Hidden Markov Models and Kalman Filters

CS536: Probabilistic Graphical Models

Li-Ping Liu

liuli@eecs

Feb. 5, 2016

Original slides by Alex Simma and Erik Sudderth
Speech Recognition

• Given an audio waveform, would like to robustly extract & recognize any spoken words

• Statistical models can be used to
  ➢ Provide greater robustness to noise
  ➢ Adapt to accent of different speakers
  ➢ Learn from training

S. Roweis, 2004
Target Tracking

Radar-based tracking of multiple targets

Visual tracking of articulated objects
(L. Sigal et. al., 2006)

- Estimate motion of targets in 3D world from indirect, potentially noisy measurements
Analysis of Sequential Data

• Sequential structure arises in a huge range of applications
  - Repeated measurements of a temporal process
  - Online decision making & control
  - Text, biological sequences, etc

• Standard machine learning methods are often difficult to directly apply
  - Do not exploit temporal correlations
  - Computation & storage requirements typically scale poorly to realistic applications
Outline

Introduction to Sequential Processes

- Markov chains
- Hidden Markov models

Discrete-State HMMs

- Inference: Filtering, smoothing, Viterbi, classification

Continuous-State HMMs

- Linear state space models: Kalman filters
- Nonlinear dynamical systems: Particle filters
Sequential Processes

• Consider a system which can occupy one of $N$ discrete *states* or *categories*

\[ x_t \in \{1, 2, \ldots, N\} \rightarrow \text{state at time } t \]

• We are interested in *stochastic* systems, in which state evolution is random

• Any *joint* distribution can be factored into a series of *conditional* distributions:

\[
p(x_0, x_1, \ldots, x_T) = p(x_0) \prod_{t=1}^{T} p(x_t \mid x_0, \ldots, x_{t-1})
\]
Markov Processes

• For a *Markov* process, the next state depends only on the current state:

\[ p(x_{t+1} \mid x_0, \ldots, x_t) = p(x_{t+1} \mid x_t) \]

• This property in turn implies that

\[ p(x_0, \ldots, x_{t-1}, x_t, x_{t+1}, \ldots, x_T \mid x_t) = p(x_0, \ldots, x_{t-1} \mid x_t)p(x_{t+1}, \ldots, x_T \mid x_t) \]

“**Conditioned on the present, the past & future are independent**”
State Transition Matrices

• A \textit{stationary} Markov chain with \(N\) states is described by an \(N\times N\) \textit{transition matrix}:

\[
Q = \begin{bmatrix}
q_{11} & q_{12} & q_{13} \\
q_{21} & q_{22} & q_{23} \\
q_{31} & q_{32} & q_{33}
\end{bmatrix}
\]

\[q_{ij} \triangleq p(x_{t+1} = i \mid x_t = j)\]

• Constraints on valid transition matrices:

\[q_{ij} \geq 0\]

\[
\sum_{i=1}^{N} q_{ij} = 1 \quad \text{for all } j
\]
State Transition Diagrams

\[ q_{ij} \triangleq p(x_{t+1} = i \mid x_t = j) \]

\[
Q = \begin{bmatrix}
0.5 & 0.1 & 0.0 \\
0.3 & 0.0 & 0.4 \\
0.2 & 0.9 & 0.6 \\
\end{bmatrix}
\]

- Think of a particle randomly following an arrow at each discrete time step
- Most useful when \( N \) small, and \( Q \) sparse
Markov Chains: Graphical Models

\[ p(x_0, x_1, \ldots, x_T) = p(x_0) \prod_{t=1}^{T} p(x_t \mid x_{t-1}) \]

- Graph interpretation differs from state transition diagrams:
  - **nodes**: state values at particular times
  - **edges**: Markov properties

\[
Q = \begin{bmatrix}
0.5 & 0.1 & 0.0 \\
0.3 & 0.0 & 0.4 \\
0.2 & 0.9 & 0.6
\end{bmatrix}
\]
Embedding Higher-Order Chains

\[ p(x_0, x_1, \ldots, x_T) = p(x_0) \prod_{t=1}^{T} p(x_t \mid x_{t-1}, x_{t-2}) \]

- Each new state depends on fixed-length window of preceding state values
- We can represent this as a first-order model via state augmentation:

\[ \bar{x}_t \triangleq \{x_t, x_{t-1}\} \quad p(\bar{x}) = p(\bar{x}_1) \prod_{t=2}^{T} p(\bar{x}_t \mid \bar{x}_{t-1}) \]

\( (N^2 \text{ augmented states}) \)
Continuous State Processes

• In many applications, it is more natural to define states in some continuous, Euclidean space: \( x_t \in \mathbb{R}^d \)

\[
p(x_0) \xrightarrow{p(x_1 | x_0)} x_0 \xrightarrow{p(x_2 | x_1)} x_1 \xrightarrow{p(x_3 | x_2)} x_2 \xrightarrow{q(x_t | x_{t-1}; \theta)} x_3
\]

• Examples: stock price, aircraft position, …
Hidden Markov Models

• Few realistic time series directly satisfy the assumptions of Markov processes:

  "Conditioned on the present, the past & future are independent"

• Motivates hidden Markov models (HMM):

\[
p(x_0, x_1, \ldots, x_T) = p(x_0) \prod_{t=1}^{T} p(x_t | x_{t-1})p(y_t | x_t)
\]
• Given \( x_t \), earlier observations provide no additional information about the future:

\[
p(y_t, y_{t+1}, \ldots \mid x_t, y_{t-1}, y_{t-2}, \ldots) = p(y_t, y_{t+1}, \ldots \mid x_t)
\]

• Transformation of process under which dynamics take a simple, first-order form
Where do states come from?

- **Analysis of a physical phenomenon:**
  - Dynamical models of an aircraft or robot
  - Geophysical models of climate evolution

- **Discovered from training data:**
  - Recorded examples of spoken English
  - Historic behavior of stock prices
Outline

Introduction to Sequential Processes

- Markov chains
- Hidden Markov models

Discrete-State HMMs

- Inference: Filtering, smoothing, Viterbi, classification

Continuous-State HMMs

- Linear state space models: Kalman filters
- Nonlinear dynamical systems: Particle filters
Discrete State HMMs

\[ x_t \in \{1, 2, \ldots, N\} \]

- Associate each of the \( N \) hidden states with a different observation distribution:
  \[ p(y_t \mid x_t = 1) \quad p(y_t \mid x_t = 2) \quad \cdots \]
- Observation densities are typically chosen to encode domain knowledge
Discrete HMMs: Observations

Discrete Observations

\[ y_t \in \{1, 2, \ldots, M\} \]

\[
p(y_t \mid x_t = 1) = \begin{bmatrix} 0.3 \\ 0.1 \\ 0.5 \\ 0.1 \end{bmatrix} \quad p(y_t \mid x_t = 2) = \begin{bmatrix} 0.2 \\ 0.2 \\ 0.1 \\ 0.5 \end{bmatrix}
\]

Continuous Observations

\[ y_t \in \mathbb{R}^k \]

\[ p(y_t \mid x_t = 1) \]
\[ p(y_t \mid x_t = 2) \]
Specifying an HMM

- Observation model: $P(y_i|x_i)$
- Transition model: $P(x_i|x_{i-1})$
- Initial state distribution: $P(x_0)$
Gilbert-Elliott Channel Model

Hidden State:

\[ x_t \in \{0, 1\} \]

\[ Q = \begin{bmatrix} 1 - \epsilon & \epsilon \\ \epsilon & 1 - \epsilon \end{bmatrix} \]

Observations:

\[ y_t \sim \mathcal{N}(0, \sigma^2_{x_t}) \]

\[ \sigma^2_0 = \text{small} \]

\[ \sigma^2_1 = \text{large} \]

Simple model for correlated, bursty noise

(Elliott, 1963)
Discrete HMMs: Inference

- In many applications, we would like to infer hidden states from observations.

- Suppose that the cost incurred by an estimated state sequence decomposes:

$$C(x, \hat{x}) = \sum_{t=0}^{T} C_t(x_t, \hat{x}_t)$$

  \(x_t \rightarrow \text{true state}\)

  \(\hat{x}_t \rightarrow \text{estimated state}\)

- The expected cost then depends only on the posterior marginal distributions:

$$\mathbb{E}[C(x, \hat{x}) | y] = \sum_{t=0}^{T} \sum_{x_t} C_t(x_t, \hat{x}_t) p(x_t | y)$$
Filtering & Smoothing

• For online data analysis, we seek *filtered* state estimates given earlier observations:
  \[ p(x_t | y_1, y_2, \ldots, y_t) \quad t = 1, 2, \ldots \]

• In other cases, find *smoothed* estimates given earlier and later of observations:
  \[ p(x_t | y_1, y_2, \ldots, y_T) \quad t = 1, 2, \ldots, T \]

• Lots of other alternatives, including *fixed-lag smoothing & prediction*:
  \[ p(x_t | y_1, \ldots, y_{t+\delta}) \quad p(x_t | y_1, \ldots, y_{t-\delta}) \]
Markov Chain Statistics

\[ p(x_0) \xrightarrow{p(x_1 \mid x_0)} x_0 \xrightarrow{p(x_2 \mid x_1)} x_1 \xrightarrow{p(x_3 \mid x_2)} x_2 \xrightarrow{p(x_3 \mid x_2)} x_3 \]

\( \alpha_t \triangleq \begin{bmatrix} p(x_t = 1) \\ p(x_t = 2) \\ p(x_t = 3) \end{bmatrix} \)

\[
Q = \begin{bmatrix}
q_{11} & q_{12} & q_{13} \\
q_{21} & q_{22} & q_{23} \\
q_{31} & q_{32} & q_{33}
\end{bmatrix}
\]

\( q_{ij} \triangleq p(x_{t+1} = i \mid x_t = j) \)

- By definition of conditional probabilities,

\[
\alpha_1(i) = \sum_{j=1}^{N} q_{ij} \alpha_0(j)
\]

\[
\begin{align*}
\alpha_1 &= Q \alpha_0 \\
\alpha_t &= Q^t \alpha_0 \\
\alpha_\infty &= ???
\end{align*}
\]
Discrete HMMs: Filtering

\[ \alpha_t(x_t) \triangleq p(x_t \mid y_1, \ldots, y_t) \]

\[ = \frac{1}{Z_t} p(y_t \mid x_t) \sum_{x_{t-1}} p(x_t \mid x_{t-1}) \alpha_{t-1}(x_{t-1}) \]

Normalization constant

Prediction: \( p(x_t \mid y_1, \ldots, y_{t-1}) \)

Update: \( p(x_t \mid y_1, \ldots, y_t) \)

Incorporates \( T \) observations in \( \mathcal{O}(TN^2) \) operations
Discrete HMMs: Smoothing

\[ p(x_t \mid y) \propto p(x_t \mid y_1, \ldots, y_t)p(y_{t+1}, \ldots, y_T \mid x_t) \]

\[ \alpha_t(x_t) \quad \beta_t(x_t) \]

- The forward-backward algorithm updates filtering via a reverse-time recursion:

\[
\beta_t(x_t) = \frac{1}{Z_t} \sum_{x_{t+1}} p(x_{t+1} \mid x_t)p(y_{t+1} \mid x_{t+1})\beta_{t+1}(x_{t+1})
\]
Optimal State Estimation

\[ p(x_t | y) = \frac{1}{Z_t} \alpha_t(x_t) \beta_t(x_t) \]

• Probabilities measure the posterior confidence in the true hidden states

• The posterior mode minimizes the number of incorrectly assigned states:

\[ C(x, \hat{x}) = T - \sum_{t=1}^{T} \delta(x_t, \hat{x}_t) \]

• What about the state sequence with the highest joint probability?

Bit or symbol error rate

Word or sequence error rate
**Viterbi Algorithm**

\[ \hat{x} = \operatorname{arg\ max}_x p(x_0, x_1, \ldots, x_T \mid y_1, \ldots, y_T) \]

- Use *dynamic programming* to recursively find the probability of the most likely state sequence ending with each \( x_t \in \{1, \ldots, N\} \)

\[ \gamma_t(x_t) \triangleq \max_{x_1, \ldots, x_{t-1}} p(x_1, \ldots, x_{t-1}, x_t \mid y_1, \ldots, y_t) \]

\[ \propto p(y_t \mid x_t) \cdot \left[ \max_{x_{t-1}} p(x_t \mid x_{t-1}) \gamma_{t-1}(x_{t-1}) \right] \]

- A reverse-time, *backtracking* procedure then picks the maximizing state sequence...
Time Series Classification

• Suppose I’d like to know which of 2 HMMs best explains an observed sequence

• This \textit{classification} is optimally determined by the following log-likelihood ratio:

\[
\log \frac{p(y_1, \ldots, y_T \mid \mathcal{M}_1)}{p(y_1, \ldots, y_T \mid \mathcal{M}_0)} = \log \frac{p(y \mid \mathcal{M}_1)}{p(y \mid \mathcal{M}_0)}
\]

\[
\log p(y \mid \mathcal{M}_i) = \log \sum_x p(y \mid x, \mathcal{M}_i)p(x \mid \mathcal{M}_i)
\]

• These log-likelihoods can be computed from filtering \textit{normalization constants}
Outline

Introduction to Sequential Processes

- Markov chains
- Hidden Markov models

Discrete-State HMMs

- Inference: Filtering, smoothing, Viterbi, classification
- Learning: EM algorithm

Continuous-State HMMs

- Linear state space models: Kalman filters
- Nonlinear dynamical systems: Particle filters

More on Graphical Models
Linear State Space Models

\[
x_{t+1} = Ax_t + w_t \\
y_t = Cx_t + v_t
\]

\[
w_t \sim \mathcal{N}(0, Q) \\
v_t \sim \mathcal{N}(0, R)
\]

• States & observations jointly Gaussian:
  - All marginals & conditionals Gaussian
  - Linear transformations remain Gaussian
Simple Linear Dynamics

**Brownian Motion**

\[ x_{t+1} = x_t + \omega_t \]

**Constant Velocity**

\[
\begin{bmatrix}
    x_{t+1} \\
    \delta_{t+1}
\end{bmatrix} =
\begin{bmatrix}
    1 & 1 \\
    0 & 1
\end{bmatrix}
\begin{bmatrix}
    x_t \\
    \delta_t
\end{bmatrix} + \omega_t
\]
Kalman Filter

\[ x_{t+1} = Ax_t + w_t \quad \text{with} \quad w_t \sim \mathcal{N}(0, Q) \]
\[ y_t = Cx_t + v_t \quad \text{with} \quad v_t \sim \mathcal{N}(0, R) \]

• Represent Gaussians by \textit{mean} & \textit{covariance}:

\[ p(x_t \mid y_1, \ldots, y_{t-1}) = \mathcal{N}(x; \tilde{\mu}_t, \tilde{\Lambda}_t) \]
\[ p(x_t \mid y_1, \ldots, y_t) = \mathcal{N}(x; \mu_t, \Lambda_t) \]

**Prediction:**

\[ \tilde{\mu}_t = A\mu_{t-1} \]
\[ \tilde{\Lambda}_t = A\Lambda_{t-1}A^T + Q \]

**Kalman Gain:**

\[ K_t = \tilde{\Lambda}_tC^T(C\tilde{\Lambda}_tC^T + R)^{-1} \]

**Update:**

\[ \mu_t = \tilde{\mu}_t + K_t(y_t - C\tilde{\mu}_t) \]
\[ \Lambda_t = \tilde{\Lambda}_t - K_tC\tilde{\Lambda}_t \]
Kalman Filtering as Regression

- The posterior mean minimizes the mean squared prediction error:

  \[ p(x_t \mid y_1, \ldots, y_t) = \mathcal{N}(x; \mu_t, \Lambda_t) \]

- The posterior mean minimizes the mean squared prediction error:

  \[ \mu_t = \arg \min_{\mu} \mathbb{E} \left[ \| x_t - \mu \|^2 \mid y_1, \ldots, y_t \right] \]

- The Kalman filter thus provides an optimal online regression algorithm
Constant Velocity Tracking

Kalman Filter

Kalman Smoother

(K. Murphy, 1998)
Nonlinear State Space Models

\[ x_{t+1} = f(x_t, w_t) \quad \quad w_t \sim \mathcal{F} \]
\[ y_t = g(x_t, v_t) \quad \quad v_t \sim \mathcal{G} \]

- State dynamics and measurements given by potentially complex nonlinear functions
- Noise sampled from non-Gaussian distributions
Examples of Nonlinear Models

Observed image is a complex function of the 3D pose, other nearby objects & clutter, lighting conditions, camera calibration, etc.

Dynamics implicitly determined by geophysical simulations
Nonlinear Filtering

\[ p(x_t \mid y_1, \ldots, y_{t-1}) = \tilde{q}_t(x_t) \]

\[ p(x_t \mid y_1, \ldots, y_t) = q_t(x_t) \]

**Prediction:**

\[ \tilde{q}_t(x_t) = \int p(x_t \mid x_{t-1}) q_{t-1}(x_{t-1}) \, dx_{t-1} \]

**Update:**

\[ q_t(x_t) = \frac{1}{Z_t} \tilde{q}_t(x_t) p(y_t \mid x_t) \]
Approximate Nonlinear Filters

\[ q_t(x_t) \propto p(y_t \mid x_t) \cdot \int p(x_t \mid x_{t-1}) q_{t-1}(x_{t-1}) \, dx_{t-1} \]

- Typically cannot directly represent these continuous functions, or determine a closed form for the prediction integral
- A wide range of approximate nonlinear filters have thus been proposed, including
  - Histogram filters
  - Extended & unscented Kalman filters
  - Particle filters
  - ...
Nonlinear Filtering Taxonomy

**Histogram Filter:**
- Evaluate on fixed discretization grid
- Only feasible in low dimensions
- Expensive or inaccurate

**Extended/Unscented Kalman Filter:**
- Approximate posterior as Gaussian via linearization, quadrature, …
- Inaccurate for multimodal posterior distributions

**Particle Filter:**
- Dynamically evaluate states with highest probability
- Monte Carlo approximation
Question 1

• Think about the key roles of two distributions: \( p(x_t \mid y_1, \ldots, y_t) \) and \( p(y_{t+1}, \ldots, y_T \mid x_t) \)

What properties of the two functions make general HMM hard?

- Whether the two functions can be represented easily.
- Whether the two functions can be propagate to the next \( t \)?
- Multinomial distributions and Gaussian distributions are easy for both
Question 2

• HMM is obtained by increasing the complexity of Markov Chain. How to further increase the complexity of HMM to model more problems?
  ➢ Increase hidden layers