x_1, \theta) + \sum_{t=2}^{T} \log p(x_t \mid x_{t-1}, \theta) + \log p(y_t \mid x_t, \theta) \right]
\]

\[
= \mathbb{E} \left[ \log \mathcal{N}(x_1 \mid \mu_0, \Sigma_0) + \log \mathcal{N}(y_1 \mid Hx_1, R) \right. \\
\left. + \sum_{t=2}^{T} \log \mathcal{N}(x_t \mid Fx_{t-1}, \Sigma) + \log \mathcal{N}(y_t \mid Hx_t, R) \right]
\]

\[
= \ldots \text{ Algebra left for exercise } \ldots
\]
M-Step Update estimate of the parameters,

\[ \theta^{\text{new}} = \arg \max_\theta \mathbb{E} \left[ \log p(x_1^T, y_1^T \mid \theta) \right] \]

- E-step + algebra = expected complete data log-like likelihood
- Solve for zero-gradient conditions of parameters,

\[ \theta^{\text{new}} = \{ F^{\text{new}}, H^{\text{new}}, \Sigma^{\text{new}}, R^{\text{new}} \} \]

- We won’t go through calculations here (they are somewhat tedious)
Linear dynamical system,

\[ x_t \mid x_{t-1} \sim \mathcal{N}(Fx_{t-1}, \Sigma) \quad \text{Linear Gaussian Dynamics} \]

\[ y_t \mid x_t \sim \mathcal{N}(Hx_t, R) \quad \text{Linear Gaussian Observation} \]

State-space representation same as linear regression,

\[ x_t = Fx_{t-1} + \epsilon \quad \text{where} \quad \epsilon \sim \mathcal{N}(0, \Sigma) \]

Exact posterior inference via Kalman filter

• Recursively pass marginal moments
• Two steps: prediction, measurement update
• Forward pass filtering, backward pass smoothing

Kalman is special case of Gaussian sum-product BP
• Sequence Models

• Linear Dynamical Systems

• LDS Extensions
Nonlinear Dynamical System

Pendulum with mass $m=1$, pole length $L=1$:

$$x_t = \begin{pmatrix} \theta_t \\ \dot{\theta}_t \end{pmatrix} = \begin{pmatrix} \theta_{t-1} + \dot{\theta}_{t-1} \\ \dot{\theta}_{t-1} - g \sin(\theta_{t-1}) \end{pmatrix} + \epsilon$$

$$y_t = \sin(\theta_t) + \omega$$

Nonlinear dynamics / measurement:

$$x_t = f(x_{t-1}) + \epsilon \sim \mathcal{N}(0, \Sigma)$$

$$y_t = h(x_t) + \omega \sim \mathcal{N}(0, R)$$
Linearizing a Nonlinear Function

Suppose that we have a nonlinear function:

\[ h(\theta) = \sin(\theta) \]

**Taylor series representation**

- Choose any evaluation point \( \theta_0 \)
- Represent via successive derivatives, evaluated at \( \theta_0 \)

\[
h(\theta) = h(\theta_0) + \frac{h'(\theta_0)}{1!}(\theta - \theta_0) + \frac{h''(\theta_0)}{2!}(\theta - \theta_0)^2 + \frac{h'''(\theta_0)}{3!}(\theta - \theta_0)^3 + \ldots
\]

Infinite series holds with equality

Linear approximation discards higher-order derivatives:

\[
h(\theta) \approx h(\theta_0) + \frac{h'(\theta_0)}{1!}(\theta - \theta_0)
\]
Linearizing Vector-Valued Functions

• Let $f : \mathbb{R}^N \rightarrow \mathbb{R}^M$ be a vector-valued function.

• Linear approximation about $a$ is given by:

$$f(x) \approx f(a) + \mathbf{J}(a)(x - a)$$

• $\mathbf{J}(a)$ is the Jacobian matrix of partial derivatives (evaluated at $a$)

$$\mathbf{J} = \begin{pmatrix}
\frac{\partial f_1}{\partial x_1} & \frac{\partial f_1}{\partial x_2} & \cdots & \frac{\partial f_1}{\partial x_N} \\
\frac{\partial f_2}{\partial x_1} & \frac{\partial f_2}{\partial x_2} & \cdots & \frac{\partial f_2}{\partial x_N} \\
\vdots & \vdots & \ddots & \vdots \\
\frac{\partial f_M}{\partial x_1} & \frac{\partial f_M}{\partial x_2} & \cdots & \frac{\partial f_M}{\partial x_N}
\end{pmatrix}
$$

• Partial derivatives of each output dim w.r.t each input dim
• First dim. Matches function output, second dim. Input
• All partials will be evaluated at chosen point $a$
• Thus function is approximation about the point $a$
Nonlinear Dynamical System

Filter equations lack a closed-form:

**Prediction:**

\[ p(x_t \mid y_{1}^{t-1}) = \int \mathcal{N}(x_t \mid f(x_{t-1}), \Sigma)p(x_{t-1} \mid y_{1}^{t-1}) \, dx_{t-1} \]

**Measurement Update:**

\[ p(x_t \mid y_{1}^{t}) \propto \mathcal{N}(y_t \mid h(x_t), R)p(x_t \mid y_{1}^{t-1}) \]

**Idea** Linearize \( f(.) \) and \( h(.) \) about a point \( m \):

\[
\begin{align*}
  f(x) &\approx f(m) + J_f(m)(x - m) \\
  h(x) &\approx h(m) + J_h(m)(x - m)
\end{align*}
\]

where \( J_f = \left( \frac{\partial f_i}{\partial x_j} \right) \) is Jacobian matrix of partials.

1st Order Taylor expansion

\[
\sin(\theta)
\]

- \(-\frac{1}{2} \pi\)
- \(\frac{1}{2} \pi\)
Extended Kalman Filter

1. Linearize $f(.)$ and $h(.)$ about filter mean
2. Assume linear Gaussian model
3. Do standard Kalman updates

Example Linearization of pendulum model

$$x_t = \begin{pmatrix} \theta_t \\ \dot{\theta}_t \end{pmatrix} = \begin{pmatrix} \theta_{t-1} + \dot{\theta}_{t-1} \\ \dot{\theta}_{t-1} - g \sin(\theta_{t-1}) \end{pmatrix} + \epsilon$$

$$y_t = \underbrace{\sin(\theta_t)}_{h(x_t)} + \omega$$

$$J_f(x) = \begin{pmatrix} 1 & 1 \\ -g \cos(x^1) & 1 \end{pmatrix} \quad J_h(x) = (\cos(x^1), 0)$$
EKF Update Equations

Prediction Step:

State Prediction: $\mu_{t|t-1} = f(\mu_{t-1|t-1})$

Covariance Prediction:

$\Sigma_{t|t-1} = \Sigma + J_f(\mu_{t-1|t-1})\Sigma_{t-1|t-1}J_f(\mu_{t-1|t-1})^T$

Measurement Update Step:

Gain: $K_t = \Sigma_{t|t-1}J_h(\mu_{t|t-1})^T(J_h(\mu_{t|t-1})\Sigma_{t|t-1}J_h(\mu_{t|t-1})^T + R)^{-1}$

Filter Covariance: $\Sigma_{t|t} = \Sigma_{t|t-1} - K_tJ_h(\mu_{t|t-1})\Sigma_{t|t-1}$

Filter Mean: $\mu_{t|t} = \mu_{t|t-1} + K_t(y_t - h(\mu_{t|t-1}))$
**Extended Kalman Filter**

**PROS:**
- Easy to implement – updates analogous to standard Kalman
- Computationally efficient
- Known theoretical stability results

**CONS:**
- Linearity assumption poor for highly nonlinear models
- Requires model differentiability
- Jacobian matrices can be hard to calculate & implement

*Unscented Kalman filter (UKF) typically more accurate in practice*
Other Nonlinear Filtering Options

Key issue is approximating integrals:

\[ p(x_t \mid y_{1}^{t-1}) = \int p(x_t \mid x_{t-1}) p(x_{t-1} \mid y_{1}^{t-1}) \, dx_{t-1} \]

Dynamics Filter at t-1

\[ = \int \mathcal{N}(x_t \mid f(x_{t-1}), Q) p(x_{t-1} \mid y_{1}^{t-1}) \, dx_{t-1} \]

**Option 1**: Use Gaussian quadrature → *Unscented Kalman Filter (UKF)*

- Approximates integral using deterministic control points
- Tends to be more accurate than EKF

**Option 2**: Sample-based approximation → *Particle Filter (PF)*

- Draw samples from filter \( \{x_{t-1}^{(i)}\}_{i=1}^{M} \sim p(x_{t-1} \mid y_{1}^{t-1}) \)
- Monte Carlo approximation of integral using samples,

\[ p(x_t \mid y_{1}^{t-1}) \approx \mathbb{E}_{\{x_{t-1}^{(i)}\}} [\mathcal{N}(x_t \mid f(x_{t-1}), Q)] \]

We will cover this…
Dynamic Bayesian Networks

- Multivariate latent state (e.g. $x \in \mathbb{R}^3$)
- Dynamics for each component within and across time
- Sometimes used as catch-all term for dynamical systems

Caution: DBN terminology is somewhat vague and overloaded (e.g. Deep Belief Net)
Switching Linear Dynamical System

Discrete switching state:
\[ z_t \mid z_{t-1} \sim \text{Cat}(\pi(z_{t-1})) \quad \text{With stochastic transition matrix } \pi \]

Switching state selects linear dynamics:
\[ x_t \mid x_{t-1} \sim \mathcal{N}(A_{z_t} x_{t-1}, \Sigma_{z_t}) \quad \text{(e.g. Linear Gaussian)} \]
We can do sum-product for HMM and LDS, so maybe we can do it for SLDS...

Forward message,

\[
m_{t-1,t}(z_t, x_t) \propto \int \sum_{z_{t-1}} m_{t-2,t-1}(z_{t-1}, x_{t-1}) p(y_{t-1} \mid x_{t-1})
\]

\[
p(z_t \mid z_{t-1}) p(x_t \mid x_{t-1}, z_t) dx_{t-1}
\]

\[
= \int \sum_{z_{t-1}} m_{t-2,t-1}(z_{t-1}, x_{t-1}) \mathcal{N}(y_{t-1} \mid H x_{t-1}, R) \text{Cat}(z_t \mid \pi(z_{t-1})) \mathcal{N}(x_t \mid A_{zt} x_{t-1}, \Sigma_{zt}) dx_{t-1}
\]

- Message is Gaussian mixture over K states (for some K)
- But incoming message is also a Gaussian mixture
- Number of components (K) grows exponentially with time \(t\)
Summary

• Linear dynamical system is time-extension of PCA,

\[ x_t \mid x_{t-1} \sim \mathcal{N}(Fx_{t-1}, \Sigma) \quad y_t \mid x_t \sim \mathcal{N}(Hx_t, R) \]

• Exact inference via Kalman filtering,

Prediction step updates moments of predictive distribution \( p(x_t \mid y_{1}^{t-1}) \)

Measurement step updates filter distribution with newest measurement \( p(x_t \mid y_1^t) \)

• Kalman filter is special case of Gaussian belief propagation

Messages represented as Gaussian natural parameters
Summary

• Nonlinear state-space models allow more complex dynamics,

\[ x_t = \begin{pmatrix} \theta_t \\ \dot{\theta}_t \end{pmatrix} = \begin{pmatrix} \theta_{t-1} + \dot{\theta}_{t-1} \\ \dot{\theta}_{t-1} - g \sin(\theta_{t-1}) \end{pmatrix} + \epsilon / f(x_{t-1}) \]

\[ y_t = \sin(\theta_t) + \omega / h(x_t) \]

Approximate Kalman filter inference via linearization (Extended Kalman Filter) or Gaussian quadrature (Unscented Kalman Filter)

• Switching state-space model represents discrete & continuous states,

Exact inference intractable due to exponential growth in message parameters

Both nonlinear and SSMs can be addressed using sample-based methods (e.g. Particle Filtering) as we will see