PPAML Challenge Problem 4: The Small Problems Collection

Tom Dietterich, Version 9, November 23, 2014

The goal of the "Small Problems Collection" is to create a set of problems that span important dimensions of the space of probabilistic programs in terms of both program formulation and probabilistic inference. We hope that this set of problems can help the PPAML teams identify important tradeoffs in the design and implementation of Probabilistic Programming Systems. (Note that this is an important change in direction from the previous focus on benchmarking of PPS systems.)

This document gives an overview description of each small problem. The PPAML public wiki contains a tar file with all of the supporting documents and files.

Problem Dimensions

At the Kickoff meeting, we described several problem dimensions. Some of those related to application domains. For the Small Problems Collection, we want to ignore those and focus instead on the dimensions that capture abstract problem structure. These include the following:

1. Data types: Continuous, Discrete, Mixed
2. Data structures: Atoms, Vectors, Relations, Grammars, Graphs
3. Model structure: Directed vs. Undirected, Parametric vs. Non-parametric, Fixed vs. Variable number of variables, Presence of latent variables
4. Query type: MAP, Marginal MAP, Expectations, Posterior Distribution, Posterior Summary (e.g., moments), Anomalies
5. Query timing: One-shot (data and query are presented together), Online (fixed query, but data arrive incrementally, so the query results need to be updated incrementally), Amortized (fixed data, but multiple, related queries arrive incrementally, and inference costs can be amortized across them), Online Amortized (both the data and the queries arrive incrementally).

Our focus is primarily on taking a given model plus some observations and answering queries (i.e., as opposed to having a separate learning phase). In this setting, we will say that latent variables are present if there are variables that are neither observed nor queried (and hence, must be marginalized out). This means that the "latent variables" dimension becomes a property of the observations + query rather than of the model, per se.

A matrix placing each of these small problems along these dimensions appears at the end of this document. Many of the problems could be extended in subsequent phases to involve more complex query timing.

Metrics and Submission Guidelines

For each problem, there are one or more accuracy metrics defined. We would like to measure the values of these metrics as a function of CPU time so that we can see the performance profile of the inference procedures. For sampling-based methods, this curve may have interesting structure. For a direct method, it may be a simple step function.

For each inference problem and query, you should submit a CSV file with the name "problem-*p*-query-*q*-metric-*m*.csv" that contains two columns. The first column is the execution time in seconds and the second column is the value of the metric. You should choose your reporting interval (elapsed time between rows) to capture the behavior of your method in less than 1000 rows. You may wish to progressively increase the sampling interval (e.g., quadratically or exponentially) to provide a more informative profile.

In addition, you should submit the source code for your solution as a "tar" archive with the name "problem-*p*-query-*q*-metric-*m*-solution.tar". If all metrics share the same solution, you can just provide a single file for that query. If all queries share the same solution, you can just provide a single file for the problem. We expect to evaluate the source program with respect to such measures as compactness, elegance, comprehensibility, and extensibility.

For problem 8, instead of submitting performance profiles, you should just submit the output produced by the python evaluation script under the file name "problem-8-metrics-solution.txt".

Problems 6 and 9 are expressiveness challenges. The primary requirement is to demonstrate a probabilistic program and show that it runs and computes the right answer. Teams should submit their source code as file "problem-*p*-solution.tar". Teams may optionally produce performance profiles for a metric of their choice as well. Please define the metric in a file names "problem-*p*-query-*q*-metric.pdf".

Initial Problem Collection

1. Bayesian Linear Regression

Given:

A set of training points $\{(x_i, y_i)\}_{i=1}^{N}$ where $x \in \Re^d$ and $y \in \Re$

Prior probability distribution over the β and σ parameters of $y \sim Norm(\beta \cdot x, \sigma)$. Priors will be diagonal Gaussian for β and Gamma for σ^{-1}

Find:

Query 1: The posterior distribution of β.

Metrics:

Metric 1: Expected squared Euclidean distance between the predicted mean $\hat{\beta}$ and the true mean β, where the expectation is taken with respect to the posterior distribution.

Metric 2: Total variation distance between the computed posterior and the correct posterior over β.

2. Medical Diagnosis

Given:

A bipartite graph of variables linking disease states to observable symptoms (along with appropriate conditional probability models, e.g., for QMR-DT)

The observed values of a subset of the findings (also called symptoms)

The cost of treating each disease

The cost of observing each finding

Find:

Query 1: Posterior distribution over the disease state variables

Query 2: Joint MAP value of the disease state variables

Query 3: Expected value of information for observing each (not-yet-observed) finding

Metrics:

Metric 1: Total variation distance between the true posterior and the posterior output by the probabilistic program.

Metric 2: Hamming distance between the true disease states and the predicted MAP disease states.

Metric 3: Squared error between the true and computed expected value of information for each finding, summed over all unobserved findings.

3. Discrete time Hidden Markov Model

Given:

Discrete time HMM (including transition matrix, initial state distribution, and emission probabilities)

Observed sequence of emissions

Find:

Query 1: MAP state sequence.

Query 2: Marginal posterior distribution of states at each time step given all of the observations (smoothing)

Query 3: Marginal posterior distribution of states at each time step given only the observations up to and including that time step (filtering)

Metrics:

Metric 1: Hamming distance between the true and the computed MAP state sequence

Metric 2: Smoothing: Total variation distance between the true and computed marginal posteriors at each time step. Mean and variance of this over all time steps.

Metric 3: Filtering: Total variation distance between the true and computed marginal posteriors at each time step. Mean and variance of this over all time steps.

4. Latent Dirichlet Allocation under Hierarchical Dirichlet Prior

Given:

A corpus of documents.

The hyper-parameters for the HDP-LDA model

A set of partial documents in which exactly half of the words have been removed.

Find:

Query 1: For each word not appearing in the partial document, compute the marginal probability that it appears in each of the partial documents.

Query 2: For each partial document, compute the MAP completion of the document.

Metrics:

Metric 1: $\sum_w \mathbb{I}[w \in d_i] - P(w|D, \ell_i)$, which is the difference between the indicator variable for whether word w appears in test document d_i and the predicted probability that it appears in the document.

Metric 2: Hamming distance (computed at the token level) between the true document and the predicted document. For example, if the word “pickle” appears 3 times in the ground truth document and it is predicted to appear only once, then the Hamming distance is 2.

5. Conditional Queries for Probabilistic Context-Free Grammars

Given:

PCFG G

Two strings x and y

Find:

What is the conditional probability that the grammar G will generate $S = xy$ given that the string begins with the prefix string x? See the details file for the grammar and the queries.

Metrics:

Square of the difference in negative log probability ("surprise") between the true and the computed conditional probability.

6. Network Analysis

This is an expressiveness challenge. The challenge is to represent a probabilistic model that generates an undirected graph. Nodes are added sequentially to the graph, and edges are added according to a mixture of two approaches: random attachment (each new node is randomly attached to existing nodes) and preferential attachment (where each new node is attached to existing nodes according to a "rich-get-richer" preference). Full details of the model are provided on the CP4 web page.

The challenge is to express the model in such a way that constraints can be placed on such properties as the clustering coefficient and the number of edges while still permitting reasonably efficient inference.

7. Friends and Smokers (Undirected Graphical Model)

Given:

An undirected graph in which the nodes represent people and an edge connects two people if they are friends (assumed to be symmetric). We are told that all else being equal, friends are 3 times more likely to have the same smoking habit (both be smokers or both be non-smokers) as to have different smoking habits). The prior probability of being a smoker is 0.2.

The observed smoking habit status of a subset of the nodes in the graph

Find:

Query 1: For each of the unobserved nodes, the marginal posterior probability that that person is a smoker.

Metrics:

Metric 1: The sum, over all of the queried nodes, of the absolute difference between the computed and the true posterior probability.

8. Seismic Event Detection

Given:

A model of signal propagation through a medium (i.e., as a consequence of point events occurring at various locations and with various magnitudes in the medium)

A model of signal measurement for instruments placed at various locations in the medium including false detection probability and true detection probability (as a function of signal strength)

A model of the prior probability of the point events (underground nuclear tests).

Time series of observations for each instrument

Find:

Queries: For each of 25 test episodes, compute a "bulletin", which consists of the MAP number, location, magnitude, and time of each point event

Metrics:

Evaluation code is provided that first computes a best matching between the predicted and the actual events for an episode and then computes (Metric 1) precision, (Metric 2) recall, and (Metric 3) average error.

9. Recursive Reasoning: Scalar Implicature

This is an expressiveness challenge.

Given:

Two mutually-recursive models (of a speaker and a hearer). There is a world in which there are three seeds that may have sprouted, so the world is in one of four states: 0 seeds sprouted, 1 seed sprouted, 2 seeds sprouted, or 3 seeds sprouted. The speaker observes this world and wishes to communicate the state to the hearer (who does not directly observe the world). The speaker can say one of three possible sentences: "All of the seeds have sprouted", "Some of the seeds have sprouted", or "None of the seeds have sprouted". The speaker should choose the utterance that is most likely to cause the hearer to infer the correct state of the world. The hearer is supposed to infer the state of the world based on what the speaker is saying knowing that the speaker is trying to communicate the correct state of the world (hence the recursion). Assume a uniform prior on the four possible states of the world.

Do:

Express this problem in the PPS with an explicit limit on the degree of the recursion. If the speaker says "Some of the seeds have sprouted" and the recursion is at least of depth 2, then the model should infer that with high probability the world is in either states 1 or 2 but not state 3 (because the speaker would have said "All of the seeds have sprouted" instead).

This is an expressiveness challenge, so no performance statistics are required. It is sufficient to show that your PPS can compute the correct answer.

For more detail on this and related problems, see https://probmods.org/inference-about-inference.html

10. Lifted Inference

Given:

A probabilistic program directly implementing the generative model:

$$\text{cause} \sim \text{Bernoulli}(0.01);$$
`for all` i `in` $\{1, \ldots, n\}$
$$\text{effect}_i \sim \text{Bernoulli}(0.6), \texttt{ if } \text{cause},$$
$$\text{Bernoulli}(0.05), \texttt{ if not } \text{cause}$$

Find:

Query: The exact value of $P(\text{cause}|\ \#\{\ i:\ \text{effect}_i\} = k)$. Here, $\#set$ returns the cardinality of the set.

Metric:

Time required to answer the query for values of
$n \in \{10, 20, 40, 80, 160, 320, 640, 1280, 2560, 5120\}$

$k \in \{1, 2, 4, 8, 16, 32, 64\}\ k < n$

Instead of a performance profile for each of these values, please submit a single CSV file named "problem-10-metrics.csv" with three columns: n, k, and running time (in seconds with decimal fraction) of your solution.

Note that ideal running time should scale as $O(\log(n + k))$