Query Planning in HMMs and Graphical Models

CPS 570

Ron Parr

Anderson & Moore Paper

Motivation: Activity Recognition

- Intruder detection on a computer system.
  - You could have models of normal user activities and suspicious user activities. (*hacker*)
- Intruder detection in a building with sensors/cameras.
  - You could have models of normal and suspicious activities. (*bank robber)*
- Your smartwatch/phone could use accelerometers & GPS to guess if you are running/doing elliptical etc.
- You could have models of what users are doing while carrying mobile phone based upon location and accelerometer inputs, and use these to determine what notifications to present.

Suppose State is Observable at a Cost

- Computer intruder detection: Stop user activity: password challenge, or require MFA
- Physical intruder detection: Send policy to interrogate suspicious person
- Activity recognition on a mobile device: Pop up an alert and ask user what he/she is doing

Other Examples of Costly Queries

- Send a diver/sub to examine an underwater phenomenon
- Send a scientist into the forest to make measurements
- Send a fighter plane to check out a radar blip
- Run a diagnostic test on a patient
- Many others…

What Is A Query

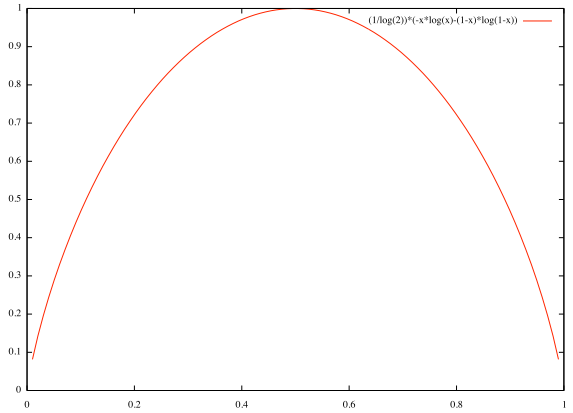
- A&M paper uses general notion of a query
  - View ordinary observation
  - Ask a noisy oracle
  - Ask a true oracle
- More general than K&G notion of query discussed later

Entropy

$$H(P) = -\sum_{i=1}^{n} p_i \log_2(p_i)$$

For an event space with n events p_i = prob of event i

Entropy of a Coin



Is Entropy Bad?

- Reducing entropy is often thought of a good thing, but…
- Maximizing entropy can sometimes imply making the weakest assumptions – maximum entropy solution is considered the "most conservative" one
- Entropy can increase by changing your event space
  - Suppose you have a distribution over threat types
  - Landmines, coca-cola cans, logs, etc.
  - Increasing the number of types of mines in your database increases entropy, but doesn't necessarily lead to worse outcomes

Symmetry of Mutual Information

$$
\begin{aligned}
&H(Q) - H(Q|X) \\
&\quad = -\sum_q p(q)\log p(q) - (-\sum_x p(x)\sum_q p(q|x)\log p(q|x)) \\
&\quad = -\sum_q p(q)\log p(q) - (-\sum_x p(x)\sum_q \frac{p(q,x)}{p(x)}\log\frac{p(q,x)}{p(x)}) \\
&\quad = -\sum_q p(q)\log p(q) - (-\sum_x \sum_q (p(q,x)\log p(q,x) - p(q,x)\log p(x))) \\
&\quad = -\sum_q p(q)\log p(q) + \sum_x \sum_q p(q,x)\log p(q,x) - \sum_x \sum_q p(q,x)\log p(x) \\
&\quad = -\sum_q p(q)\log p(q) + \sum_x \sum_q p(q,x)\log p(q,x) - \sum_x p(x)\log p(x)
\end{aligned}
$$

Other direction reduces to this too, thus H(Q)-H(Q|X) = H(X) – H(X|Q)

Time Complexity

- N states, M observations, T time steps
- Sum of state entropy reductions: $O(T^2N^2M)$
  - $O(TN^2)$ to compute entropy reduction from a particular query outcome (forward-backward)
  - O(TM) queries and outcomes
- Expected confusion cost: $O(TN^2M)$
  - Assigns cost to confusing one state with another
  - Same basic idea as entropy calculation
  - Factor of T saved by some pre-computation

Path Space Size

- Suppose you want the Viterbi path?
- How many possible paths are there?
- $O(N^T)$ possible paths: N choices at each time
- Even computing the entropy over paths, $H(\Pi)$, seems hopeless, let alone computing the entropy-minimizing query! ☹☹☹

Using Symmetry of Mutual Information

- Want: $\text{argmin}_Q H(\Pi)-H(\Pi|Q)$
- $H(\Pi)-H(\Pi|Q) = H(Q)-H(Q|\Pi)$
- $H(Q)$ is easy
- $H(Q|\Pi) = H(Q|S)$ (Markov property)
- $H(Q|S)$ is also easy
- Cost O(TMN) to find query that gives highest expected reduction in path entropy (assuming you have already run forward-backward)
  (N.B.: This is the surprising result!)

Method Concerns

- Is entropy reasonable?
- Is the cost model reasonable?
- Is myopic query selection reasonable?

Krause and Guestrin Paper

Different Assumptions

- Assumes all HMM observations made at all time steps
- Assumes forward-backward algorithm has already been run to completion
- Result = chain of pairwise-correlated variables
- Queries reveal the exact state at a given time

"Rewards"

- Assume improvement from knowing the value of a variable decomposes into a sum of "rewards" for knowing individual variables
- Each sub-reward is a function over the distribution on the state variable at a particular time
- Rewards could be negative entropy, etc.

Optimal Query Subset Selection

- Set of queries are the best one to choose *a priori*?
- Once you pick a set, can't change your mind!
- Main idea:
  - Compute $L_{a:b}(k)$ = biggest expected improvement achievable between t=a and t=b, assuming you have k choices left, and a,b are observed
  - Compute $L_{a:b}(k)$ from $L_{a:b}(k-1)$
  - $O(T^2)$ choices of a,b
  - O(T) to maximize over all choices between a and b
  - Total: $O(T^3)$ BC, where B is the total query budget, C is cost of evaluating improvement at a particular state-time – typically O(N)
  - Also: $O(T^3N^2)$ setup time to compute $L_{a:b}(0)$ + cost of forward-backward

Size of a Conditional Plan

- What is a conditional Plan?
  - Conditional plan tells us what to do based upon different histories
  - Conditional plans are non-myopic because they plan ahead (possible futures when planning = possible histories when acting)
- Need to explore all the possible paths to get a plan in general - size will be N^T
- Authors provide much more efficient algorithm!

Non-Myopic Observation Planning

- Main idea:
  - Compute $J_{a:b}(x_a,x_b;k)$: The biggest expected improvement picking variables between t=a and t=b, with a,b observed as x_a x_b.
  - Compute $J_{a:b}(x_a,x_b;k)$ from $J_{a:b}(x_a,x_b;k-1)$
  - Adds a factor M^3 to previous algorithm (need to iterate over all combinations of x_a,x_b,x_c, a<b<c
- This is a shocking result
- Optimal conditional plan has polynomial computation time and polynomial size
- Why: Markov property

Now The Bad News

- This only works for HMMs (graphical models with chain structures)
- Finding the optimal plan or even computing the expected improvement for trees is worse than NP-hard

Hardness Of Reward Computation (Reduction from 3SAT)

- $U_0 \ldots U_n$ correspond to variables
- Y_0 picks a clause (from m clauses)
- $Y_1 \ldots Y_n$ check if a variables satisfies the clause or if clause is already satisfied (=0) otherwise passes through value of Y_0
- Reward function has: $R(P(Y_n|X_1 \ldots X_n)=0) = 2^n$, 0 OTW
- The expected value of the reward must count the number of satisfying assignments

Conclusion

- Query selection is an important resource management problem
- Myopic HMM query selection can be done somewhat efficiently
- Non-myopic query selection can be done surprisingly efficiently for chain graphical models, e.g., HMMs
- Non-myopic query selection is intractable for anything other than chains