CPSC 440: Advanced Machine Learning

Hidden Markov Models

Mark Schmidt

University of British Columbia

Winter 2021

Last Time: Chapman-Kolmogorov Equations

- Chapman-Kolmogorov (CK) equations:
  - Recursive formula for computing $p(x_j = s)$ for all j and s in a Markov chain.

$$p(x_j) = \sum_{x_{j-1}=1}^{k} p(x_j \mid x_{j-1})p(x_{j-1}),$$

  - Allows us to compute all these marginal probabilities $p(x_j = s)$ in $O(dk^2)$.
    - For a length-d chain with k states.

- We also discussed stationary distributions of homogeneous Markov chains.

$$\pi(c) = \sum_{c'} p(x_j = c \mid x_{j-1} = c')\pi(c'),$$

which are sets of marginal probabilities π that don't change over time.

  - You can think of this as the "long-run average probability of being in each state".
  - Stationary distribution exists and is unique if all transitions are positive.

Application: Voice Photoshop

- Adobe VoCo uses decoding in a Markov chain as part of synthesizing voices:

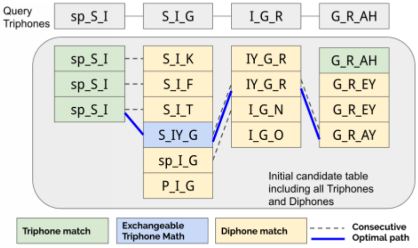


Fig. 7. Dynamic triphone preselection. For each query triphone (top) we find a candidate set of good potential matches (columns below). Good paths through this set minimize differences from the query, number and severity of breaks, and contextual mismatches between neighboring triphones.

http://gfx.cs.princeton.edu/pubs/Jin_2017_VTI/Jin2017-VoCo-paper.pdf

- https://www.youtube.com/watch?v=I3l4XLZ59iw

Decoding: Maximizing Joint Probability

- Decoding in density models: finding x with highest joint probability:

$$\underset{x_1, x_2, \ldots, x_d}{\text{argmax}} \; p(x_1, x_2, \ldots, x_d).$$

- For CS grad student ($d = 60$) the decoding is “industry” for all years.
    - The decoding often doesn't look like a typical sample.
    - The decoding can change if you increase d.

- Decoding is easy for independent models:
    - Here, $p(x_1, x_2, x_3, x_4) = p(x_1)p(x_2)p(x_3)p(x_4)$.
    - You can optimize $p(x_1, x_2, x_3, x_4)$ by optimizing each $p(x_j)$ independently.

- Can we also maximize the marginals to decode a Markov chain?

Example of Decoding vs. Maximizing Marginals

- Consider the "plane of doom" 2-variable Markov chain:

$$X = \begin{bmatrix} \text{“land”} & \text{“alive”} \\ \text{“land”} & \text{“alive”} \\ \text{“crash”} & \text{“dead”} \\ \text{“explode”} & \text{“dead”} \\ \text{“crash”} & \text{“dead”} \\ \text{“land”} & \text{“alive”} \\ \vdots & \vdots \end{bmatrix}.$$

- 40% of the time the plane lands and you live.
- 30% of the time the plane crashes and you die.
- 30% of the time the explodes and you die.

Example of Decoding vs. Maximizing Marginals

- Initial probabilities are given by

$$p(x_1 = \text{“land”}) = 0.4, \quad p(x_1 = \text{“crash”}) = 0.35, \quad p(x_1 = \text{“explode”}) = 0.25,$$

  and transition probabilities are:

$$p(x_2 = \text{“alive”} \mid x_1 = \text{“land”}) = 1, \quad p(x_2 = \text{“alive”} \mid x_1 = \text{“crash”}) = 0,$$
$$p(x_2 = \text{“alive”} \mid x_1 = \text{“explode”}) = 0.$$

- If we apply the CK equations we get

$$p(x_2 = \text{''alive”}) = 0.4, \quad p(x_2 = \text{“dead”}) = 0.6,$$

  so maximizing the marginals $p(x_j)$ independently gives (“land”, “dead”).
    - This actually has probability 0, since $p(\text{“dead''} \mid \text{“land''}) = 1$.

- Decoding considers the joint assignment to x_1 and x_2 maximizing probaiblity.
    - In this case it's (“land”, “alive”), which has probability 0.4.

Decoding with Dynamic Programming

- Note that decoding can't be done forward in time as in CK equations.
  - Even if $p(x_1 = 1) = 0.99$, the most likely sequence could have $x_1 = 2$.
  - So we need to optimize over all k^d assignments to all variables.

- Fortunately, we can solve this problem using dynamic programming.

- Ingredients of dynamic programming:
  1. Optimal sub-structure.
     - We can divide the problem into sub-problems that can individual be solved.
  2. Overlapping sub-problems.
     - The same sub-problems are reused several times.

Decoding with Dynamic Programming

- For decoding in Markov chains, we will use the following sub-problem:
  - Compute the highest probability sequence of length j ending in state s.
  - We'll use $M_j(s)$ as the probability of this sequence.

$$M_j(s) = \max_{x_1, x_2, \dots, x_{j-1}} p(x_1, x_2, \dots, x_j = s).$$

- Optimal sub-structure:
  - We can find the decoding by finding the s maximizing $M_d(s)$ (then "backtracking").
  - We can compute other $M_j(s)$ recursively (derivation of this coming up),

$$M_j(s) = \max_{x_{j-1}} \underbrace{p(x_j = s \mid x_{j-1})}_{\text{given}} \underbrace{M_{j-1}(x_{j-1})}_{\text{recurse}},$$

  with a base case of $M_1(s) = p(x_1 = s)$ (which is given by the initial probability).
- Overlapping sub-problems:
  - The same k values of $M_{j-1}(s)$ are used to compute the k values of $M_j(s)$.

Digression: Recursive Joint Maximization

- To derive the M_j formula, it will be helpful to re-write joint maximizations as

$$\max_{x_1,x_2} f(x_1,x_2) = \max_{x_1} f_1(x_1),$$

where $f_1(x_1) = \max_{x_2} f(x_1,x_2)$ (this f_1 "maximizes out" over x_2).

  - This is similar to the marginalization rule in probability.

- Plugging in the definition of $f_1(x_1)$ we obtain:

$$\max_{x_1,x_2} f(x_1,x_2) = \max_{x_1} \underbrace{\max_{x_2} f(x_1,x_2)}_{f_1(x_1)}.$$

- You can do this trick repeatedly and/or with any number of variables.

Decoding with Dynamic Programming

- Derivation of recursive calculation $M_j(x_j)$ for decoding Markov chains:

$$\begin{aligned}
M_j(x_j) &= \max_{x_1,x_2,\ldots,x_{j-1}} p(x_1,x_2,\ldots,x_j) && \text{(definition of } M_j(x_j)\text{)}\\
&= \max_{x_1,x_2,\ldots x_{j-1}} p(x_j \mid x_1,x_2,\ldots x_{j-1})p(x_1,x_2,\ldots,x_{j-1}) && \text{(product rule)}\\
&= \max_{x_1,x_2,\ldots x_{j-1}} p(x_j \mid x_{j-1})p(x_1,x_2,\ldots,x_{j-1}) && \text{(Markov property)}\\
&= \max_{x_{j-1}} \left\{ \max_{x_1,x_2,\ldots x_{j-2}} p(x_j \mid x_{j-1})p(x_1,x_2,x_{j-1}) \right\} && (\max_{a,b} f(a,b) = \max_a \{\max_b f(a,b)\})\\
&= \max_{x_{j-1}} \left\{ p(x_j \mid x_{j-1}) \max_{x_1,x_2,\ldots x_{j-2}} p(x_1,x_2,x_{j-1}) \right\} && (\max_i \alpha a_i = \alpha \max_i a_i \text{ for } \alpha \geq 0)\\
&= \max_{x_{j-1}} \underbrace{p(x_j \mid x_{j-1})}_{\text{given}} \underbrace{M_{j-1}(x_{j-1})}_{\text{recurse}} && \text{(definition of } M_{j-1}(x_{j-1})\text{)}
\end{aligned}$$

- For each (j, s) we also store the maximizing value of x_{j-1}.
  - Once we have $M_j(x_j = s)$ for all j and s values, backtrack using these values to solve problem.

Example: Decoding the Plane of Doom

- We have $M_1(x_1) = p(x_1)$ so in "plane of doom" we have

$$M_1(\text{"land"}) = 0.4, \quad M_1(\text{"crash"}) = 0.3, \quad M_1(\text{"explode"}) = 0.3.$$

- We have $M_2(x_2) = \max_{x_1} p(x_2 \mid x_1) M_1(x_1)$ so we get

$$M_2(\text{"alive"}) = 0.4, \quad M_2(\text{"dead"}) = 0.3.$$

- $M_2(2) \neq p(x_2 = 2)$ because we needed to choose either "crash" or "explode".
  - And notice that $\sum_{c=1}^{k} M_2(x_j = c) \neq 1$ (this is not a distribution over x_2).

- We maximize $M_2(x_2)$ to find that the optimal decoding ends with "alive".
  - We now need to backtrack to find the state that lead to "alive", giving "land".

Viterbi Decoding

- The Viterbi decoding dynamic programming algorithm:
  1. Set $M_1(x_1) = p(x_1)$ for all x_1.
  2. Compute $M_2(x_2)$ for all x_2, store value of x_1 leading to the best value of each x_2.
  3. Compute $M_3(x_3)$ for all x_3, store value of x_2 leading to the best value of each x_3.
  4. ...
  5. Maximize $M_d(x_d)$ to find value of x_d in a decoding.
  6. Bactrack to find the value of x_{d-1} that lead to this x_d.
  7. Backtrack to find the value of x_{d-2} that lead to this x_{d-1}.
  8. ...
  9. Backtrack to find the value of x_1 that lead to this x_2.

- For a fixed j, computing all $M_j(x_j)$ given all $M_{j-1}(x_{j-1})$ costs $O(k^2)$.
  - Total cost is only $O(dk^2)$ to search over all k^d paths.
  - Has numerous applications like decoding digital TV.

Decoding with Dynamic Programming

- What Viterbi decoding data structures might look like ($d = 4, k = 3$):

$$M = \begin{bmatrix} 0.25 & 0.25 & 0.50 \\ 0.35 & 0.15 & 0.05 \\ 0.10 & 0.05 & 0.05 \\ 0.02 & 0.03 & 0.05 \end{bmatrix}, \quad B = \begin{bmatrix} \emptyset & \emptyset & \emptyset \\ 1 & 1 & 3 \\ 2 & 1 & 1 \\ 2 & 2 & 1 \end{bmatrix}.$$

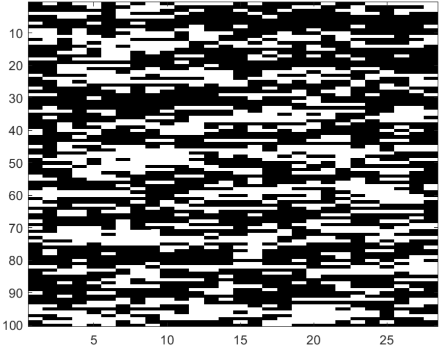
- The $d \times k$ matrix M stores the values $M_j(s)$, while B stores the argmax values.

- From the last row of M and the bactracking matrix B, the decoding is $x_1 = 1, x_2 = 2, x_3 = 1, x_4 = 3$.

Outline

Back to the Rain Data

- We previously considered the "Vancouver Rain" data:



- We used homogeneous Markov chains to model between-day dependence.

Back to the Rain Data

- But doesn't it rain less in the summer?
- There are hidden clusters in the data not captured by the Markov chain.
  - But mixture of independent models are inefficient at representing direct dependency.
- Mixture of Markov chains could capture direct dependence *and* clusters,

$$p(x_1, x_2, \ldots, x_d) = \sum_{c=1}^{k} p(z = c) \underbrace{p(x_1 \mid z = c)p(x_2 \mid x_1, z = c) \cdots p(x_d \mid x_{d-1}, z = c)}_{\text{Markov chain } c}.$$

- Cluster z chooses which homogeneous Markov chain parameters to use.
  - We could learn that we're more likely to have rain in winter.
  - Can modify CK equations to take into account z, and then apply EM.

Comparison of Models on Rain Data

- Independent (homogeneous) Bernoulli:
  - Average NLL: 18.97 (1 parameter).
- Independent Bernoullis:
  - Average NLL: 18.95, (28 parmaeters).
- Mixture of Bernoullis ($k = 10$, five random restarts of EM):
  - Average NLL: 17.06 ($10 + 10 \times 28 = 290$ parameters)
- Homogeneous Markov chain:
  - Average NLL: 16.81 (3 parameters)
- Mixture of Markov chains ($k = 10$, five random restarts of EM):
  - Average NLL: 16.53 ($10 + 10 \times 3 = 40$ parameters).
  - Included what I call a "summer" cluster:

$$p(z = 5) = 0.14$$

$$p(x_1 = \text{"rain"} \mid z = 5) = 0.22 \qquad \text{(instead of usual 37\%)}$$

$$p(x_j = \text{"rain"} \mid x_{j-1} = \text{"rain"}, z = 5) = 0.49 \qquad \text{(instead of usual 65\%)}$$

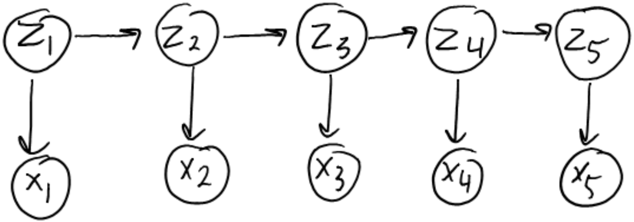
$$p(x_j = \text{"rain"} \mid x_{j-1} = \text{"not rain"}, z = 5) = 0.11 \qquad \text{(instead of usual 35\%)}$$

Back to the Rain Data

- The rain data is artificially divideded into months.
- We previously discussed viewing rain data as one very long sequence ($n = 1$).
- We could apply homogeneous Markov chains due to parameter tieing.
- But a mixture doesn't make sense when $n = 1$.
- What we want: different "parts" of the sequence come from different clusters.
  - We transition from "summer" cluster to "fall" cluster at some time j.
- One way to address this is with a "hidden" Markov model (HMM):
  - Instead of months being assigned to clusters, days are assigned to clusters.
  - Have a Markov dependency between cluster values of adjacent days.

Hidden Markov Models (MEMORIZE)

- Hidden Markov models have each x_j depend on hidden Markov chain.

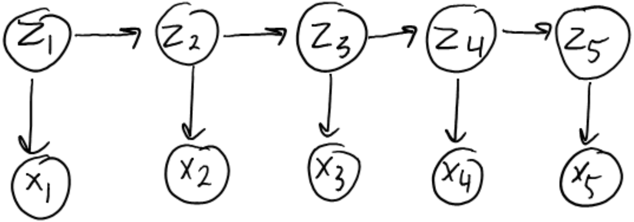


$$p(x_1, x_2, \dots, x_d, z_1, z_2, \dots z_d) = p(z_1) \prod_{j=2}^{d} p(z_j \mid z_{j-1}) \prod_{j=1}^{d} p(x_j \mid z_j).$$

- We're going to learn clusters z_j and the hidden dynamics.
  - Hidden cluster z_j could be "summer" or "winter" (we're learning the clusters).
  - Transition probability $p(z_j \mid z_{j-1})$ is probability of staying in "summer".
    - Initial probability $p(z_1)$ is probability of starting chain in "summer".
  - Emission probability $p(x_j \mid z_j)$ is probability of "rain" during "summer".

Hidden Markov Models

- Hidden Markov models have each x_j depend on hidden Markov chain.



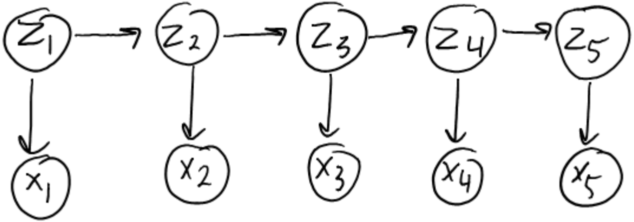
$$p(x_1, x_2, \dots, z_1, z_2, \dots z_d) = p(z_1) \prod_{j=2}^{d} p(z_j \mid z_{j-1}) \prod_{j=1}^{d} p(x_j \mid z_j).$$

- You observe the x_j values but do not see the z_j values.
  - There is a "hidden" Markov chain, whose state determines the cluster at each time.

- Note that the x_j can be continuous even with discrete clusters z_j.
  - Data could come from a mixture of Gaussians, with cluster changing in time.

Hidden Markov Models

- Hidden Markov models have each x_j depend on hidden Markov chain.



$$p(x_1, x_2, \ldots, z_1, z_2, \ldots z_d) = p(z_1) \prod_{j=2}^{d} p(z_j \mid z_{j-1}) \prod_{j=1}^{d} p(x_j \mid z_j).$$

- If the z_j are continuous it's often called a state-space model.
  - If everything is Gaussian, it leads to Kalman filtering.
  - Keywords for non-Gaussian: unscented Kalman filter and particle filter.

- Variants of HMMs are probably the most-used time-series model...

Applications of HMMs and Kalman Filters

Applications [edit]

HMMs can be applied in many fields where the goal is to recover a data sequence that is not immediately observable (but other data that depend on the sequence are). Applications include:

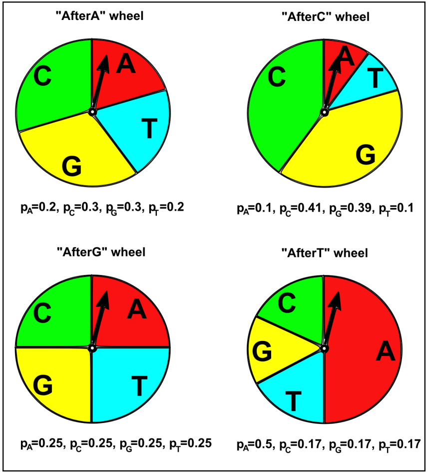
- Single Molecule Kinetic analysis[16]
- Cryptanalysis
- Speech recognition
- Speech synthesis
- Part-of-speech tagging
- Document Separation in scanning solutions
- Machine translation
- Partial discharge
- Gene prediction
- Alignment of bio-sequences
- Time Series Analysis
- Activity recognition
- Protein folding[17]
- Metamorphic Virus Detection[18]
- DNA Motif Discovery[19]

Applications [edit]

- Attitude and Heading Reference Systems
- Autopilot
- Battery state of charge (SoC) estimation[39][40]
- Brain-computer interface
- Chaotic signals
- Tracking and Vertex Fitting of charged particles in Particle Detectors[41]
- Tracking of objects in computer vision
- Dynamic positioning
- Economics, in particular macroeconomics, time series analysis, and econometrics[42]
- Inertial guidance system
- Orbit Determination
- Power system state estimation
- Radar tracker
- Satellite navigation systems
- Seismology[43]
- Sensorless control of AC motor variable-frequency drives
- Simultaneous localization and mapping
- Speech enhancement
- Visual odometry
- Weather forecasting
- Navigation system
- 3D modeling
- Structural health monitoring
- Human sensorimotor processing[44]

Example: Modeling DNA Sequences

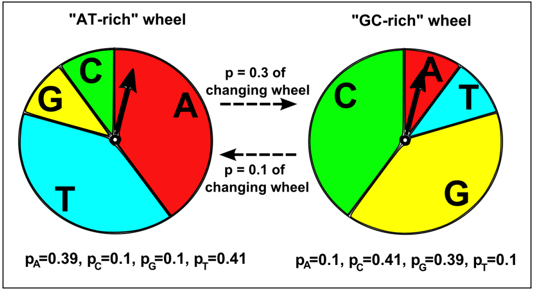
- Markov model for elements of sequence (dependence on previous symbol):



https://www.tes.com/lessons/WE5E9RncBhieAQ/dna

Example: Modeling DNA Sequences

- Hidden Markov model (HMM) for elements of sequence (two hidden clusters):



- This is a (hidden) state transition diagram.
    - Can reflect that probabilities are different in different regions.
    - The actual regions are not given, but instead are nuissance variables handled by EM.
- A better model might use a hidden and visible Markov chain.
    - With 2 hidden clusters, you would have 8 "probability wheels" (4 per cluster).
    - Would have "treewidth 2", which we'll show later means it's tractable to use.

Summary

- Decoding is task of finding most probable x.
- Viterbi decoding allow efficient decoding with Markov chains.
  - A special case of dynamic programming.
- Hidden Markov models model time-series with hidden per-time cluster.
  - Tons of applications, typically more realistic than Markov models.
- Next time: measuring defence in the NBA.