CPSC 540: Machine Learning

Message Passing

Mark Schmidt

University of British Columbia

Winter 2020
If we want to approximate expectations of random functions,

$$E[g(x)] = \sum_{x \in X} g(x)p(x) \quad \text{or} \quad E[g(x)] = \int_{x \in \mathcal{X}} g(x)p(x)dx,$$

the Monte Carlo estimate is

$$E[g(x)] \approx \frac{1}{n} \sum_{i=1}^{n} g(x^i),$$

where the $x^i$ are independent samples from $p(x)$.

We can use this to approximate marginals,

$$p(x_j = c) \approx \frac{1}{n} \sum_{i=1}^{n} I[x_j^i = c].$$
Exact Marginal Calculation

- In typical settings Monte Carlo has slow convergence like stochastic gradient.
  - $O(1/t)$ convergence rate where constant is variance of samples.
    - If all samples look the same, it converges quickly.
    - If samples look very different, it can be painfully slow.

- For discrete-state Markov chains, we can actually compute marginals directly:
  - We’re given initial probabilities $p(x_1 = s)$ for all $s$ as part of the definition.
  - We can use transition probabilities to compute $p(x_2 = s)$ for all $s$:

$$p(x_2) = \sum_{x_1=1}^k p(x_2, x_1) = \sum_{x_1=1}^k p(x_2 | x_1)p(x_1).$$

- We can repeat this calculation to obtain $p(x_3 = s)$ and subsequent marginals.
Exact Marginal Calculation

- **Recursive formula for marginals at time $j$:**

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

  called the **Chapman-Kolmogorov (CK) equations**.

- The CK equations can be implemented as **matrix-vector multiplication:**
  - Define $\pi^j$ as a vector containing the marginals at time $t$:
    \[ \pi^j_c = p(x_j = c). \]
  - Define $T^j$ as a matrix containing the transition probabilities:
    \[ T^j_{cc'} = p(x_j = c \mid x_{j-1} = c'). \]
Exact Marginal Calculation

- Implementing the CK equations as a matrix multiplications:

\[ T^j \pi_j^{-1} = \begin{bmatrix}
  p(x_j = 1 | x_{j-1} = 1) & p(x_j = 1 | x_{j-1} = 2) & \cdots & p(x_j = 1 | x_{j-1} = k) \\
  p(x_j = 2 | x_{j-1} = 1) & p(x_j = 2 | x_{j-1} = 2) & \cdots & p(x_j = 2 | x_{j-1} = k) \\
  \vdots & \vdots & \ddots & \vdots \\
  p(x_j = k | x_{j-1} = 1) & p(x_j = k | x_{j-1} = 2) & \cdots & p(x_j = k | x_{j-1} = k)
\end{bmatrix}
\begin{bmatrix}
  p(x_{j-1} = 1) \\
  p(x_{j-1} = 2) \\
  \vdots \\
  p(x_{j-1} = k)
\end{bmatrix}
\]

\[ = \begin{bmatrix}
  \sum_{c=1}^k p(x_j = 1 | x_{j-1} = c) p(x_{j-1} = c) \\
  \sum_{c=1}^k p(x_j = 2 | x_{j-1} = c) p(x_{j-1} = c) \\
  \vdots \\
  \sum_{c=1}^k p(x_j = k | x_{j-1} = c) p(x_{j-1} = c)
\end{bmatrix}
\begin{bmatrix}
  p(x_j = 1) \\
  p(x_j = 2) \\
  \vdots \\
  p(x_j = k)
\end{bmatrix} = \pi_j.
\]

- Cost of multiplying a vector by a $k \times k$ matrix is $O(k^2)$.

- So cost to compute marginals up to time $d$ is $O(dk^2)$.
  - This is fast considering that last step sums over all $k^d$ possible paths.

\[ p(x_d) = \sum_{x_1=1}^k \sum_{x_2=1}^k \cdots \sum_{x_{j-1}=1}^k \sum_{x_{j+1}=1}^k \cdots \sum_{x_{d-1}=1}^k p(x_1, x_2, \ldots, x_d). \]
Marginals in CS Grad Career

- CK equations can give all marginals $p(x_j = c)$ from CS grad Markov chain:

- Each row $j$ is a state and each column $c$ is a year.
Continuous-State Markov Chains

- The CK equations also apply if we have continuous states:

\[ p(x_j) = \int_{x_{j-1}} p(x_j \mid x_{j-1}) p(x_{j-1}) \, dx_{j-1}, \]

but this integral may not have a closed-form solution.

- **Gaussian probabilities** are an important special case:
  - If \( p(x_{j-1}) \) and \( p(x_j \mid x_{j-1}) \) are Gaussian, then \( p(x_j) \) is Gaussian.
    - Joint distribution is a product of Gaussians.
    - So we can write \( p(x_j) \) in closed-form in terms of mean and variance.

- If the probabilities are non-Gaussian, usually cannot represent \( p(x_j) \) distribution.
  - You are stuck using Monte Carlo or other approximations.
A stationary distribution of a homogeneous Markov chain is a vector $\pi$ satisfying

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

“Probabilities don’t change across time” (also called “invariant” distribution).

Here are talking about the “marginal” probabilities $p(x_j)$, not the “transition” probabilities $p(x_j \mid x_{j-1})$.

Under certain conditions, marginals converge to a stationary distribution.

- $p(x_j = c) \to \pi(c)$ as $j$ goes to $\infty$.
- If we fit a Markov chain to the rain example, we have $\pi(\text{“rain”}) = 0.41$.
- In the CS grad student example, we have $\pi(\text{“dead”}) = 1$.

Stationary distribution is basis for Google’s PageRank algorithm.
Application: PageRank

- Web search before Google:

- It was also easy to fool search engines by copying popular websites.

State Transition Diagram

- **State transition diagrams** are common for visualizing homogenous Markov chains:

  \[
  P = \begin{bmatrix}
  0 & 0 & 0.2 & 0.8 \\
  0 & 0 & 0 & 1 \\
  0.2 & 0 & 0 & 0.8 \\
  0 & 0.45 & 0.45 & 0.1
  \end{bmatrix}
  \]

  - Each node is a state, each edge is a non-zero transition probability.
  - For web-search, each node will be a webpage.
  - **Cost of CK equations is only** $O(z)$ if you have only $z$ edges.
Application: PageRank

- Wikipedia’s cartoon illustration of Google’s PageRank:
  - Large face means higher rank.

- “Important webpages are linked from other important webpages”.
- “Link is more meaningful if a webpage has few links”.

[Link to Wikipedia page on PageRank](https://en.wikipedia.org/wiki/PageRank)
Google’s PageRank algorithm for measuring the importance of a website:
- Stationary probability in “random surfer” Markov chain:
  - With probability $\alpha$, surfer clicks on a random link on the current webpage.
  - Otherwise, surfer goes to a completely random webpage.

To compute the stationary distribution, they use the power method:
- Repeatedly apply the CK equations.
- Iterations are faster than $O(k^2)$ due to sparsity of links.
  - Transition matrix is “sparse plus rank-1” which allows fast multiplication.
- Can be easily parallelized.
Application: Game of Thrones

- PageRank can be used in other applications.
- “Who is the main character in the Game of Thrones books?”

Existence/Uniqueness of Stationary Distribution

Does a stationary distribution $\pi$ exist and is it unique?

A sufficient condition for existence/uniqueness is that all $p(x_j = c \mid x_{j'} = c') > 0$.
- PageRank satisfies this by adding probability $\alpha$ of jumping to a random page.

Weaker sufficient conditions for existence and uniqueness ("ergodic"):  
1. "Irreducible" (doesn’t get stuck in part of the graph).  
2. "Aperiodic" (probability of returning to state isn’t on fixed intervals).
Outline

1. Exact Marginals and PageRank
2. Message Passing
Decoding: Maximizing Joint Probability

- **Decoding** in density models: finding \( x \) with highest joint probability:
  \[
  \arg\max_{x_1, x_2, \ldots, x_d} p(x_1, x_2, \ldots, x_d).
  \]

- 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 plane explodes and you die.
Example of Decoding vs. Maximizing Marginals

- Initial probabilities are given by
  \[ p(x_1 = \text{"land"}) = 0.4, \quad p(x_1 = \text{"crash"}) = 0.3, \quad p(x_1 = \text{"explode"}) = 0.3, \]
  and \( x_2 \) is "alive" iff \( x_1 \) is "land".

- 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.

- Decoding considers the joint assignment to \( x_1 \) and \( x_2 \) maximizing probability.
  - In this case it's ("land", "alive"), which has probability 0.4.
To decode Markov chains, 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:

\[
\max_{x_1, x_2} f(x_1, x_2) = \max_{x_1} \max_{x_2} f(x_1, x_2) = \max_{x_1} f_1(x_1).
\]
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.

- Key quantity is solving the sub-problem $M_j(x_j)$.
  - Find the "highest probability sequence of length $j$ ending in $x_j"$,

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

- Base case: $M_1(x_1) = p(x_1)$ (which is given by the initial probability).
- We can compute other $M_j(x_j)$ recursively (next slide).
Decoding with Dynamic Programming

Recursive calculation of “highest probability sequence of length \( j \) ending in \( x_j \):

\[
M_j(x_j) = \max_{x_1, x_2, \ldots, x_{j-1}} p(x_1, x_2, \ldots, x_j)
\]

(definition of \( M_j(x_j) \))

\[
= \max_{x_1, x_2, \ldots, x_{j-1}} p(x_j | x_1, x_2, \ldots, x_{j-1}) p(x_1, x_2, \ldots, x_{j-1})
\]

(product rule)

\[
= \max_{x_1, x_2, \ldots, x_{j-1}} p(x_j) p(x_{j-1}) p(x_1, x_2, \ldots, x_{j-1})
\]

(Markov property)

\[
= \max_{x_{j-1}} \left\{ \max_{x_1, x_2, \ldots, x_{j-2}} p(x_j | x_{j-1}) p(x_1, x_2, x_{j-1}) \right\}
\]

\[
= \max_{x_{j-1}} \left\{ \max_{x_1, x_2, \ldots, x_{j-2}} p(x_j | x_{j-1}) \max_{x_1, x_2, \ldots, x_{j-2}} p(x_1, x_2, x_{j-1}) \right\}
\]

\[
= \max_{x_{j-1}} p(x_j | x_{j-1}) M_{j-1}(x_{j-1})
\]

(given \( M_{j-1}(x_{j-1}) \))

Once we have computed \( M_j(x_j = c) \) for all \( j \) and \( c \) values, we can backtrack to solve the problem (later).
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”.
The Viterbi decoding algorithm (special case of dynamic programming):

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. Backtrack 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 $x_2$.

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.
Application: Voice Photoshop

Application: Adobe VoCo uses Viterbi as part of synthesizing voices:


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

- Chapman-Kolmogorov equations compute exact univariate marginals.
  - For discrete or Gaussian Markov chains.

- Stationary distribution of homogenous Markov chain.
  - Marginals as time goes to $\infty$.
  - Basis of Google’s PageRank method.

- Decoding is task of finding most probable $x$.

- Viterbi decoding allow efficient decoding with Markov chains.

- Next time: measuring defence in the NBA.
Label Propagation as a Markov Chain Problem

- Basic label propagation method has a Markov chain interpretation.
  - We have \( n + t \) states, one for each [un]labeled example.

- Monte Carlo approach to label propagation ("adsorption"):
  - At time \( t = 0 \), set the state to the node you want to label.
  - At time \( t > 0 \) and on a labeled node, output the label.
    - Labeled nodes are absorbing states.
  - At time \( t > 0 \) and on an unlabeled node \( i \):
    - Move to neighbour \( j \) with probability proportional to \( w_{ij} \) (or \( \bar{w}_{ij} \)).

- Final predictions are probabilities of outputting each label.
  - Nice if you only need to label one example at a time (slow if labels are rare).
  - Common hack is to limit random walk time to bound runtime.