Learning and Intelligent Systems

Sequential (time-series) models

Prof. Thomas Hofmann
Data Analytics Laboratory (da.inf.ethz.ch)
Slides: Andreas Krause
Gaussian non-linear time series

- We might more generally assume

\[
P(Y_{t+1} = y \mid y_{t-k+1}, \ldots, y_t) = \mathcal{N}(y; f(y_{t-k+1}, \ldots, y_t; \theta), \sigma^2)
\]

for some (possibly nonlinear, multivariate) function \( f \)

- This is equivalent to assuming

\[
y_{t+1} = f(y_{t-k+1}, \ldots, y_t; \theta) + \epsilon_t
\]

where \( \epsilon_t \sim \mathcal{N}(0, \sigma^2) \)

- For example, might assume \( f(\cdot; \theta) \) is specified as a neural network with weights given by \( \theta \)
- Can train via backpropagation / SGD
Non-Gaussian time series models

- Can replace the Gaussian likelihood by a different one.
- E.g., if all $Y_t$ are binary, might use Bernoulli likelihood

$$P(Y_{t+1} = 1 \mid y_{t-k+1}, \ldots, y_t) = \frac{1}{1 + \exp(-f(y_{t-k+1}, \ldots, y_t; \theta))}$$

- Can train using stochastic gradient descent
Predicting multiple time steps

- Suppose we fit a model

\[
P(Y_{t+1} = y \mid y_{t-k+1}, \ldots, y_t) = \mathcal{N}(y; f(y_{t-k+1}, \ldots, y_t; \theta), \sigma^2)
\]

- How about predicting multiple time steps ahead?

\[
P(Y_{t+\tau} = y \mid y_{t-k+1}, \ldots, y_t) = \mathcal{N}(y; f(y_{t-k+1}, \ldots, y_t; \theta), \sigma^2)
\]

- What if variance is 0?

\[
\Rightarrow y_{t+1} = f(y_t) = f(y_t; \theta)
\]

\[
y_{t+2} = f(y_{t+1}) = f(f(y_t))
\]
Uncertainty in prediction

\[ P(Y_{t+1} = y \mid y_t) = \mathcal{N}(y; f(y_t), \sigma^2) \]

\[ P(Y_{t+2} \mid y_t) = \int P(Y_{t+2} \mid y_{t+1}) P(y_{t+1} \mid y_t) \, dy_{t+1} \]

In general, integral intractable;

\[ P(Y_{t+2} \mid y_t) \text{ is not Gaussian} \]
Predicting multiple time steps

- In general, computing exact predictions (as possible in Markov models) is intractable.
- However, can approximate it, e.g., via sampling from the model.

\[
\text{Draw samples } y^{(1)}_{t+1:t+2}, \ldots, y^{(N)}_{t+1:t+2} \\
\text{Then compute, e.g., } \mathbb{E}[Y_{t+2}] \approx \frac{1}{N} \sum_{i=1}^{N} y^{(i)}_{t+2} 
\]
Sample approximations of expectations

- $x_1, ..., x_N, ...$ independent samples from $P(X)$
- (Strong) Law of large numbers:

$$\mathbb{E}_P[f(X)] = \lim_{N \to \infty} \frac{1}{N} \sum_{i=1}^{N} f(x_i)$$

Hereby, the convergence is with probability 1 (almost sure convergence)

- Suggests approximation using finite samples:

$$\mathbb{E}_P[f(X)] \approx \frac{1}{N} \sum_{i=1}^{N} f(x_i)$$
How many samples do we need?

- **Hoeffding’s inequality**
  
  Suppose $f$ is bounded in $[0,C]$. Then
  
  $$P\left(\left|\mathbb{E}_P[f(X)] - \frac{1}{N} \sum_{i=1}^N f(x_i)\right| > \varepsilon\right) \leq 2 \exp\left(-2N\varepsilon^2/C^2\right)$$

- Thus, probability of error decreases exponentially in $N$!
Forward sampling algorithm

**Input:** model $P(Y_{t+1} \mid Y_{t-k+1}, \ldots, Y_t)$, data $y_{1:T}$, prediction horizon $\tau \in \mathbb{N}$, #samples $N$

For $i=1:N$

- Set $y_{1:T}^{(i)} = y_{1:T}$
- For $t = T : T + \tau - 1$ do

  Sample $y_{t+1}^{(i)} \sim P(Y_{t+1} \mid y_{t-k+1}^{(i)}, \ldots, y_t^{(i)})$

Then, e.g., predict mean (for continuous Y) by

$$\mathbb{E}[Y_{T+\tau}] \approx \frac{1}{N} \sum_{i=1}^{N} y_{T+\tau}^{(i)}$$
Model selection

- Wait.. If time-series modeling can be reduced to standard (iid) supervised learning, what about model selection?
- In particular, how can we implement cross-validation?
Temporal cross-validation

Partition data into $r+1$ fold, respecting time order
I.e., $D_1$ contains the first $T/(r+1)$ data points, $D_2$ the second etc.

For $i=2:k+1$ do
- Train on folds $D_1, ..., D_{i-1}$, and test on fold $D_i$ only
- Report average over the $k$ performance estimates
Sequence prediction (with bounded memory) can be reduced to supervised learning

Can use all the standard tools we learned in this class!

For the special case of Markov models, prediction can be done recursively (via matrix multiplication)

For the general case, can try to fit a parametric model, and then produce samples from the model

Can do model selection via temporal cross-validation
Dynamic models with hidden variables

- So far, have only considered time series models involving the „labels“ $Y$
- Before discussing time series, we were concerned with estimating $P(Y,X)$, where labels $Y$ are hidden, and inputs $X$ are observed

- Is there a natural probabilistic model that generalizes this to time series?
You observe one sequence

\[ x_1, x_2, \ldots, x_t \]

Your goal is to predict another, related sequence:

\[ y_1, y_2, \ldots, y_t \]
Examples of sequence-to-sequence prediction

- Speech recognition
- Handwritten character recognition
- Tracking
- Map matching
- Activity recognition
- Language translation
- ...

...
**Hidden Markov Models (HMMs)**

- $Y_1, \ldots, Y_T$: Unobserved (hidden) variables (called **states**)
- $X_1, \ldots, X_T$: Observations
- **Types**: $Y_i$ categorical, $X_i$ categorical
HMMs for speech recognition

Words

\[ Y_1 \rightarrow Y_2 \rightarrow Y_3 \rightarrow Y_4 \rightarrow Y_5 \rightarrow Y_6 \]

Phoneme

\[ X_1 \rightarrow X_2 \rightarrow X_3 \rightarrow X_4 \rightarrow X_5 \rightarrow X_6 \]

“Machine learning is great”
An HMM is a generative model:

- Sample the starting state $y_1$ from a categorical distribution

\[ P(Y_1 = y) = p_y \]

- For $t=1:T-1$

  given $Y_t=y$, sample $X_t$

  \[ P(X_t = x \mid Y_t = y) = \eta_{x|y} \]

  given $Y_t=y'$, sample $Y_{t+1}$

  \[ P(Y_{t+1} = y \mid Y_t = y') = \theta_{y|y'} \]
Assumptions in HMMs

- Hidden sequence $Y_{1:t}$ forms (stationary) Markov Chain
  \[ P(Y_t | Y_{1:t-1}, X_{1:t-1}) = P(Y_t | Y_{t-1}) \]

- Emission probabilities do not depend on $t$
  \[ P(X_t = x | Y_t = y) = P(X_{t+1} = x | Y_{t+1} = y) \]

- Each observation $X_t$ is independent of all other variables given $Y_t$
  \[ P(X_t | Y_{1:t}, X_{1:t-1}) = P(X_t | Y_t) \]
Inference tasks in HMMs

Filtering

\[ P(Y_t \mid x_{1:t}) \]

Prediction

\[ P(Y_{t+\tau} \mid x_{1:t}) \]

Smoothing

\[ P(Y_t \mid x_{1:T}) \text{ for } 1 \leq t \leq T \]

Most probable explanation / seq. to sequence prediction

\[ y^*_{1:T} = \arg \max_{y_{1:T}} P(y_{1:T} \mid x_{1:T}) \]
Bayesian filtering

- Start with $P(Y_1)$
- At time $t$
  - Assume we have $P(Y_t \mid x_{1:t-1})$
  - Conditioning:
    \[
    P(Y_t = y \mid x_{1:t}) = \frac{P(Y_t = y \mid x_{1:t-1}) \eta_{x_t \mid y}}{\sum_{y'} P(Y_t = y' \mid x_{1:t-1}) \eta_{x_t \mid y'}}
    \]
- Prediction:
  \[
  P(Y_{t+1} = y \mid x_{1:t}) = \sum_{y'} P(Y_t = y' \mid x_{1:t}) \theta_{y \mid y'}
  \]
Matrix/vector notation of filtering

- Represent $P(Y_t | x_{1:t})$ as vector
  \[ p^{(t)} = [p_1^{(t)}, p_2^{(t)}, \ldots, p_c^{(t)}] \in \mathbb{R}^c \]

- Represent $P(Y_{t+1} = y \mid Y_t = y')$ as matrix $T \in \mathbb{R}^{c \times c}$
  \[ T_{y',y} = P(Y_{t+1} = y \mid Y_t = y') \]

- Represent $P(X_t = x_t \mid Y_t = y)$ as vector
  \[ o^{(t)} = [\eta_{x_t|1}, \ldots, \eta_{x_t|c}] \]

Then it holds that
\[ p^{(t+1)} \propto (p^{(t)} T) \cdot o^{(t+1)} \]
\[ \ll \text{remormalization} \]
\[ \ll \text{pointwise multiplication} \]
\[ \ll \text{prediction} \]
\[ \ll \text{conditioning} \]
Illustration: Bayesian filtering

The diagram illustrates the process of Bayesian filtering, showing the transition probabilities between different states. The network includes nodes representing weather conditions (Rain_0, Rain_1, Rain_2) and the corresponding prediction and conditioning values. The diagram also shows the connections and probabilities leading to the decision of whether to take an umbrella (Umbrella_1, Umbrella_2).
HMM for robot localization

(a) Posterior distribution over robot location after $E_1 = \text{NSW}$

(b) Posterior distribution over robot location after $E_1 = \text{NSW}, E_2 = \text{NS}$
Prediction in Hidden Markov Models

- **Given:** \( P(Y_t \mid X_{1:t}) \) [as vector \( \mathbf{p}^{(t)} \)]
- **Goal:** Compute \( P(Y_{t+\tau} \mid X_{1:t}) \) [as vector \( \mathbf{p}^{(t+\tau)} \)] by

\[
\mathbf{p}^{(t+\tau)} = \mathbf{p}^{(t)} T^{\tau}
\]
Smoothing in Hidden Markov Models

- **Given**: All observations (past and future): $X_{1:T}$
- **Goal**: Compute $P(Y_t \mid X_{1:T})$

We know how to condition on $x_{1:t}$. How can we condition on $x_{t+1:T}$?

\[
P(Y_t = y \mid X_{1:t}) = \frac{P(Y_t = y, X_{1:t})}{P(X_{1:t})} = \frac{P(Y_t = y, X_{1:t})}{P(X_{t+1:T} \mid Y_t = y, X_{1:t})} \cdot P(X_{t+1:T} \mid Y_t = y, X_{1:t})
\]

\[
= q_t(y) \cdot b_t(y)
\]

HMM property
Breaking up the computation

Define

\[ a_t(y) = P(Y_t = y, x_{1:t}) \]
\[ b_t(y) = P(x_{t+1:T} \mid Y_t = y) \]

Then it holds that

\[ P(Y_t = y \mid x_{1:T}) = \frac{1}{Z} a_t(y) b_t(y) \]

\[ Z = \sum_{y'} a_t(y') b_t(y') \]
Computing \( a_t \)

\[ a_t(y) = P(Y_t = y, x_{1:t}) \]

For \( t = 1 \),

\[ P(Y_1 = y_1, x_1) = \frac{P(Y_1 = y)}{P_y} \cdot P(x_1 | Y_1 = y) \]

For \( t = 2 \),

\[ P(Y_2 = y_1, x_{1:2}) = \sum_{y'} \frac{P(Y_2 = y_2, Y_1 = y_1', x_{1:2})}{P(Y_1 = y_1', x_1)} \cdot \frac{P(Y_1 = y_1', x_1)}{P_y} \cdot P(x_2 | Y_1 = y_1', Y_2 = y_2) \]

In general,

\[ P(Y_t = y_t | x_{1:t}) = \sum_{y_{t-1}} a_{t-1}(y_{t-1}) \cdot \Theta_{y_{t-1}, y_t} \cdot P(x_t | Y_{1:t-1}, Y_t = y_t) \]
Computing $b_t$

$$b_t(y) = P(x_{t+1:T} \mid Y_t = y)$$

$$b_0(y) := 1$$

$$b_{T-1}(y) = P(x_T \mid Y_{T-1} = y) = \sum_{y'} P(x_T, Y_T = y' \mid Y_{T-1} = y)$$

$$= \sum_{y'} P(Y_T = y' \mid Y_{T-1} = y) \cdot P(x_T \mid Y_T = y', Y_{T-1} = y)$$

$$b_{T-2}(y) = P(x_{T-1:T} \mid Y_{T-2} = y) = \sum_{y'} P(x_{T-1:T}, Y_{T-1} = y' \mid Y_{T-2} = y)$$

$$= \sum_{y'} P(Y_{T-1} = y' \mid Y_{T-2} = y) \cdot P(x_{T-1} \mid Y_{T-1} = y', Y_{T-2} = y) \cdot P(x_T \mid Y_{T-1} = y', x_{T-1})$$

$$b_{T-3}, b_T = \ldots, b_0$$
**Forward-backward algorithm:**

**Forward pass:** Initialize

\[ a_1(y) = p_y \eta_{x_1} | y \]

For \( t=2:T \) compute

\[ a_t(y) = \sum_{y'=1}^{c} a_{t-1}(y') \theta_{y'y} \eta_{x_t} | y' \]

**Backward pass:** Initialize

\[ b_T(y) = 1 \]

For \( t=T-1:-1:1 \) compute

\[ b_t(y) = \sum_{y'=1}^{c} b_{t+1}(y') \theta_{y'y} \eta_{x_t} | y' \]

For \( t=1:T \) compute

\[ P(Y_t = y \mid x_{1:T}) \propto a_t(y) b_t(y) \]

**Complexity:** \( \mathcal{O}(T \cdot c^2) \)
Maximum likelihood estimation

- **Given:** complete data (labels & observations)

\[
D = \left\{ \left( (x^{(1)}, y^{(1)}_1), \ldots, (x^{(1)}, y^{(1)}_{n_1}) \right), \ldots, \left( (x^{(1)}, y^{(m)}_1), \ldots, (x^{(1)}, y^{(m)}_{n_m}) \right) \right\}
\]

The MLE of the parameters is given by:

\[
\hat{p}_y = \frac{\text{Count}(Y_1 = y)}{m}
\]

\[
\hat{\theta}_{y|y'} = \frac{\text{Count}(Y_t = y, Y_{t-1} = y')}{\text{Count}(Y_{t-1} = y')}
\]

\[
\hat{\eta}_{x|y} = \frac{\text{Count}(X_t = x, Y_t = y)}{\text{Count}(Y_t = y)}
\]

- Might want to regularize (use pseudocounts, as in GBC / Markov models)