Learning to do Inference

Frederik H. Eaton
frederik@ofb.net

6 June 2014

Abstract

In this paper we study the problem of learning to do approximate inference. We propose a protocol allowing two inference algorithms to communicate about a shared model, with one algorithm serving as a teacher and the other as a student. We derive several heuristics for choosing data to transfer under this protocol, and perform experiments to evaluate the effectiveness of each one. We do not propose a concrete inference algorithm which could serve as the student. Instead, we attempt to estimate the ideal maximum efficiency of each heuristic under all possible students, substituting a synthetic student which simulates computational uncertainty by reasoning over the space of models.

1 Introduction

The act of learning in statistical AI is typically understood as a process of reproducing a statistical distribution by observing data. In this paradigm the student acquires data to which he previously had no access, and then attempts to model this data. Here we extend the idea of learning to a broader setting, in which the student already has access to the relevant data, but may lack the computational resources to query it in some desired way. We are interested in learning "how", as contrasted with learning "what". An example domain would be a game like chess, in which the constraints on moves can be quickly understood, but where becoming a good player takes considerable additional learning. We might consider a better domain from a theoretical standpoint to be that of statistical inference, because it is general enough to reduce most problems of interest, and because it represents uncertainty explicitly. This paper is addressed to learning in the inference framework.

The ability for a computer program to learn to do computation should have wide application, because so many useful computations are intractable. The catch is that our notion of learning presupposes a teacher who already possesses at least some of the skills desired by the student. The alternative, in which learning is carried out solitarily, as a form of experimentation or self-adaptation, is also interesting but misses the opportunity to incorporate the results of previous effort. Examples of this simpler kind of adaptation can already be found in approximate inference: for example, in a message-passing algorithm, one can think about using heuristics or experimentation to optimize the order in which messages are sent [4], or the representation of messages in continuous variable models [3]. In MCMC algorithms, parameters of transition kernels can be adaptively tuned in ways that preserve stationarity [9].

On the other hand, the learning "how" that takes place between a student and a teacher has, to our knowledge, yet to be explored, and it poses difficulties of a practical nature owing to the need for an intelligent teacher. The problem is that either the teacher is a human or some other kind of black-box oracle to be found in the environment, in which case the domain is likely to be very specialized; or that the teacher is another algorithm, and then one asks why this algorithm shouldn't simply be run a bit longer, rather than transferring its knowledge to a student. The first possibility will be touched upon again in the discussion section, but we are more interested in the second, where the teacher is another program. Here a number of scenarios could be imagined. Perhaps an inference

problem is being attacked on many processors in parallel, and there is a need to merge the results without losing information. In this case, each intermediate result could transfer its knowledge by serving as a teacher to an approximation which then becomes the final output. Another possibility is that of having two very different varieties of algorithms merge their best qualities together through learning. Or perhaps the learning protocol itself could serve as the backbone of an evolutionary adaptive algorithm. In any case we hope that the idea of learning to do inference is sufficiently compelling to justify investigating for its own sake.

We are interested in the problem of approximate inference, rather than exact inference, because it seems to provide greater scope for learning. We consider that an approximate inference algorithm is given a fixed amount of time in which to provide the most accurate available estimate of each variable's marginal distribution (or conditioned marginal distribution, if provided a set of additional variable assignments). Such an algorithm would want to learn how to produce more accurate marginals in the time available to it.

However, because no approximate inference algorithm has yet been developed which is capable of learning from a teacher, we are obliged to begin by studying the learning protocol in isolation. We propose a protocol by which information could be transferred between two algorithms, and simulate the use of this protocol under various heuristics for choosing which information to transfer. Without a specific algorithm to serve as a student, we simulate an optimal student using uncertainty over the space of models as a substitute for actual computational limitations. Note that the student in our framework has access to the full model specification, since it is not a question of learning "what" but learning "how". The synthetic student of our experiments pretends not to know these model parameters, but maintains a distribution over possible models and updates this distribution whenever he receives information from the teacher.

The design of our protocol follows from the intuitive assumption that the information to be transferred between teacher and student consists of locations of regions of high or low probability in the model. For simplicity, these locations are identified using full variable assignments, or "states", of the model. Both teacher and student are inference algorithms, and the heuristic which chooses states to transfer has access to the marginals and conditioned marginals of both algorithms.

2 Related work

Although the present paper is the first, to our knowledge, to explore learning to do inference in the presence of a teacher, there is a variety of existing work which relates tangentially to our effort. We have already cited some work on tuning parameters of MCMC and message passing algorithms.

Learning about a model by collecting a list of interesting states is related to the work of Rasmussen and Ghahramani [8], which fits a kernel-based approximation to a density function given a collection of samples. Their kernel method is shown to be superior to simply treating the samples as probabilistic estimates, which is in agreement with our approach. But there is no teacher, and samples are chosen randomly. Their method is demonstrated on continuous models with few variables, and unfortunately we know of no kernel which could give tractable mass estimates in our discrete, multi-variate setting.

The work of Santos and Shimony [10] attempts to perform inference in discrete models by collecting states of high probability, although unlike our work it only applies to directed networks. We examine similar heuristics in our own setting in Section 8.

Finally, there are some approaches to building models from data by representing the empirical distribution as a collection of modes, or regions of high probability [6, 11]. Our own work assumes a model has already been supplied, and is able to associate example states with both high and low probability mass. What is common to both efforts is a state-wise decomposition of distribution function.

Building models by collecting states could be contrasted to variable-wise decompositions, which might encode independencies as in Cut-set Conditioning [7]. Such "multiplicative" decompositions could provide the basis for a complementary version of the protocol we present in this paper, which in effect decomposes a model additively.

The synthetic student of Section 5 is performing a kind of "tensor recovery", which is the problem of filling in the unknown entries of a tensor when some of them have been observed, and has been well studied [2].

3 Background

We assume that we are given a model specified by a "factor graph", which is a general representation for statistical models [5]. A factor graph defines a distribution over n variables $x := (x_1, \ldots, x_n)$ as a normalized product of non-negative factors ψ_α:

$$P(x) = \frac{1}{Z} \prod_\alpha \psi_\alpha(x_\alpha) \tag{1}$$

where α indexes a collection of sets of variables and Z is a normalizing constant, also called the "partition function". We will refer to $ZP(x) = \prod_\alpha \psi_\alpha(x_\alpha)$ as the *unnormalized joint* (at state x). Note that ZP is always tractable to compute. The object of approximate inference is to estimate the marginals of the model:

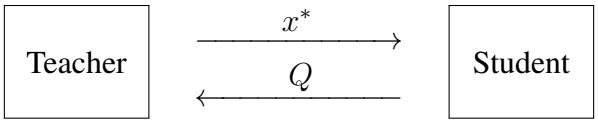
$$P(x_i) = \frac{1}{Z} \sum_{x_{\setminus i}} \prod_\alpha \psi_\alpha(x_\alpha) \tag{2}$$

as well as conditioned marginals $P(x_i|x_C)$, which are the same as marginals of a conditioned model $\prod_\alpha \psi_\alpha(x_\alpha) \prod_c \delta(x_c, x_c^*)$.

4 Protocol

We now define the learning protocol. This protocol takes place between a "teacher" and a "student". It is motivated by the idea that the student, in exploring areas of the probability distribution defined by a model, might miss certain regions of importance. The teacher could then call his attention to these regions using the communication protocol. In theory we could define a region of state space using any number of representations, such as a partial assignment or even another model.

This paper considers the simplest scenario, representing a region using a full assignment of values to variables, which we call a state. This representation, in addition to its simplicity, has the added advantage that it obviates the need for the teacher to include any notion of probability mass in his transmission. This is because the student can easily measure the relative mass of different states using ZP (see above) from his own copy of the model specification. Thus, the student never has to duplicate any of the teacher's opinions about the marginals or partition functions, and so is allowed to exceed the accuracy of the teacher.



The interaction cycle is formalized in the following pseudo-code:

Protocol 1. *Repeat for an arbitrary number of turns.*

At turn m:

1. *Student proposes an approximate distribution Q based on the example states and unnormalized joints he has seen so far: $\{(x^{*(i)}, ZP(x^{*(i)}))\}_{i=1:m-1}$*
2. *Teacher selects a new example $x^{*(m)}$ (perhaps in response to errors in the student's distribution)*

5 Implementation of Synthetic Student

Our experiments (Section 8) will explore various heuristics for choosing an example state to pass from teacher to student under the learning protocol. The state will depend in some way on marginals

currently expressed by the student, which should in turn depend on states previously transmitted by the teacher. Rather than using a particular approximate inference algorithm to calculate these marginals, we simulate an optimal student using exact inference on a distribution over possible models. In this way we hope to obtain generally applicable results.

The idea behind our simulation of the student is that, although the student knows the entire model specification and could in theory evaluate ZP at any desired state, we pretend that he is only aware of the values of ZP at those example states shown to him by the teacher, and knows nothing else about the model. The student maintains a distribution over possible models, which is conditioned to agree with ZP at the example states. He uses this conditioned distribution to produce new marginals for the teacher. We can argue that the student defined in this way is making optimal use of the information provided by the teacher, without doing any additional exploration of his own.

The student's distribution, for simplicity, assumes the model to be a fully connected binary pairwise factor graph, meaning that all variables are *binary*, all factors have size two (*pairwise*), and there is one factor for every pair of variables (*fully connected*). The potentials of the graph are distributed as exponentials of normal random variables, which is a traditional way of generating models for experiments in approximate inference. We write $\psi_{ij}(x_i, x_j) = \exp(\beta W)$, where $W \sim N(0, 1)$.

Initially the potentials are believed by the student to be sampled independently, but when he incorporates his set of observations of the unnormalized joint $\{(x^{*(i)}, ZP(x^{*(i)}))\}_{i=1:m-1}$, then correlations will be introduced in his beliefs. If he represents the log-potentials using a multivariate normal distribution, then these correlations can be represented in a covariance matrix, and after each observation the posterior of his beliefs will be in the same class as the prior (i.e., it is a conjugate prior). The observations

$$z^* = ZP(x^*) \tag{3}$$

are equivalent to

$$z^* = \prod_{jk} \psi_{jk}(x^*_{jk}) \tag{4}$$

$$\implies \log z^* = \sum_{jk} \log \psi_{jk}(x^*_{jk}) \tag{5}$$

For our distribution over models, the quantities $\log \psi_{jk}(x_j, x_k)$ are distributed according to a multivariate normal distribution with mean μ and variance Σ (indexed by $(j, k > j, x_{jk})$ and initialized to $\beta^2 I$). Then (5) is a statement that some subset of the dimensions of this normal distribution should have a certain sum (namely $\log z$). A set of such constraints is in turn a special case of a linear constraint, say

$$B \cdot y = v, \tag{6}$$

on draws y from a multivariate normal, where B is a matrix and v a vector. More specifically, in our experiments, y is indexed by $(j, k > j, x_j, x_k)$ and represents a vector of log potentials specifying the whole model, while B contains entries which are 0 or 1 according to whether a particular potential entry contributes to a given state, and v is a column vector of the $\log$ unnormalized joint entries corresponding to each example $x^{*(i)}$:

$$v_i = \log z^{*(i)} = \log ZP(x^{*(i)}) \tag{7}$$

Conditioning on this linear constraint is equivalent to transforming the mean and the variance of the multivariate normal distribution:

$$\mu' = \mu - \Sigma B^T (B \Sigma B^T)^{-1} (B\mu - v) \tag{8}$$

$$\Sigma' = \Sigma - \Sigma B^T (B \Sigma B^T)^{-1} B \Sigma \tag{9}$$

Although we were able to discover an analytic form for the updates to the distribution parameters in this case, there appears to be no simple analytic expression for the expected marginals of factor graphs drawn from the distribution. So, when quantities such as expected marginals are needed, we simply draw many sample graphs, compute their marginals, and average together the results. This averaging seems to be more sensible in the log domain since some marginals may be very close to 0 or 1. Averaging sampled marginals in the log domain is equivalent to taking the geometric average and renormalizing.

6 Optimal Teacher

Drawing on the work of the previous section, we derive a criterion which the teacher can use to maximally reduce one measure of the student's error. This requires examining the student's beliefs at every state, and so is not intended to be useful in practice, but it is useful to us as a benchmark in our experiments.

Below, we write $Z\tilde{P}(x)$ for a random variable representing a draw from the student's beliefs about $ZP(x)$, while $Z\hat{P}(x)$ represents an estimator of $ZP(x)$. We use a geometric average unless stated otherwise: $Z\hat{P}(x) \equiv \exp \mathbb{E}[\log Z\tilde{P}(x)]$.

We can derive an analytic expression for the additive change in μ when a new example x^* is seen. From equation 8 we have

$$\Delta\mu = \Sigma\Delta B^T(\Delta B\Sigma\Delta B^T)^{-1}\Delta \log Z\hat{P}(x^*) \tag{10}$$

where ΔB encodes the new constraint $z^* = ZP(x^*)$ and $\Delta \log Z\hat{P}(x^*) = B\mu - v$ is the difference between the old and new estimates of $ZP(x^*)$. Note that $\Delta B\Sigma\Delta B^T$ is just the variance of $\log Z\tilde{P}(x^*)$. If Σ is initialized to a positive multiple of the identity matrix, $\beta^2 I$, encoding a spherical normal distribution, then $\beta^{-2}\Sigma$ will be a projection and we will have $\Sigma\Sigma = \beta^2\Sigma$. This allows us to write a simple expression for the L_2 norm of $\Delta\mu$, which we obtain by multiplying equation 10 by its transpose and taking the square root:

$$||\Delta\mu||_2 = \beta\frac{\left|\Delta \log Z\hat{P}(x^*)\right|}{\sqrt{\mathrm{Var}(\log Z\tilde{P}(x^*))}} \tag{11}$$

Since each update brings μ closer to the parameters of the true model, this equation tells us how to find updates which will maximize one measure of the speed of convergence of the student's representation: We should choose states x^* for which the error in his point estimate of $\log ZP(x^*)$, relative to the standard deviation of the same, is greatest. This heuristic is called "ML2DM" in the experiments ("max L_2 delta μ"). It is related to Mahalanobis distance. Note that although the distribution and its intersection with subspaces are both spherical, its projections along axes are not. Thus the denominator of (11) is not constant but depends on the overlap between x^* and the example states observed by the student.

7 Variable-Greedy Teacher

The "optimal teacher" or ML2DM, as noted above, is not computationally useful, since if we could examine every state in a model then we could also do exact inference in that model. In this section we describe a straightforward variant which only uses the marginals and conditioned marginals expressed by the student and teacher. We call this method "CG", since it turns out to be related to a method for comparing the accuracy of two approximate inference algorithms which is called the "Conditional Game" [1].

If we ignore the denominator in the ML2DM criterion (11), the teacher searches for a state whose true $\log ZP$ is as much larger or smaller than the student's estimated $\log Z\hat{P}$ as possible. This is to say that the ratio of the teacher's to the student's ZP is biggest or smallest at that state. A faster approach which only uses the variable marginals would be to look for the (variable, value) pair (i, x_i^*) where the ratio of the teacher's to the student's marginal is as big as possible, i.e. maximizing $P(x_i^*)/Q(x_i^*)$ where Q represents the student's marginals. Then, conditioning the model on $x_i = x_i^*$, we would have the teacher and student generate new (conditioned) marginals and repeat, until all variables are assigned and a full state x^* is obtained. In other words, over n turns we choose a new variable i_t and value $x_{i_t}^*$ according to:

$$(i_t, x_{i_t}^*) = \underset{\substack{(j,x_j)\\ j\notin i_{1:t-1}}}{\mathrm{argmax}} \frac{P(x_j|x_{i_{1:t-1}}^*)}{Q(x_j|x_{i_{1:t-1}}^*)} \tag{12}$$

This approximately maximizes the expression

$$\frac{ZP(x)}{ZQ(x)} = \frac{P(x_{i_1})}{Q(x_{i_1})}\frac{P(x_{i_2}|x_{i_1})}{Q(x_{i_2}|x_{i_1})}\cdots\frac{P(x_{i_n}|x_{i_1}\ldots x_{i_{n-1}})}{Q(x_{i_n}|x_{i_1}\ldots x_{i_{n-1}})} \tag{13}$$

by greedily maximizing each term on the right hand side in turn. The left-hand side is just the exponential of $\Delta \log Z\hat{P}(x^*)$ of 11.

It would also make sense to minimize the ratio of the teacher's to the student's marginal, as long as the same operation (namely maximization or minimization) is employed consistently when generating each state. Either version is an instance of the Conditional Game.

8 Experiments

In our experiments, we compare five different methods for the teacher to choose example states at which to update the student's distribution over potentials.

- **max L_2 delta μ** - (ML2DM) Chooses states which maximize $||\Delta\mu||_2$ according to equation 11.
- **max diff entry** - (MDE) At each turn, all states are examined and the state is chosen at which the entry in the student's unnormalized joint estimate is most different from the teacher's (exact) unnormalized joint, e.g. $\mathrm{argmax}_x \left|ZP(x) - Z\hat{P}(x)\right|$. (This is similar to ML2DM above, but using $\left|\Delta Z\hat{P}(x^*)\right|$ instead of $\left|\Delta \log Z\hat{P}(x^*)\right|$, and without weighting the states by the the inverse of the standard deviation of $\log Z\tilde{P}(x^*)$.)
- **conditional game** - (CG) The student's marginals (their geometric average over samples drawn from his distribution over models) are used to play in a conditional game against the teacher. Whether the teacher is trying to maximize or minimize is decided uniformly at random before each game. The state chosen by the game is used as the next example. The teacher employs an exact distribution (except in section 9.1, which describes experiments with an approximate teacher).
- **max var log entry** - (MVLE) The state is chosen at which the student is maximally uncertain about the value of $\log ZP(x)$, as measured by the variance of this value over sampled models. I.e. $\max_x \mathrm{Var}(\log Z\tilde{P}(x))$
- **uniform random** - (UR) A state is chosen uniformly at random.

The first three methods (ML2DM, MDE, and CG) compare estimates from the student's distribution with the true distribution. The the fourth (MVLE) only uses information from the student, and fifth method (UR) makes no reference to either.

We tested the five methods (ML2DM, MDE, CG, UR, MVLE) on four different models. The models all fully-connected with 11 binary variables, but differ in the standard deviation β of the log-potentials. The potentials were drawn independently as exponentiated normals: $\psi_{ij}(x_i, x_j) = \exp(\beta W)$ where $W \sim N(0, 1)$. We explored values of 0.125, 1, and 3 for β. For higher values of β, most of the probability mass is placed on a few dominant states, whereas for lower values it tends to be spread out across many states (see Figure 6 in the Supplementary Material).

In a fully-connected binary pairwise factor graph of n variables, there are $\frac{n(n+1)}{2}$ parameters in the potentials, thus we expect most methods to require 66 examples to learn our 11-variable models completely.

The student's distribution over models uses as a prior the same normal parameters from which the true graph was generated. Using other parameters (including putting a normal prior on univariate factors ψ_i, and on a scalar factor for the whole graph) did not produce substantially different behavior.

9 Results

The outcomes of running the five methods on the $\beta = 1$ model are shown in Figure 1. We have plotted $L_1^{\log}$ error of marginals (defined as $\sum_{x_i} |\log P(x_i) - \log Q(x_i)|$ [1]), but L_1 error gives similar results. The student's marginals are calculated by geometrically averaging the marginals of 256 sample models drawn from the student's distribution, and renormalizing. Errors were averaged over 10 random models. Using estimates from these 10 samples, we show ± 1 standard deviation error bars for CG, MDE, and ML2DM; error bars are omitted for MVLE and UR, for plot readability, but would appear slightly wider than the others if shown. Sample counts are grouped into bins of size four. (Plots for $\beta = 0.125$ and $\beta = 3$ are in the Supplementary Material, Figure 3)

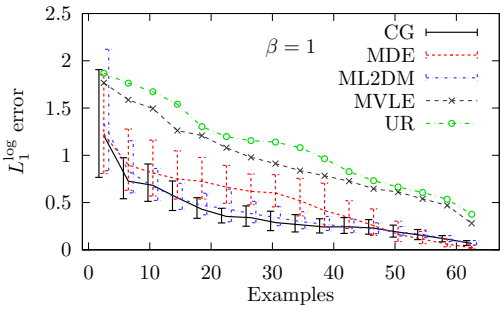


Figure 1: Student's error as a function of example count, for $\beta = 1$.

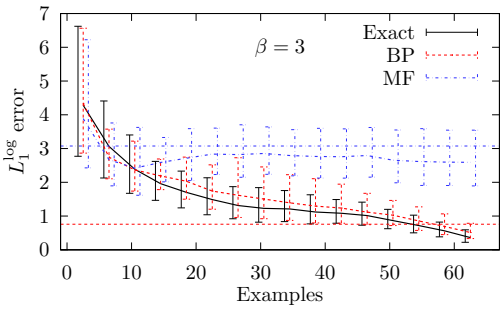


Figure 2: Learning curves for approximate teachers, for $\beta = 3$.

The plots demonstrate a number of consistent relationships between the five methods. MVLE is close to UR, although usually better, and both perform relatively poorly. CG and ML2DM seem to have the best overall performance. Although ML2DM sometimes does better than CG, they are generally indistinguishable. For small β, CG and ML2DM are clear winners. For $\beta = 1$ or 3, they sometimes trail MDE, but without ever being far behind. On the other hand, there are conditions under which MDE performs significantly worse than CG.

It is perhaps surprising that CG does so well, since unlike the other methods (all except UR) it does not require examining all entries of the unnormalized joint ZP. It is also interesting that CG overlaps so closely with ML2DM. The CG can be seen as a greedy procedure for finding the biggest $\Delta \log Z\hat{P}(x^*)$, while ML2DM directly maximizes a scaled version of the same quantity. We might have predicted that this distinction would give the two methods very different behavior.

A number of other methods for choosing examples were explored but were found to perform poorly, and are not shown in the plots. Choosing states of the factor graph in decreasing order of the true joint distribution at each state, so that only the states of highest probability mass are shown to the student, also performs very poorly. This is because the examples generated, while distinct, yield degenerate constraints and so are uninformative. Choosing an example at random from the exact distribution performs about as well as UR, as does choosing a random state from a model drawn from the student. Choosing a state by selecting the variable and value with the largest $\mathrm{Var}(\sum_{x_{\setminus i}} Z\tilde{P}(x))$, conditioning the variable to that value and recursing, works about as well as MVLE. Rather than finding the state with maximum difference in entry, as in MDE, if we were to choose the state with maximum difference in log entry, the result is similar and sometimes better.[1] The application of the conditional game we used chooses randomly to have the teacher maximize or minimize the value. If instead we eliminate the random choice and select each point with the teacher only maximizing or only minimizing, then the error does not decrease - in most instances the same state is chosen repeatedly. For the CG, we found worse performance when having the student use marginals from a sampled $Z\tilde{P}$ rather than an averaged estimator $Z\hat{P}$, as well as when using arithmetic rather than geometric averaging of the marginals. Modifying the CG so that the teacher iteratively chooses a (variable, value) pair (i, x_i^*) maximizing $P(x_i^*) - Q(x_i^*)$ rather than $P(x_i^*)/Q(x_i^*)$ gave similar but slightly worse results compared to the standard CG.

[1] XXX This will be clarified in the final version of the paper [Post-submission update: experiments showed that the just-proposed "maximum difference in log entry" generally starts off better but ends worse than MDE. Its performance is similar to CG.]

9.1 Approximate Teacher

It is interesting to ask how the student performs when the teacher's beliefs differ from the true distribution. We can get an idea by using the CG method with approximate message-passing algorithms such as Belief Propagation (BP) and Mean Field (MF) to generate the teacher's marginals. (Other methods such as ML2DM are not so meaningful in this case, since they don't make use of marginals.) In Figure 2, we plotted the "learning curve" with the CG method for three different choices of teacher: with exact beliefs (as before), and with beliefs calculated from the BP or MF approximations. The $L_1^{\log}$ error of BP and MF are shown by horizontal lines. For $\beta = 3$, the most difficult model, with MF it is easy to see that the student's performance is being held back by the teacher, although it is possible for the student to improve on the teacher to a limited extent. For BP the situation is apparently similar but the effect is less pronounced because of BP's higher accuracy. Since it is possible to achieve perfect accuracy for the student simply by choosing 66 random states (which are almost certain to be linearly independent), we would guess that the reason for the student's accuracy to plateau in this experiment is that the teacher is showing the same state repeatedly. This is indeed the case (see Figure 5 in the Supplementary Material). The student does not seem limited by the teacher's inaccuracy for the easiest model, $\beta = 0.125$ (Figure 4 in the Supplementary Material).

10 Discussion

We considered the problem of learning in the inference setting, using one approximation to instruct a second approximation, and proposed a simple learning protocol by which such an interaction could be carried out. In this protocol, a teacher selects example states to show to a student, based on perceived errors in the student's approximation. We examined five methods for choosing example states in this protocol. One, "ML2DM", was based on an analytically-derived criterion which, although intractable, is definably optimal within each round. Two were based on intuitive metrics of a similar complexity. The fourth chose points at random, as a simple benchmark for comparison. The last method was based on the "conditional game" (CG) of Eaton. We found that ML2DM had the best overall performance, but that the more tractable CG method was comparable and occasionally superior. We also examined the performance of CG in the case where the teacher is governed by an approximation such as BP or MF rather than by exact marginals. We found that in this case the student's performance was reduced, but that he was still able to outperform the teacher in accuracy.

The results are interesting because the CG was designed for the purpose of determining the more accurate of two inference algorithms. The fact that it naturally generalizes to the learning task could be seen as a sort of validation, in an artificial or computational setting, of the use of debate in pedagogy (e.g. Socratic dialog).

Because the protocol is so simple, it may be considered to contribute little to the larger goal of approximate inference. Ultimately we would like to develop algorithms which can automatically adapt to problem complexity; whether this can be done by combining the outputs of parallel threads using the protocol we proposed (or something similar) remains to be seen. However, if one were motivated to proceed in that direction, one might be inspired by the present work to think along the lines of an evolutionary computation in which natural selection and sharing of information are unified into one operation, something like the conditional game.

We end by noting that although our primary interests were computational, it is not difficult to imagine using our protocol with a human teacher, in some instances. For example, consider the model where the first variable contains the identity of the first move in a game of chess; the second variable encodes the second move, and so on; and where $ZP(x) = 1$ if white wins, say, and some small number otherwise. In this case the CG represents not the chess game itself, but a betting game concerning the utility of each move. Upon reflection, the result of using the CG with our protocol would be seen to be closer to the way that games like chess are actually taught, with in-depth move by move analysis, rather than simply playing multiple games against an expert.

References

[1] F. Eaton. A conditional game for comparing approximations. In *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, volume 15, 2011.

[2] Silvia Gandy, Benjamin Recht, and Isao Yamada. Tensor completion and low-n-rank tensor recovery via convex optimization. *Inverse Problems*, 27(2):025010, 2011.

[3] Nicolas Heess, Daniel Tarlow, and John Winn. Learning to pass expectation propagation messages. In *Advances in Neural Information Processing Systems*, pages 3219–3227, 2013.

[4] Jiarong Jiang, Taesun Moon, Hal Daum III, and Jason Eisner. Prioritized asynchronous belief propagation. In *ICML Workshop on Inferning: Interactions between Inference and Learning*, June 2013.

[5] F.R. Kschischang, B.J. Frey, and H.A. Loeliger. Factor graphs and the sum-product algorithm. *IEEE Transactions on information theory*, 47(2):498–519, 2001.

[6] Christopher Meek, Bo Thiesson, and David Heckerman. Staged mixture modelling and boosting. In *Proceedings of the Eighteenth conference on Uncertainty in artificial intelligence*, pages 335–343. Morgan Kaufmann Publishers Inc., 2002.

[7] J. Pearl. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann, 1988.

[8] C.E. Rasmussen and Z. Ghahramani. Bayesian Monte Carlo. *Advances in neural information processing systems*, pages 505–512, 2003.

[9] Gareth O Roberts and Jeffrey S Rosenthal. Examples of adaptive MCMC. *Journal of Computational and Graphical Statistics*, 18(2):349–367, 2009.

[10] Eugene Santos and Solomon Eyal Shimony. Belief updating by enumerating high-probability independence-based assignments. In *Proceedings of the Tenth international conference on Uncertainty in artificial intelligence*, pages 506–513. Morgan Kaufmann Publishers Inc., 1994.

[11] Jakob J Verbeek, Nikos Vlassis, and B Kröse. Efficient greedy learning of Gaussian mixture models. *Neural computation*, 15(2):469–485, 2003.

A Supplementary Material

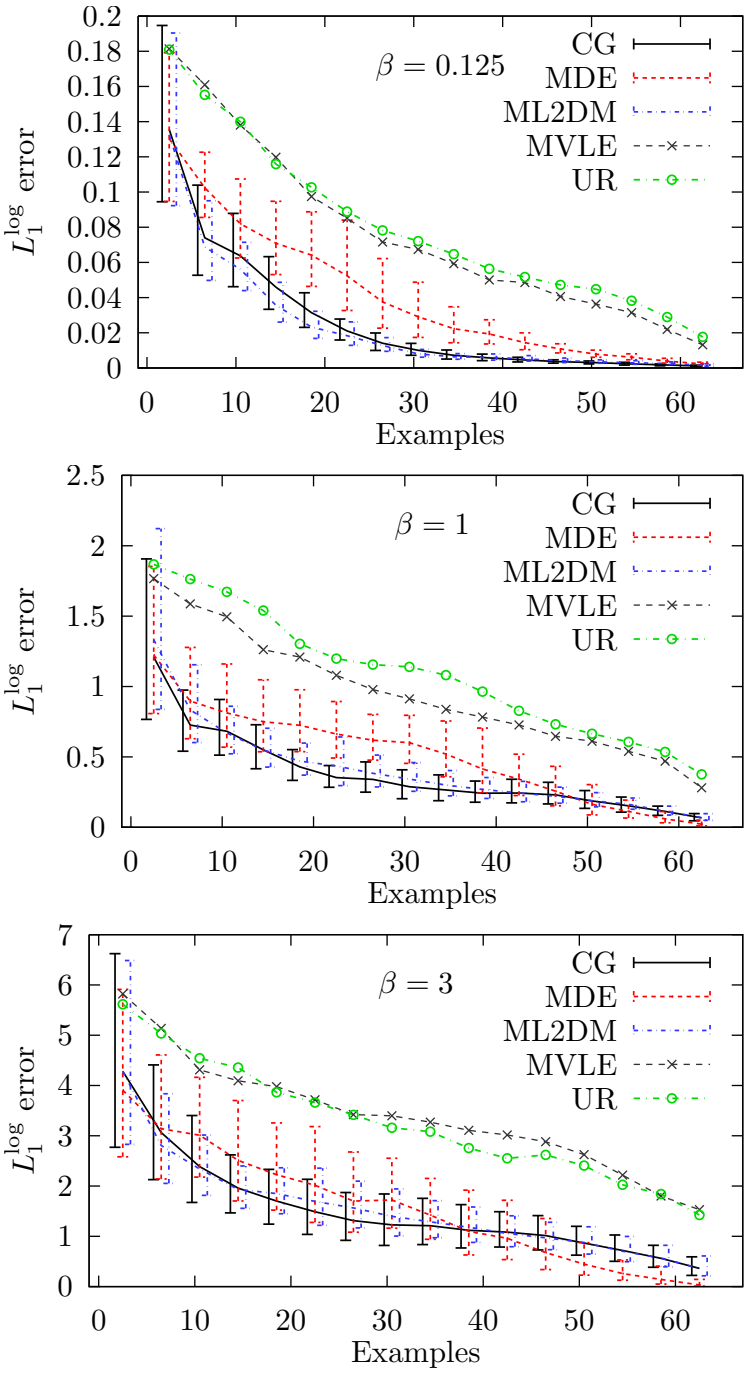


Figure 3: Student's error as a function of example count, for three values of β.

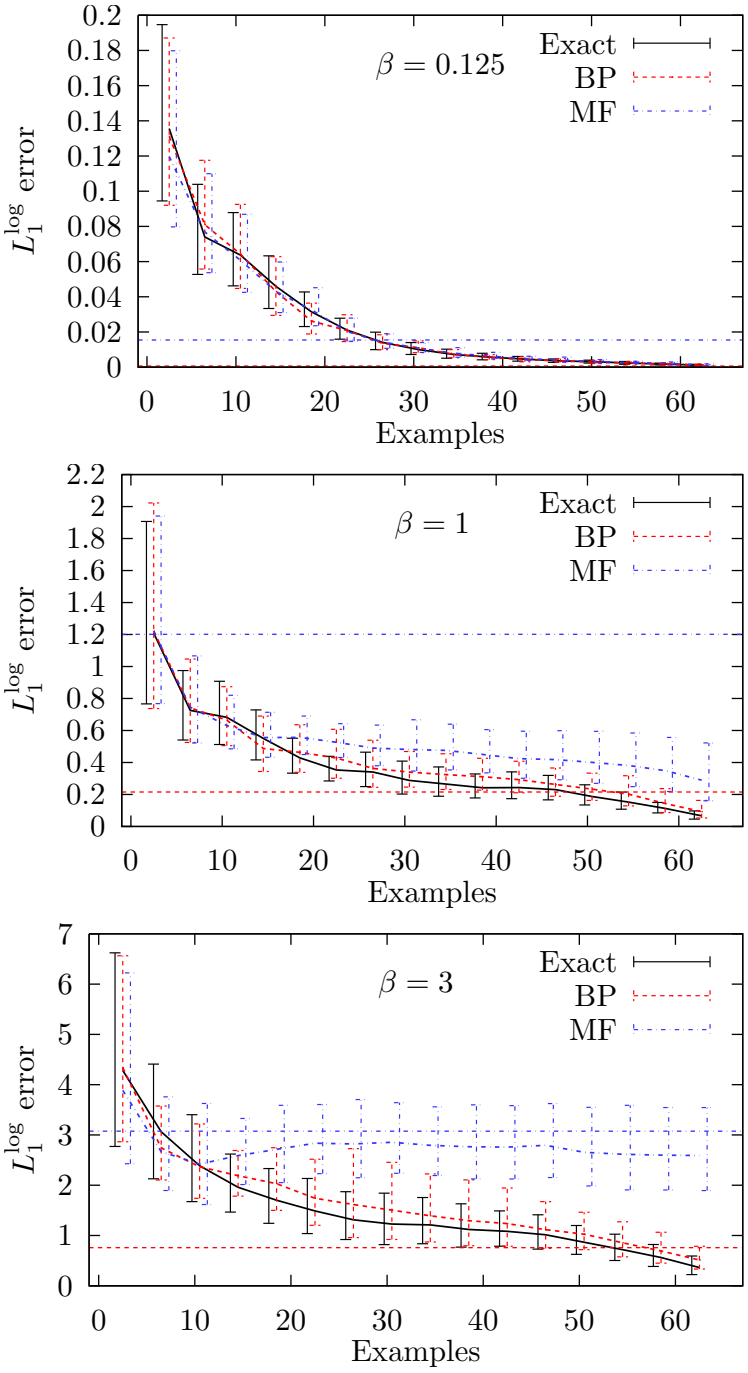


Figure 4: Learning curves for approximate teachers, for three values of β.

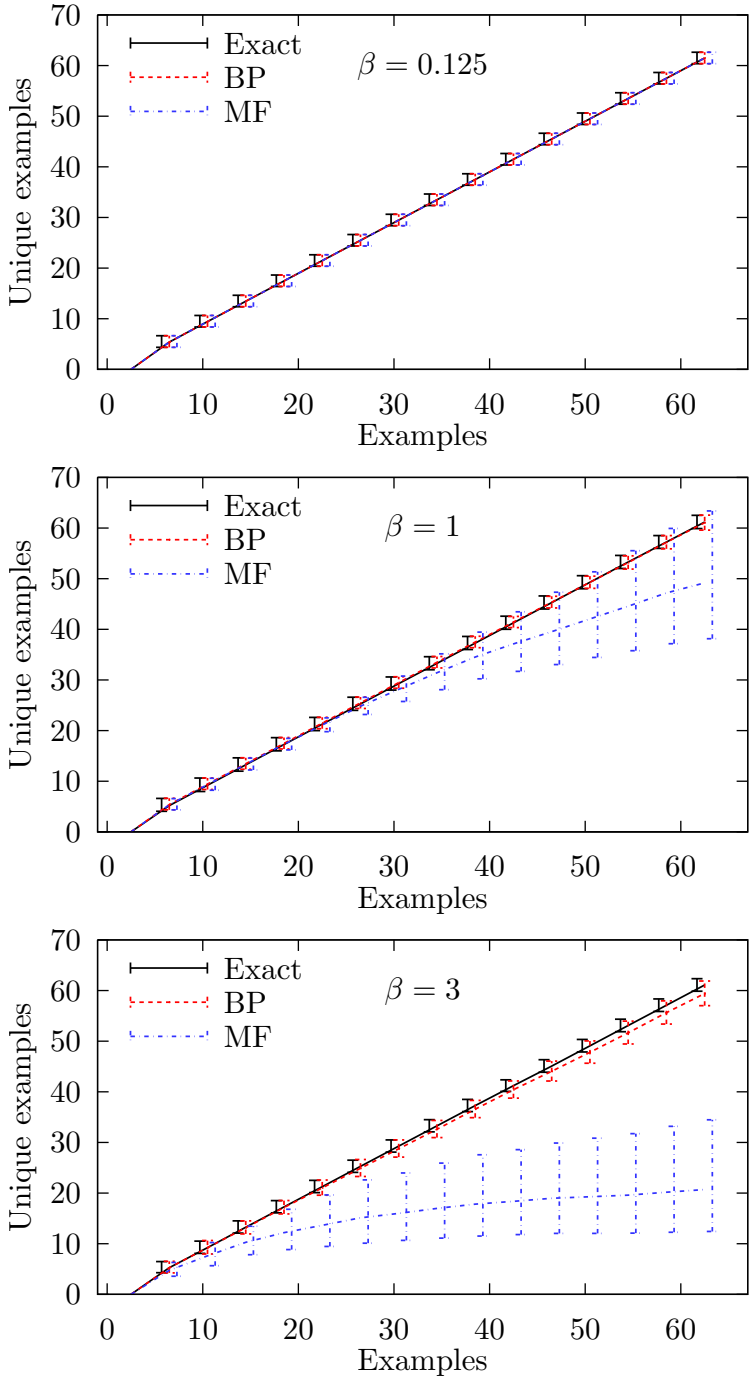


Figure 5: Unique examples as a function of examples, for approximate teachers and $\beta = 0.125$, 1 and 3.

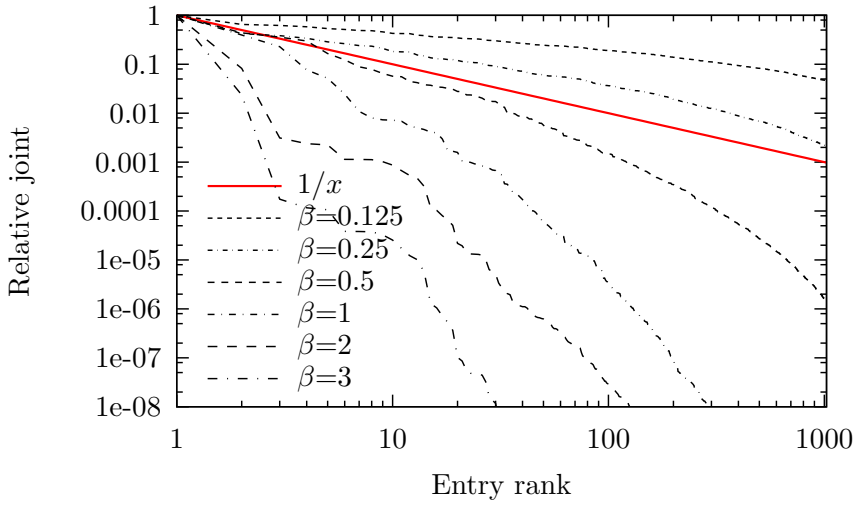


Figure 6: Zipf plots for entries in the joint distributions of six typical factor graphs, with reference line. In other words, ZP values for each state are sorted and plotted against their rank on a log-log scale; additionally we normalized the values so that the largest is 1. This is a useful way to visualize the difference in the models produced by different values of β. The steeper the slope of the line, the more probability mass is placed on a few dominant states, rather than being spread out across many states. A line with slope -1 has been included for reference. The slope -1 is a special case, which corresponds to a relationship $P(x_r) \propto \frac{1}{r}$ where r is the rank of the state x_r. Considering r to be continuous-valued, we note that $\int_0^1 \frac{1}{r^\alpha} \mathrm{d}r$ is ∞ if $\alpha \geq 1$, and $\int_1^\infty \frac{1}{r^\alpha} \mathrm{d}r$ is ∞ if $\alpha \leq 1$. Only when $\alpha = 1$ are both integrals infinite. This can therefore be compared to a case where probability mass is fairly divided between likely and unlikely states. From the plot, we see that the value $\beta = 0.5$ has a slope which is close to -1 over the first decade, but decreases thereafter.