Approximation Inference on Factor Graph with Bounds

Yewen Pu
yewenpu@mit.edu

Armando Solar-Lezama
asolar@csail.mit.edu

Swarat Chaudhuri
swarat@rice.edu

Abstract

We present a framework for approximation inference on a factor graph with bounds. The potential functions of the factors are bounded below and above by piecewise polynomials, and a variant of the variable elimination algorithm is executed on these bounds, producing a pair of lower/upper bounds of the queried marginal distribution. Our approach applies to potential functions whose minimum and maximum can be bounded over a domain, if efficiency is not a concern. For efficient inference, we describe a new class of potential functions, called *distance induced potentials*, which is both expressive and can be bounded easily with piecewise polynomials with our method.

1 Introduction

We present an approximate inference algorithm over a factor graph that produces a pair of lower and upper bounds for the queried distribution. The starting point for our approach is the general variable elimination algorithm on factor graphs [Zhang and Poole, 1994]. This algorithm works over the discrete domain, summing away the variables that are not present in the queried distribution. We work with the continuous analog of this algorithm, replacing summation with integration. The main challenge of the continuous domain is that it requires the computation of integrals and products, which may not have closed form solutions. Instead of computing the approximate integrals and products, our algorithm computes these operations on a pair of upper-bound and lower-bound. As a result, the algorithm produces not an approximation of the queried distribution, but sound upper and lower bounds of the true distribution, which can be used to give bounds on the probabilities of its events.

To achieve good lower and upper bounds, our algorithm works in two stages. In the first stage, the potential functions of the factor graph are bounded below and above by piecewise polynomials. We describe a class of potential functions that we call *distance induced potentials*, which is both expressive in representing different potential functions and can be bounded tightly with piece-wise polynomials. In the second stage, the lower and upper bounds are propagated in a variant version of the variable elimination algorithm, preserving the bounds. Upon obtaining an initial result, the algorithm can adaptively choose to refine the bound by forming new partitions for the piece-wise polynomials, achieving better bounds.

In contrast with sampling-based approaches such as MCMC [Andrieu *et al.*, 2003], our algorithm can provide strong guarantees on the probabilities of particular events, giving a sound lower and upper bound at each stage of its refinement toward a better precision. In contrast with variational inference methods [Beal, 2003], our approach is non-parametric and can provide guaranees without making assumptions about the forms of the underlying distribution. It also provides sound bounds which the variational method cannot guarantee.

2 Background Information

2.1 Factor Graph

A factor graph [Kschischang *et al.*, 1998] is a form of representation for a probability distribution. Formally, a factor graph is a tuple (X, F) where X is the set of variables and F is the set of factors over these variables. A variable $x \in X$ can take on values from a domain $dom(x) \subseteq \mathbb{R}$, a domain $dom(\{x_0, \ldots, x_k\})$ over a set of variables is the cartesian product of each of the variables domain. A factor $f \in F$ has a set of contributing variables denoted by $var(f)$. A factor also has an associated potential function over its variables, which maps an assignment of these variables to a non-negative real number. Intuitively, the potential function encodes the local compatibilities of these assignments. We will denote the potential function by f as well when the context makes it clear.

A factor graph defines a joint probability distribution p over X as a product of its potential functions:

$$p(x_1 \ldots x_n) = \frac{1}{Z} \prod_{f_j \in F} f_j(x_{j1} \ldots x_{jk}) \qquad (1)$$

As the potential functions are not necessarily distributions, the entire product is normalized in the end with Z to produce a valid distribution. Often, the normalization is avoided until one wishes to query for the probability of a specific event, and we will work mainly with un-normalized distribution in this paper.

A factor graph is usually represented as a bipartite graph where the nodes are the variables and the factors, and edges denote whether a variable contributes to a factor. We give an example of a factor graph:

Example Suppose there is a snail that crawls on the curb. The snail's starting position, s, has a prior, uniform distribution f_s between 0 and 2 meters, and its speed, v, has a prior, uniform distribution f_v between 1 and 3 meters per hour. Given we observed the snail at the 4.5 meters mark, t, after an hour, what is our updated beliefs of its speed? This problem can be represented as a factor graph in the figure below:

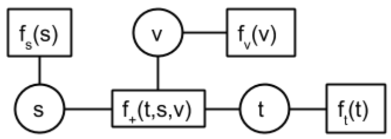

Figure 1: Visualizing the factor graph of the snail problem

Here f_+ is a potential function that imposes the constraint $t = v + s$. The observation of the snail's final position is encoded by adding the factor f_t, which restricts the variable t to take on the value 4.5. In general, conditional distributions can be modeled in a factor graph by adding factors.

A marginalized distribution over a subset $A = \{x_1, \ldots, x_r\} \subseteq X$ of the variables is given by marginalizing away the variables in $X \backslash A$

$$p(x_1 \ldots x_r) = \int_{\vec{x} \in X \backslash A} p(x_1 \ldots x_n) d\vec{x} \tag{2}$$

In our example, the queried marginal distribution can be computed by:

$$p(v) = \int_s \int_t f_s(s) f_v(v) f_t(t) f_+(t, s, v) dt\, ds$$

In this paper, we focus on the following question: given a factor graph (X, F) in the continuous domain, and a subset of the variables $A \subseteq X$, produce a bound on the marginal distribution such that, for any event E in the marginal, one can derive a lower and upper bound on the probability of that event.

2.2 Variable Elimination

One way of solving for the marginal distribution is using the variable elimination (VE) algorithm[Zhang and Poole, 1994]. Given a factor graph (X, F) and a subset of the variables $A \subseteq X$, the VE algorithm successively pick variables $x' \in X \backslash A$, and marginalize it away until only the variables that remain are in A. The algorithm is given below, in its recursive formulation:

```
VE$(X, F, A)$ :
  if$(X \backslash A) = \emptyset$ :
    $\prod_{f \in F} f$
  else :
    pick $x' \in (X \backslash A)$
    $F_{x'} = \{f \in F \mid x' \in var(f)\}$
    $f_{\pi x'} = \prod_{f \in F_{x'}} f$
    $f_{\backslash x'} = \int_{x'} f_{\pi x'} dx'$
    VE$(X \backslash \{x'\}, (F \backslash F_{x'}) \bigcup \{f_{\backslash x'}\}, A)$
```

Notice this algorithm creates a two new factors: $f_{\pi x'}$ for multiplication and $f_{\backslash x'}$ for integration at each step of its elimination, and the factor produced at the base case is the un-normalized marginal distribution in our query. Below is a figure showing all the factors generated by running the VE algorithm on our example problem. Of the original factors f_s, f_+, f_t, f_v, the algorithm created the new factors $f_1, \ldots, f_5$.

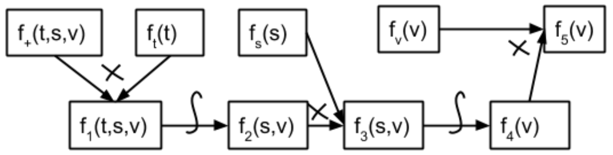

Figure 2: The algorithm first eliminates t, then s. The output is f_5, which is the posterior distribution on the speed v

The main difficulty in the VE algorithm lies in the integration operation $f_{\backslash x'} = \int_{x'} f_{\pi x'} dx'$. When the problem domain is discrete, the integration reduces to summation, and VE provides an exact inference. In the continuous domain, integration has no closed form in general. Therefore, in order for our algorithm to succeed, it must address these challenges:

1. Uses lower/upper bounds that has closed-form multiplications and integrations.
2. Identify a family of expressive potential functions that can be bounded by functions in 1.
3. Preserve the bounds with the operators of the VE algorithm: multiplication and integration.

3 Algorithm Overview

We now give an overview of our algorithm, and explain on a high level how does our algorithm addresses the challenges posed in the previous section. The algorithm uses piece-wise polynomials as lower and upper bounds for the potential functions, as piece-wise polynomials have closed-form multiplication and integrations. We will show that, as long as efficiency is not a concern, if a potential function f can have its maximum and minimum bounded within a domain dom, our algorithm can derive sound lower-bound and upper-bound for it. However, if efficiency is a concern, we describe a family of potential functions which we call *distance induced potentials*, which can be easily bounded. Finally, we briefly outline that after the potentials of the factor graph are bounded, how to preserve the bounds through a variant of the variable elimination algorithm.

4 Bounding the Potential Function

For a potential function f, we define the bounds f_{min} and f_{max} such that:

$$\forall x \in dom(f), f_{min}(x) \leq f(x) \leq f_{max}(x)$$

Given this abstraction, if the potential function f happen to be our queried un-normalized distribution, we can bound the probability of an event E by defining these two functions:

$$f_{E_{min}}(x) = f_{min}(x)\ if\ x \in E, f_{max}(x)\ otherwise$$
$$f_{E_{max}}(x) = f_{max}(x)\ if\ x \in E, f_{min}(x)\ otherwise$$

One can think of these functions as the most conservative distributions for bounding the probability of E. For instance, the function $f_{E_{min}}$ is used to compute the lower bound, and it assumes the smallest values for $x \in E$ and the largest values for $x \notin E$. The bound is given as:

$$\frac{\int_{x \in E} f_{E_{min}}(x)dx}{\int_x f_{E_{min}}(x)dx} < Pr(E) < \frac{\int_{x \in E} f_{E_{max}}(x)dx}{\int_x f_{E_{max}}(x)dx}$$

A nice property to note is that given $\bar{f}$, and two events E_1 and E_2 such that $E_1 \subseteq E_2$, $lowerbound(Pr(E_1)) \leq lowerbound(Pr(E_2))$ and $upperbound(Pr(E_1)) \leq upperbound(Pr(E_2))$ using our abstraction.

4.1 Piecewise Polynomials

In our algorithm, the piece-wise polynomials f_min and f_max are implicitly represented as a collection of patches. A patch pat is a tuple of (dom, p_l, p_u) where dom is the domain of the patch, and p_l and p_u are lower and upper bound polynomials(not piece-wise) for the domain, respectively. Then, the piece-wise polynomials f_min and f_max can be thought of as a collection of patches, whose domains form a partition on $dom(f)$. The task of obtaining a piecewise polynomial bound for a factor is thus reduced to being able to compute p_l and p_u for an arbitrary domain dom.

4.2 Obtaining a Patch

One intuitive way of obtaining the bounds p_l and p_u of a potential function f over a domain dom is to compute an approximate function p over the domain dom, then find $c1$ and $c2$ such that: $c1 > max_{x \in d} p(x) - f(x)$ and $c2 > max_{x \in d} f(x) - p(x)$. By shifting the approximation p by $c1$ and $c2$, we obtain our desired bounds as follows: $p_l = p - c1$ and $p_u = p + c2$. Note that since p is from a family of polynomials, shifting by a constant c of a polynomial results in a polynomial as well.

To obtain an approximate function p, we use techniques developed in the chebfun package of matlab [Townsend and Trefethen, 2013], which works by iteratively project the function f onto an outer product of Chebychev polynomials. One drawback of using this method of approximation is, while it is the state of the arts algorithm for multi-variate polynomial approximation for continuous functions in practice, for a fixed degree (in our case 2), there is no bound readily available that will help us computing the shift $c1$ (the case for $c2$ is analogous). To obtain $c1$, we use a refinement algorithm that iteratively divides the initial domain dom into smaller and smaller sub-domains, and bound the maximum differences between f and p by subtracting the minimum of f from the maximum of p over the sub-domaini d. The intuition is that, if f is well approximated by p, as both function are bounded on smaller and smaller sub-domains individually, their differences will also become smaller. It is then safe to let $c1$ to be the largest difference out of all the sub-domains.

```
bound(f, p, d)
  Q = priorityQueue()
  push!(Q, (d, max(d)(p)-min(d)(f)))
  do
    cur_d, cur_err = pop!(Q)
    diameter = diameter(cur_d)
    d1, d2 = split(cur_d)
    push!(Q, (d1, max(d1)(p)-min(d1)(f)))
    push!(Q, (d2, max(d2)(p)-min(d2)(f)))
  while (diameter < epsilon)
  return cur_err
```

By organizing the sub-domains into queues and splitting the domain with the largest difference, it is more efficient than if the initial domain was uniformly splitted. Also notice that this algorithm decomposes the problem of bounding $max_d(p - f)$ where both functions p and f are considered together inside the max into separate concerns of finding $max_d(p)$ and $max_d(f)$. This separation allows us to consider bounding each class of functions f and p independently from each other.

To bound the polynomial p over a sub-domain d, one may use interval arithmetic [Tucker, 2011]. The main idea of interval arithmetic is generalizing the usual arithmetic operators such as $+$ and $*$ to work on intervals. For instance, $[1, 2] + [4, 6] = [5, 8]$ and $[-4, 5] * [-4, 5] = [-20, 25]$. Note that even though interval arithmetic is not precise, for a polynomial of small degree (we use 2) and a small enough sub-domain d, interval arithmetic gives reasonable bounds.

The same cannot be said for the potential function f. While we certainly can bound f over a small sub-domain d with interval arithmetic, chances are for a more expressive potential function with many basic arithmetic operations, the bound given by interval arithmetic will not be tight. In the next section, we describe a class of expressive potential functions and give a closed-form expression for computing tight bounds.

5 Distance Induced Potentials

In this section we define a class of potential functions which we call *distance induced potentials*. Potential functions from this class are relaxations of hard constraints corresponding to common primitives, such as addition, less than, or equalities, of a typical programming language. Given a potential function f of this class and a domain d, we will show it is easy to derive bounds for $min_d(f)$ and $max_d(f)$.

To facilitate our discussion, recall in formulating our example problem as a factor graph, we defined a factor $f_+(t,s,v)$ that imposes a constraint $t = s+v$. It may be tempting to give a boolean potential function:

$$Plus(t,s,v) = \begin{cases} 1 & if \ \ (t = s+v) \\ 0 & \text{otherwise} \end{cases} \tag{3}$$

However, a boolean potential as the one defined above has measure 0 in the domain which the potential is defined over, which makes integration over this potential function ill-defined. To address this issue, we relax the hard constraint by measuring a "distance function" to the surface where the constraint is satisfied, arriving at the class of distance induced potential functions.

5.1 Satisfying Surface and Distance Function

A satisfying surface of a constraint c, S_c, is a set of points in the domain such that the constraint is satisfied.

$$S_c = \{x | c(x) = true\} \tag{4}$$

For instance, in the case of a plus constraint, $t = s+v$, its satisfying surface is:

$$S_{t=s+v} = \{(t,s,v) | t = s+v\} \tag{5}$$

Therefore, the point $(5,2,3)$ will be inside this satisfying surface, while the point $(2,2,2)$ will not.

Given a satisfying surface S_c, and an arbitrary point x, we define the distance function $d_c(x)$ as the minimum euclidean distance from x to any point on the surface S_c.

$$d_c(x) = min_y\{||x-y|| \mid y \in S_c\} \tag{6}$$

Note the distance function has the property that if an arbitrary point is on the satisfying surface, its distance is 0, and as the point moves further and further away from the satisfying surface, its distance decreases accordingly. Also note that since it is a distance function, it must be continuous, which makes it possible to be approximated by polynomials.

Clearly, enumerating over all the points $y \in S_c$ is intractable, so we seek to derive an analytical representation of $d_c(x)$. This is done on a case-by-case basis as each constraint c defines a different satisfying surface S_c. However, we find that in practice most satisfying surfaces are a finite union of simpler surfaces such as lines, half-planes, or points. Therefore, we can compute the distance from a point x to each of the simpler surfaces, and take a minimum of all the distances to obtain a valid distance function.

Usually, the distance from a point x to one of the simpler surfaces S can be computed by using the lagrange multipliers, where the goal is to find a point y on the surface S that is closest to the point x. The constraint is the surface S itself and the objective function is the euclidean distance squared between x and y.

For our particular constraint of plus, let the triple (t',s',v') denote a point on the satisfying surface, and let the triple (t,s,v) denote an arbitrary point in space. We can set up the lagrange multiplier as follows:

$$\begin{aligned} &constraint: \\ &s' + v' - t' \\ &objective: \\ &(t-t')^2 + (s-s')^2 + (v-v')^2 \end{aligned} \tag{7}$$

Solving this system gives the following solution for the closest point (t',s',v') on the satisfying surface to the arbitrary point:

$$\begin{cases} t' & = \frac{1}{3}(s+v+2t) \\ s' & = s - (\frac{1}{3}s + \frac{1}{3}v - \frac{1}{3}t) \\ v' & = v - (\frac{1}{3}s + \frac{1}{3}v - \frac{1}{3}t) \end{cases} \tag{8}$$

Substituting the solution into our objective function, and taking the squared root we obtain the euclidean distance function for the plus constraint:

$$\Big\{ d_+(t,s,v) \quad = \frac{\sqrt{3}}{3}(s+v-t) \tag{9}$$

Below is a table of some of constraints along with their satisfying surfaces which we have managed to derive a symbolic distance function. The distance functions are omitted for brevity, but the satisfying surfaces are given as a union of simpler surfaces which we used to derive the distance functions.

constraint	satisfying surface
$y = not(x)$	$\{(0,1),(1,0)\}$
$y = x_1 \wedge x_2$	$\{(0,0,0),(0,1,0),(0,0,1),(1,1,1)\}$
$y = x_1 \vee x_2$	$\{(0,0,0),(1,1,0),(1,0,1),(1,1,1)\}$
$y = -x$	$\{(y,x) \| y = -x\}$
$y = cx$	$\{(y,x) \| y = cx\}$
$y = x_1 + x_2$	$\{(y,x_1,x_2) \| y = x_1 + x_2\}$
$b = (x_1 < x_2)$	$\{(0,x_1,x_2) \| x_1 \geq x_2\}$ $\cup\{(1,x_1,x_2) \| x_1 \leq x_2\}$
$y = ite(b,x_1,x_2)$	$\{(y,b,x_1,x_2) \| b = 1 \wedge y = x_1\}$ $\cup\{(y,b,x_1,x_2) \| b = 0 \wedge y = x_2\}$

Note we use the real number 0 and 1 to encode the boolean values false and true respectively. Although we derived the distance function for multiplying by a constant c, we were not yet able to derive a symbolic distance function for the general multiplication $y = x_1 * x_2$. This is due to the satisfying surface of multiplication generates a non-linear constraint for the lagrange multiplier, which makes its solutions difficult to compute. The ite operator is the if-then-else operator, which assigns the variable y with the value of x_1 if b takes the value of 1, and assigns variable y with the value of x_2 if b takes the value of 0.

5.2 Distance to Potential Functions

The distance function is converted to a potential function by fitting it with a gaussian distribution centered at 0:

$$f_c(x) = \mathcal{N}(d_c(x)|\mu = 0, \sigma) \tag{10}$$

By mapping the distance function for a constraint d_c with a gaussian, we have the guarantee that as the constraint is satisfied, the value of the potential function is at its highest, and when the constraint is unsatisfied, instead of setting the potential to 0, we measure ”how far” has the point becomes unsatisfied, and gradually decays the potential function depending on the distance. One can use the parameter σ to quantify the hardness of the constraint, where a smaller value will cause the potential function to decay more quickly.

Note we get the uni-variate gaussian distribution for free, by constructing a distance function from an arbitrary constant c, we are able to center the gaussian distribution at c. Also, if we define a distance function to an interval $[a, b]$ as:

$$d(x) = \begin{cases} 0 & \text{if } \ a < x < b \\ min(|x-a|, |x-b|) & \text{otherwise} \end{cases} \tag{11}$$

We obtain an approximation of the uni-variate uniform distribution after applying the gaussian transformation to the distance function, which we use in our example problem.

5.3 Bounding a distance induced potential

We now demonstrate how to obtain a tight bound for a distance induced potential function. Recall that the potential function of a constraint factor is given by first obtaining the distance function d_c of the constraint c, then applying a gaussian function on top of the distance function to obtain the potential function. Therefore, if we are able to bound the distance function over a domain d, we can obtain the bound for the potential function, as the gaussian function is monotonically decreasing for a non-zero input. Formally:

$$\begin{aligned} min_d(f_c) &= \mathcal{N}(max_{x \in d}(d_c(x))|\mu = 0, \sigma) \\ max_d(f_c) &= \mathcal{N}(min_{x \in d}(d_c(x))|\mu = 0, \sigma) \end{aligned} \tag{12}$$

The question we must answer is: Given a distance function d_c and a domain d, can we bound the smallest and largest distances attainble over all the points in the domain?

The fact that the distance function measures euclidean distance will play a key role: We can draw a bounding sphere around the domain d, and since the distance function is measured in the euclidean distance, the smallest and largest distance value attainable over the domain d is bounded by the smallest and largest value attainble on the bounding sphere:

$$\begin{aligned} &\text{Bound } \ (f, d): \\ &\qquad o = \text{center}(d) \\ &\qquad r = \frac{\text{diameter}(d)}{2} \\ &\qquad d_{furthest} = d_c(o) + r \\ &\qquad d_{closest} = \max(0, d_c(o) - r) \end{aligned}$$

Here center gives the center of the domain d, diameter measures the largest diagonals of d. Below is a figure illustrating the bound for a distance function over a domain d.

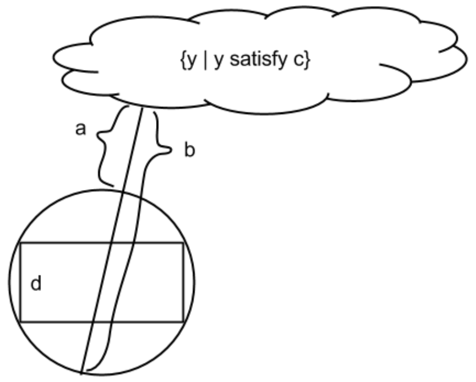

Figure 3: The domain d, a rectangle, has its distance bounded by a and b, deriving from the bounding sphere

In conclusion, we have shown that the class of potential functions induced by distance functions is expressive enough to capture many of the common programming language primitives, and by using a euclidean distance, one can symbolically compute bounds for the potential functions over a domain d. This allows all the distance induced potential functions to be bound tightly by piece-wise polynomials.

6 Operators on the Bounds

Having obtained lower and upper bounds for the potential functions in the factor graph, the algorithm executes the variable elimination algorithm on these bounds to produce bounds for the marginal distribution. The two operators of interest are multiplication and integration. In this section we briefly discuss how to compute these operations on piecewise polynomials in a way that preserves the bound.

In the start of the algorithm, each factor contains exactly 1 patch, which covers its entire domain. As the algorithm progresses, it refines the patches from each of the factors by splitting the domain of the patch in half (along its longest axis), creating two smaller patches. When a patch is split, instead of discarding the splitted patch, the factor retains the splitted patch in a patch set $PATCH$, which is a set containing all the patches ever used to define the piece-wise polynomial bounds of the factor.

Multiplication If the patch pat belongs to a factor f that is the multiplication of two other factors f_1 and f_2, we query the patch set of f_1 for the patch with the smallest domain that contains the the domain of pat, call this patch pat_1. Similarly, we query the patch set of f_2 for the patch with the smallest domain that covers pat, call this patch pat_2. The lower and upper polynomials of pat is thus the product of the lower and upper polynomials of pat_1 and pat_2. The intuition behind this definition is that pat_1 and pat_2 contain the most precise bounds that can be used to compute the bound for pat, without having to splitting pat itself due to multiple overlapping domains.

Integration If the patch pat belongs to a factor f that is the integration of a factor f_1 against a variable y, we query the patch set of f_1 to obtain not one but a set of patches with the following properties:

- the set of patches covers the entire length of $dom(y)$
- the intersections of the projections of the patches in the y direction results in the smallest domain that contains the domain of pat

By selecting a set of patches that meets these two properties, we obtain the most precise bounds for pat by summing over the definite integrals of these patches. Since the intersection of these patches projection contains the domain of pat, there is no need to split pat further into smaller pieces.

In the algorithm implementation, we use a binary space partition instead of a patch set for efficiency.

6.1 Heuristics for partitioning

We now describe a simple heuristic for picking which patch we should split. The intuition is that a patch that causes the most error in the marginalized distribution should be the patch that is split. We define the imprecision of a patch $imprecision(pat)$ informally as: Set all other patch's lower bound p_l to be equal to its upper bound p_u, thereby making every other patch precise. Then, do the VE algorithm where the only cause of imprecision is due to the difference between the lower bound p_l and upper bound p_u of pat. Once the imprecision for all the patches are computed, the patch with the greatest imprecision is split in half.

7 Experiments

We ran the algorithm on our example problem, asking for the posterior distribution on the snail's speed. The total number of refinements by splitting the patches was 500.

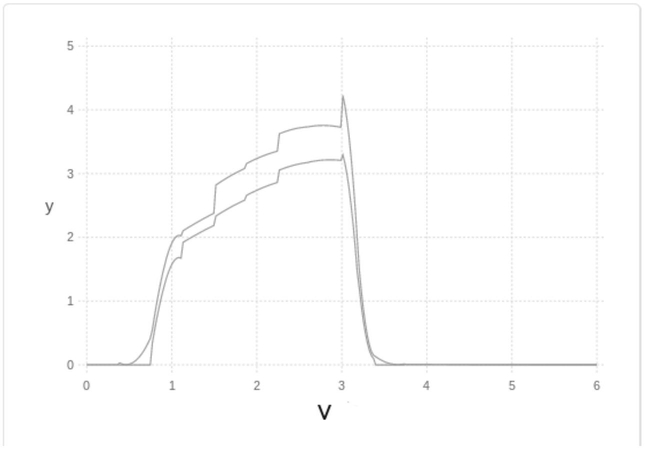

Figure 4: The resulting lower and upper bound of the snail's speed

As we can see, the lower bound and the upper bound have converged almost everywhere, giving a strong guarantee of the true distribution which lies between the lower bound and the upper bound. The distribution itself coincides with our intuition: If a snail is traveling slower, it is unlikely to reach the landmark of 4.5 within an hour, thus we observe a skew toward faster speed by the snail. Profiling of the code shows more than 90 percent of the time is spent in the heuristic function, indicating that obtaining bounds and running the VE algorithm is not yet the bottleneck.

8 Related Works

The most similar works in the literature seems to be dynamic discretization [Kozlov and Koller, 1997], which uses piece-wise constant functions to approximate the potential functions, and propagate the piece-wise constant functions through the variable elimination algorithm. [Sanner and Abbasnejad, 2012] restricts the domain to piece-wise polynomials. Our approach is different from both in that in theory, it only assumes the potential functions in the factor graph can be bounded, and provide instead of an approximation, bounds of the queried distribution. We also describe a class of distance induced potential functions which our method excels at providing bound for.

9 Conclusion and Future Works

In summary, this paper makes the following contributions:

- proposed an algorithm that computes lower and upper bound for the queried distribution
- outlined a strategy to bound the potential functions with piece-wise polynomials
- described a family of potential functions that is expressive and easily bounded
- explained how the bounds can be propagated on a variant of the VE algorithm

For future work, the author would like to improve the refinement heuristics in order to perform inferences on more complicated problems.

References

[Andrieu *et al.*, 2003] Christophe Andrieu, Nando de Freitas, Arnaud Doucet, and Michael I. Jordan. An introduction to MCMC for machine learning. *Machine Learning*, 50(1-2):5–43, 2003.

[Beal, 2003] Matthew J. Beal. *Variational Algorithms for Approximate Bayesian Inference*. PhD thesis, Gatsby Computational Neuroscience Unit, University College London, 2003.

[Kozlov and Koller, 1997] Alexander Kozlov and Daphne Koller. Nonuniform dynamic discretization in hybrid networks. In *In Proc. UAI*, pages 314–325. Morgan Kaufmann, 1997.

[Kschischang *et al.*, 1998] Frank R. Kschischang, Brendan J. Frey, and Hans-Andrea Loeliger. Factor graphs and the sum-product algorithm. *IEEE TRANSACTIONS ON INFORMATION THEORY*, 47:498–519, 1998.

[Sanner and Abbasnejad, 2012] Scott Sanner and Ehsan Abbasnejad. Symbolic variable elimination for discrete and continuous graphical models, 2012.

[Townsend and Trefethen, 2013] Alex Townsend and Lloyd N. Trefethen. An extension of chebfun to two dimensions. *SIAM J. Scientific Computing*, 35(6), 2013.

[Tucker, 2011] Warwick Tucker. *Validated numerics : A Short Introduction to Rigorous Computations*. Princeton University Press, 2011.

[Zhang and Poole, 1994] N.L. Zhang and D. Poole. A simple approach to bayesian network computation. In *proceedings of the 10th Canadian Conference on Artificial Intelligence*, pages 16–22, Banff, Alberta, Canada, 1994.