Notes on Probabilistic Inference for Computer Graphics Researchers

Niels Joubert
Stanford University
niels@cs.stanford.edu

March 2013

Contents

1 Problem Encoding

We take a **Bayesian** view of the universe, where we encode uncertainty as probabilities. For our purposes we make no comment on whether these uncertainties are intrinsic to the world or a consequence of incomplete information - either or both can apply.

> "Probability provides a way to summarize the uncertainty that comes from laziness and ignorance." [3]

We also take a **problem-centric** view of probability, rather than the world-based view usually found in Artificial Intelligence textbooks. They're quite similar: we define the **state space** as the space of all possible solutions to some posed problem, where we are exact about what is given as inputs, and what is variable.

1.1 Examples of State Spaces

Coloring of Pattern, as in Lin et al [1]

- State - An assignment of a color to every region in a pattern.
- State Space - All possible assignments of colors to the fixed regions of a pattern image.
- Number of states - Assuming a fixed color palette: $O(n^c)$ where n is the number of regions and c is the number of colors in the palette.

Interior Design Furniture Layout, as in Merrel et al [2]

- State - An arrangement (position, rotation) of furniture pieces in a given room.
- State Space - All possible arrangements of a given set of furniture pieces in a given room.
- Number of states - Assuming continuous variability in each of the parameters of a furniture piece, infinitely many arrangements exist.

Coffee Shop Interior, as in Yeh at al [4]

- State - A spatial arrangement of cutlery, plates and flowers on tables, and an arrangement of said tables with chairs in a given interior space.
- State Space - All possible arrangements of a variable number of elements in a given interior.
- Number of states - Not only is there an infinite arrangement for a given set of items, there is also a variable amount of items, thus an infinite state space exist.

1.2 Probability Model

From these examples we see that infinite-size state spaces are fairly common, thus exhaustive search methods are, as always, almost immediately ruled out for anything but the simplest of problems.

Defining the space as a **probability model** associates a numerical probability $P(w)$ with each possible world w. Notice that we make no comment yet on how to calculate this value.

2 Defining a state space: probabilistic graphical model

We can write down our problem as a probability model, where we come up with a scoring function on states. This scoring function (an energy function) can be converted to a probability model through something like the Boltzmann energy function.

3 Connecting states in the state space: Markov Chain Monte Carlo

We can define a process that moves between states in the state space. That is, consider some state. We say that we can generate a new state from the current state by changing the current state. Rather than naively say we can generate any possible state directly, we consider a set of small local changes applied to the current state. Thus, we can connect the states in the state space by the local changes applied to the one to generate the other.

Let's formalize this idea as a **Markov Chain** by introducing probabilities to the changes we make to a state. These changes from one state to another is a **transition** from state w_i to w_j. We already have a **probability model** that assigns a numerical probability $P(w_i)$ to every state.

Rather than consider these transitions as arbitrary changes, we assign a **transition probability** to each transition. This defines the three parts of a markov chain:

1. The states of the Markov Chain - each state in the state space
2. The transitions of the Markov Chain - changes that can be applied to each state, and the resulting state it produces
3. The transition probabilities - as yet undefined measure of the likelihood of a transition.

4 Sampling the state space: Metropolis-Hastings sampling of the Markov Chain

We know we cannot explore the entire state space (in many cases it is infinite!), and we'll leave it to the referenced literature to convince you that these spaces can be highly multimodal, so greedy approaches such as hillclimbing or forward sampling (mostly not even applicable due to the intractability of the undirected network) struggles with local maxima and large areas of low-scoring results.[1] We adopt a slightly different viewpoint for these types of design problems:

> *We consider the local maxima to be interesting examples of possible designs that is worth exploring (at least relative to how well they satisfy our metrics), rather than places to avoid.*

This viewpoint leads us to use the Markov Chain as an attempt to efficiently *explore* the space. To do this, we make an important connection: We set up the transition probabilities to be relative to the increase in energy we experience going from an initial to a new state.

To apply a Monte Carlo sampling technique to this Markov Chain, we iteratively move from state to state by applying a transition function, where we pick the transition function according to its transition probability. This means that, in the limit and with the correct type of transition probabilities, we explore this space relative to the probabilities of each state in the space. For example, running the Monte Carlo sampler to generate 1000 samples, we expect to see 500

[1] Yes, these are GROSS sweeping statements, and many problems are highly amenable to greedy search approaches, convex optimization approaches and even exhaustive search. Forgive me that for the purposes of this document we will gloss over those and show how to apply stochastic search methods.

samples of a state that has a probability of 0.5 (which translates to having half of the entire space's energy concentrated on this state).

Of course this is only true in the limit, and especially during the initial **burn-in** time, the sampler does not produce samples according to this probability.

How do we set up our transition probabilities to ensure this nice limit property?

One approach is to ensure that the Markov Chain is **ergodic** (also called "regular"). For a Markov Chain to be ergodic, it needs to satisfy two properties:

1. For every state w_i, there is some t such that applying the transitions functions $f_0 ... f_t$ to w_i produces any other state w_j in the space. (That is, every state is reachable from every other state)
2. There are no strictly periodic cycles in the Markov Chain.

When ergodicity holds, we can prove that the **stationary distribution** of the Markov Chain (the probability of being in a state after t steps with t being sufficiently large) is equal to the probability distribution of the underlying space. See Chapter 14, Section "Inference by Markov Chain Simulation" in Russel and Norvig.[3]

This leads to the Gibbs sampling and Metropolis-Hastings algorithms, both which iteratively samples the markov chain to produce the stationary distribution.

5 Incorporating constraints into the state space as factor graphs

Still needs to be filled out.

6 Appendix: Probability Theory

Conditional $P(X|E) = \frac{P(X \cap E)}{P(E)}$.

Independence $P(X|E) = P(X)$

Independence $P(X \cap E) = P(X)P(E)$

Chain rule $P(x, y) = P(x)P(y|x)$

Chain rule $P(x_1, ..., x_n) = P(x_1)P(x_2|x_1)P(x_3|x_1, x_2)...P(x_n|x_1...x_{n-1})$ You iteratively condition on more and more.

Bayes Rule $P(x|y) = \frac{P(y|x)P(x)}{P(y)}$ from chain rule $P(x, y) = P(x)P(y|x) = P(y)P(x|y)$

Bayes Rule $P(x|e_1...e_n) = \frac{P(e_1...e_n|x)P(x)}{P(e_1...e_n)}$. If we assume conditional independence $(e_i \perp\!\!\!\perp e_j)|x$ this factors further

Naive Bayes $P(x|e_1...e_n) = \frac{1}{P(e_1...e_n)}P(x)P(e_1|x)...P(e_n|x)$. Because we assume $(e_i \perp\!\!\!\perp e_j|x)$

Marginal from Joint $P(x) = \sum_y P(x, y)$

Marginal from Joint $P(x) = \int_y P(x, y)$

Maximum A Posteriori $MAP(W|e) = \text{argmin}_w P(W, e)$

Marginal MAP $MMAP(W|e) = argmax_w P(W|e)$

6.1 Rewrite Rules

Here's the set of rules you can use to rewrite probability distributions. Since these are rewrite rules, they are *meta-equations*. Each A_i represents one or more variables in an actual equation, and you can apply the rewrite rules as a whole. The variables being conditioned on (B_i) can be dropped from the equation.

$$
\begin{aligned}
& P(A_1, ..., A_n|B_1, ..., B_m) \\
&= \int_C P(C, A_1, ..., A_n|B_1, ..., B_m) && \text{[Introduce Vars]} \\
&= P(A_1, ..., A_{n-1}|A_n, B_1, ..., B_m) * P(A_n|B_1, ..., B_m) && \text{[Introduce Conditioning]} \\
&= P(A_1, ..., A_n, B_1|B_2, ..., B_m) * \frac{1}{P(B_1|B_2, ..., B_m)} && \text{[Remove Conditioning]} \\
\text{if}(A_1...A_n \perp\!\!\!\perp B_1) &= P(A_1, ..., A_n|B_2, ..., B_m) && \text{[Conditional Independence]} \\
\text{if}(A_1 \perp\!\!\!\perp A_2 \perp\!\!\!\perp ... \perp\!\!\!\perp A_n|B_1...B_n) &= P(A_1|B_1, ..., B_m) * ... * P(A_n|B_1, ..., B_m) && \text{[Independence]}
\end{aligned}
$$

6.2 Forward Sampling

When we generate sample $\{X_1 = x_1, X_2 = x_2, ..., X_n = x_n\}$ from a joint distribution $P(X_1, ..., X_n)$ we attempt to find assignments to all of the variables with frequency equal to the probability of that joint assignment in the distribution.

When we have the entire joint distribution, we can compute the Cumulative Distribution Function of the entire joint distribution, and perform the normal procedure of sampling the CDF's x-axis uniformly from 0 to 1 to give us a sample assignment to all the variables in the joint distribution.

When we forward sample, we attempt to assign values to variables iteratively without calculating the entire joint distribution. This, at its heart, is based on the chain rule. A distribution $P(X_1, ..., X_n)$ can be factored into $P(x_1)P(x_2|x_1)P(x_3|x_1, x_2)...P(x$ When we now generate a sample, we can sample x_1 from $P(X_1)$, x_2 from $P(X_2|x_1)$, x_3 from $P(X_3|x_1, x_2)$ and so forth.

In a Bayesian network there exists conditional independences, since if we proceed from the root to the leaves, we can use factorize the joint distribution so that we always sample the parents of a node before the node itself, and we can read the probability of any variable off the CPT for that node.

6.3 Probabilistic Graphical Models features

A PGM is a graph that encodes a joint probability distribution in a compact form by exploiting independences. There are two equivalent views of how this works:

Factorization A graph allows a distribution P to be represented.

I-Map Independencies implied by the graph G holds in the distribution P.

6.3.1 I-Maps and Active Trails

A graph implies an independence map.

6.3.2 d-separation

Two nodes X and Y in a Bayesian Network is d-separated ("directed-separated") if there is no active trail from X to Y. In other words: $dseparation(X, Y|E) \implies independence(X, Y|E)$

6.4 MAP, MMAP and MLE

References

[1] Sharon Lin. Probabilistic color-by-numbers: Suggesting pattern colorizations using factor graphs. In *Proc. SIGGRAPH 2013*, 2013.

[2] Paul Merrell, Eric Schkufza, Zeyang Li, Maneesh Agrawala, and Vladlen Koltun. Interactive furniture layout using interior design guidelines. *ACM Trans. Graph.*, 30(4):87, 2011.

[3] Stuart Russel and 15 Peter Norvig 3rd Edition Ch. 13, 14. *Artificial Intelligence, A Modern Approach*.

[4] Yi-Ting Yeh, Lingfeng Yang, Matthew Watson, Noah D. Goodman, and Pat Hanrahan. Synthesizing open worlds with constraints using locally annealed reversible jump mcmc. *ACM Trans. Graph.*, 31(4):56:1–56:11, July 2012.