CPSC 340: Machine Learning and Data Mining

Markov Chains
Fall 2015
Admin

- Assignment 6 due Friday.
  - Error in Q1.1 fixed: should be able to get to logistic loss.
- We will have office hours as usual next week.
- Final exam details:
  - December 15: 8:30-11 (WESB 100).
  - 4 pages of cheat sheet allowed.
  - 9 questions.
  - Practice questions and list of topics posted.
Last Time: Markov Chains

• **Markov chains** are common way to define probability of sequence.

  ![Markov Chain Diagram]

  - **Time 0**: Rain
  - **Time 1**: No rain
  - **Time 2**: No rain
  - **Time 3**: Rain

• We discussed several tasks and how to solve them:

  1. **Sampling**: generate sequence following probability distribution.
     - Generate $x_0$ from $p(x_0)$, then generate $x_t$ conditioned on $x_{t-1}$ using $p(x_t \mid x_{t-1})$.

  2. **Learning**: estimate parameters given examples.
     - Count number of times we go from $x_{t-1}$ to $x_t$ in data, divided by number of times in $x_{t-1}$.

  3. **Inference**: computing probability of being in state ‘s’ at time ‘t’.
     - Matrix multiplication of $p(x_{t-1})$ and $p(x_t \mid x_{t-1})$ up to time ‘t’.

  4. **Stationary distribution** is steady-state after running for a long time.
     - Unique if probabilities positives, and obtained by normalized first “row” eigenvector.

  5. **Decoding**: compute most likely sequence of states.
     - Dynamic programming (“Viterbi decoding”).
Sequence of Most Probable ≠ Most Probable Sequence

- 2 roommates alternate cleaning duties for 4 days:
  - Roommate A cleans on days 0 and 2, roommate B cleans on days 1 and 3.
  - Roommate A prefers ‘clean’, but gets discourage if roommate B didn’t clean.
  - Roommate B doesn’t mind ‘mess’, especially if it’s already messy.

- Assume the following probabilities:
  - \( p(x_0 = 'clean') = 0.65. \)
  - \( p(x_1 = 'clean' | x_0 = 'clean') = 0.20. \)
  - \( p(x_1 = 'clean' | x_0 = 'mess') = 0.05. \)
  - \( p(x_2 = 'clean' | x_1 = 'clean') = 0.80. \)
  - \( p(x_2 = 'clean' | x_1 = 'mess') = 0.50. \)
  - \( p(x_3 = 'clean' | x_2 = 'clean') = 0.20. \)
  - \( p(x_3 = 'clean' | x_2 = 'mess') = 0.05. \)
Sequence of Most Probable ≠ Most Probable Sequence

- Inference gives us probability of each state at each time.

\[
\mu_0 = \begin{bmatrix} 0.35 \\ 0.65 \end{bmatrix}
\]

\[
\mu_1 = \mu_0 P_1 = \begin{bmatrix} 0.35 & 0.65 \end{bmatrix} \begin{bmatrix} 0.95 & 0.05 \\ 0.80 & 0.20 \end{bmatrix} = \begin{bmatrix} 0.8525 & 0.1475 \end{bmatrix}
\]

\[
\mu_2 = \mu_0 P_1 P_2 = \begin{bmatrix} 0.8525 & 0.1475 \end{bmatrix} \begin{bmatrix} 0.5 & 0.5 \\ 0.2 & 0.8 \end{bmatrix} = \begin{bmatrix} 0.4557 & 0.5443 \end{bmatrix}
\]

\[
\mu_4 = \mu_0 P_1 P_2 P_3 = \mu_2 P_3 = \begin{bmatrix} 0.8684 & 0.1316 \end{bmatrix}
\]

Sequence of Most Probable ≠ Most Probable Sequence

- Probability of sequence of most probable:

\[
p(x_3 = 'mess', x_2 = 'clean', x_1 = 'mess', x_0 = 'clean') = p(x_3 = 'mess' | x_2 = 'clean')p(x_2 = 'clean' | x_1 = 'mess')p(x_1 = 'mess' | x_0 = 'clean')p(x_0 = 'clean')
\]

Note: Variables are not independent.

\[
= (0.80) (0.50) (0.80) (0.65) = 0.2080.
\]

- Ignores probability of states co-occurring due to dependence.
  - Sequence of most probable states only happens 21% of the time.
**Sequence of Most Probable ≠ Most Probable Sequence**

- Decoding gives most probable sequence:

  $$\rho(x_3 = 'm', x_2 = 'm', x_1 = 'm', x_0 = 'c')$$

  $$= (0.95)(0.50)(0.95)(0.65)$$

  $$= 0.2933.$$  
  - Happens 29% of the time.

- Why the switch on day 2?
  - Many possible sequences of states.
  - Probability over all sequences that have ‘2:clean’ higher than ‘2:mess’.
  - But no individual sequence has higher probability than decoding.
Should we use decoding or inference?

• Suppose someone asks us to predict a set of variables \((x_1, x_2, x_3, x_4)\).

• Previously, we’ve only worried about prediction one variable \((y)\):
  – No distinction between decoding/inference.

• Should you use decoding or inference?
  – If payoff is based on number of variables right: use inference.
  – If you get paid for getting the whole sequence right: use decoding.
Generalizations of Basic Markov Chain Model

- Standard **Markov chain model** is very limited.
- A variety of interesting extensions exist:
  - **Multi-variable** Markov chains:
    - $x_t$ is vector (‘rain’, ‘hot’) instead of scalar, cost is exponential in number of ‘variables’.
  - **Higher-order** Markov chains:
    - $x_t$ depends on $x_{t-1}$ and $x_{t-2}$, cost is exponential in length of ‘history’.
  - **Hidden** Markov models: (Kalman filters)
    - We observe a measurement based on $x_t$ but don’t observe $x_t$ directly.
    - E.g., tracking a player/plane/missile based on video/GPS/radar.
  - **Conditional** Markov models:
    - Supervised learning where we have Markov dependency in labels.
  - **Belief networks**.
Belief Networks

• We have a dataset with binary features:

\[ x = \begin{bmatrix}
0 & 1 & 0 & 1 & 1 & 1 \\
1 & 0 & 1 & 1 & 0 & 0 \\
1 & 0 & 1 & 1 & 1 & 1
\end{bmatrix} \]

• We want to model \( p(x_i) \), probability of seeing a binary vector.
  – Why? Outlier detection, filling in missing values, scientific discovery, etc.

• We have seen two ways to do this:
  – Independent distribution (used by naïve Bayes):
    \[ p(x_i) = \prod_{j=1}^{d} p(x_{ij}) \]
  – Markov chains:
    \[ p(x_i) = p(x_{i1}) \prod_{j=2}^{d} p(x_{ij} | x_{i:j-1}) \]
Belief Networks

• **Weird notation alert:** we’ll ignore ‘i’:
  – ‘x’ will be what we would normally call ‘$x_i$’.
  – ‘$x_j$’ will be what we would normally call ‘$X_{ij}$’.

• General representation of $p(x)$ using product rule:

$$p(x) = p(x_1)p(x_2 \mid x_1)p(x_3 \mid x_1, x_2) \cdots \cdots p(x_d \mid x_1, x_2, \ldots, x_{d-1})$$

$$= \prod_{j=1}^{d} p(x_j \mid x_1; j-1)$$

• Problem: this has **too many parameters**: last term has $2^d$ values.]

• Solution 1: ‘parsimonious’ parameterization:

$$p(x_j) = \frac{1}{1+\exp(-x_jw_jx_1; j-1)}$$

• Solution 2: **conditional independence**.
Belief Networks

- Recall definition of conditional independence:
  \[ X_1 \perp X_3 \mid X_2 \iff p(X_3 \mid X_1, X_2) = p(X_3 \mid X_2) \]

- Naive Bayes:
  \[ X_j \perp X_{i:(j-1)} \mid Y \iff p(X_j \mid X_{i:(j-1)}, Y) = p(X_j \mid Y) \]

- Markov chain:
  \[ X_j \perp X_{i:(j-2)} \mid X_{j-1} \iff p(X_j \mid X_{i:(j-1)}) = p(X_j \mid X_{j-1}) \]

- Belief networks:
  \[ X_j \perp X_{i:(j-1) \setminus \pi(j)} \mid X_{\pi(j)} \iff p(X_j \mid X_{i:(j-1)}) = p(X_j \mid X_{\pi(j)}) \]
Belief Networks

- **Belief networks** assume joint distribution factorized as:

\[
p(x) = \prod_{j=1}^{d} p(x_j | x_{\text{parents}(j)})
\]

- Based on factorization we **define a graph**:
  - One vertex for each variable \(x_j\).
  - We have an edge if ‘\(i\)’ is a parent of ‘\(j\)’.

- **Importance of graph**:
  - Visual representation of assumptions.
  - Graph structure lets us **test other conditional independencies**.
  - Computational implications of graph structure (later in lecture).

- Also known as “Bayesian networks”, “Causal networks”, or “Directed acyclic graphical (DAG) models”.
Naive Bayes:

\[ Y \to X_1 \to X_2 \to X_3 \to \cdots \to X_d \]

Markov Chain:

\[ X_1 \to X_2 \to X_3 \to \cdots \to X_d \]

2nd Order Markov Chain:

\[ \begin{align*}
X_1 & \to X_2 \\
X_2 & \to X_3 \\
X_3 & \to X_4 \\
& \vdots \\
X_d & \to X_1
\end{align*} \]

"Less-naive" Bayes:

\[ p(X_d \mid X_{d-1}, X_{d-2}) \]
Hidden Markov Model: (Kalman Filtering)

\[ h_1 \rightarrow h_2 \rightarrow h_3 \]

hidden (where submarine is)

Sonar signal at time \( t \)

Conditional DAG:

\[ x_1 \rightarrow x_2 \rightarrow x_3 \]

Conditionality Example

Vancouver

Sprinkler

Rain

Grey skies

Wet grass

Sad

\[ x_i \]

\[ x_{i+1} \]
Conditional Independence Properties

- We can use the graph to check whether or not

\[ X_A \perp X_B \mid X_G \]

follows from conditional independence assumptions.
Conditional Independence Properties

- We can use the graph to check whether or not

\[ X_A \perp X_B \mid X_E \]

follows from conditional independence assumptions.

\[ X_A \perp X_B \mid X_E \]  \(\sim\) \(A\) and \(B\) are "\(d\)-separated" given \(E\)
Conditional Independence Properties

- We can use the graph to check whether or not

\[ X_A \perp X_B \mid X_E \]

follows from conditional independence assumptions.

\[ X_A \perp X_B \mid X_E \]

A and B are "d-separated" given E.

A and B are d-separated if for all paths \( P \) between A and B, at least one of the following holds:

1. \( P \) includes a "chain":

2. \( P \) includes a "fork":

3. \( P \) contains a "collider":

Note: no \( E \) were.

where C and all its descendants are unobserved.
D-separated if for all paths from A to B at least one of these "blocks" the path.

1. P includes a "chain":
   ![Chain Diagram]

2. P includes a "fork":
   ![Fork Diagram]

3. P contains a "collider":
   ![Collider Diagram]
D-separated if for all paths from A to B at least one of these "blocks" the path.

1. P includes a "chain": 
2. P includes a "fork": 
3. P contains a "collider": 

\[ \begin{align*} 
\text{Earthquake} & \rightarrow \text{Burglary} \\
\text{Alarm} & \leftrightarrow \text{Stuff is Missing} \\
\text{Neighbour Calls Police} & \end{align*} \]

Testing conditional independence:

- Earthquake \( \perp \) call "dependent" 
- Earthquake \( \perp \) call | Alarm "independent" 
- Alarm \( \perp \) Stuff Missing 
- Alarm \( \perp \) Stuff Missing | Burglary 
- Earthquake \( \perp \) Burglary 
- Earthquake \( \perp \) Burglary | Alarm "explaining away" 
- Earthquake \( \perp \) Burglary | Call 
- Burglary \( \perp \) Call 
- Call \( \perp \) Stuff is missing
Sampling:
- similar to Markov chains

```
Learning
- fit each \( p(x_i | x_{\pi(i)}) \) independently
```

- "inference" computing \( p(x) \)
  - computing \( p(x_i | x_{\pi(i)}) \)
  - similar to matrix multiplication
    - easy

- decoding: hard in general

- sample \( x_1 \)
- sample \( x_2 \)
- sample \( x_3 | x_1, x_2 \)
- sample \( x_4 | x_3 \)
- sample \( x_5 | x_3 \)

Hard in general
Decoding and "conditional" inference are easy for naive Bayes and Markov chains.

When is decoding and "conditional" inference hard?

Easy-case: "singly-connected" tree
- solved via dynamic programming,
- called "belief propagation".

Otherwise, decoding and conditional inference may be hard.
Deep Belief Networks
Cool Picture Motivation for Deep Learning

- First layer of $z_i$ trained on 10 by 10 image patches:

- Visualization of second and third layers trained on specific objects:

![Image of visualizations](http://www.cs.toronto.edu/~rgrosse/icml09-cdbn.pdf)
Summary

• Belief networks represent conditional independence using graphs.
• Graphical representation of models like naïve Bayes and Markov.
• D-separation tests any conditional independence from graph.
• Sampling/inference and learning are easy.
• Decoding/conditional-inference and learning with hidden hard.
  – But easy if graph structure is ‘nice’.

• Next time:
  – Review of topics we’ve covered, overview of topics we didn’t.