Predicate Exchange: Inference with Declarative Knowledge

Zenna Tavares [1] **Javier Burroni** [2] **Edgar Minasyan** [3] **Armando Solar Lezama** [1] **Rajesh Ranganath** [4]

Abstract

Programming languages allow us to express complex predicates, but existing inference methods are unable to condition probabilistic models on most of them. To support a broader class of predicates, we develop an inference procedure called *predicate exchange*, which softens predicates. A soft predicate quantifies the extent to which values of model variables are consistent with its hard counterpart. We substitute the likelihood term in the Bayesian posterior with a soft predicate, and develop a variant of replica exchange MCMC to draw posterior samples. We implement predicate exchange as a language agnostic tool which performs a nonstandard execution of a probabilistic program. We demonstrate the approach on sequence models of health and inverse rendering.

1. Introduction

Conditioning in Bayesian inference incorporates observed data into a model. In a broader sense, conditioning revises a model such that a predicate of uncertain truth becomes a fact. Conventionally, this predicate is the equality of observable variables to data. Predicates outside of this class have received significantly less attention, partly because it makes the inference problem significantly more challenging, and partly because conditioning on data accommodates many applications. Nevertheless, there are many more predicates outside this class than inside; our inability to condition on them is a major limitation.

The ability to condition on a broader class of predicates would enable us to incorporate more kinds of declarative domain knowledge into generative models. For example, probabilistic variants of inverse rendering (Marschner & Greenberg, 1998; Kulkarni et al., 2015) require a prior distribution over three dimensional scenes. Some forms of knowledge, such as the fact that rigid bodies do not intersect, are easier to express declaratively as predicates, than constructively in a generative model. Conditioning on predicates allows us to express what should be true without the burden of specifying how.

Predicates can also represent observations that restrict variables to sets rather than single values. For example, a medical practitioner may observe that a patient is hypoglycemic, i.e., that their glucose levels have fallen below a critical value. Given a model over time series of glucose levels (Levine et al., 2017; Murata et al., 2004), this observation can be realized as a predicate that maps the series to 1 if it falls below the threshold. Neither hypoglycemia nor any of the infinite number of predicates that can be expressed, need to exist in the generative model apriori.

Probability theory treats conditioning on predicates and concrete observations uniformly, but sampling from models conditioned on most predicates is challenging due to the lack of a tractable likelihood function. The likelihood function quantifies the extent to which values of latent variables are consistent with observations, and is deemed intractable if it is normalized by intractable integrals or summations. This can occur, for example, if we condition random variables that are deterministic transformations of other random variables (e.g., the occurrence of hypoglycemia in the example above, or the mean of a collection of variables). Alternatively, if the model is generative, i.e. specified as a stochastic simulation, the likelihood is not explicitly available even when the condition is a conventional observation. The numerous effective likelihood-based sampling (Andrieu et al., 2003) and variational (Jordan et al., 1999; Ranganath et al., 2014) methods are inapplicable as a result.

In this paper we present *predicate exchange*: a likelihood-free method to sample from distributions conditioned on predicates from a broad class. It is composed of two parts:

1. **Predicate Relaxation** constructs soft predicates which return values in a continuous Boolean algebra: the unit interval $[0, 1]$ with continuous logical connectives $\tilde{\wedge}$. $\tilde{\vee}$ and $\tilde{\neg}$. Softened predicates approximate their hard counterparts.

2. **Replica Exchange** is a Markov Chain Monte Carlo

[1]MIT, USA [2]College of Information and Computer Science, University of Massachusetts, Amherst, USA. [3]Princeton University, USA [4]NYU, USA. Correspondence to: Zenna Tavares <zenna@mit.edu>.

Proceedings of the 36^{th} International Conference on Machine Learning, Long Beach, California, PMLR 97, 2019. Copyright 2019 by the author(s).

method that simulates Markov chains at different temperatures. Predicate relaxation is parameterized by a temperature which controls the amount of approximation introduced. We use replica exchange to draw samples from the unrelaxed model.

By returning a value in $[0, 1]$ instead of $\{0, 1\}$, a soft predicate quantifies the degree to which values of variables are consistent with the hard predicate. We concretize this concept in terms of distance: a realization of the model is almost consistent with a predicate if there is another realization that is both consistent with the predicate and close-by with respect to a metric.

Hard predicates exist in a Boolean algebra; they can be conjoined, disjoined and negated. This enables predicates to represent knowledge with complex Boolean structure. Continuing the previous example, we may know that a person is *not* hypoglycemic, or that they are hypoglycemic *or* hyperglycemic, or *neither*. To be able to relax complex predicates, we define a soft Boolean algebra with continuous counterparts to equality, inequalities and logical connectives.

To perform inference we replace the likelihood term in the Bayesian posterior with a soft predicate. This yields an *approximate posterior* which we sample from using Markov Chain Monte Carlo. However, relaxed predicates can still induce complex, multimodal posteriors. Increasing the temperature smooths the approximate posterior, but also causes it to diverge from the true posterior. We use replica exchange to mitigate the trade-off. Replica exchange simulates high temperature chains – which explore vasts regions of the sample space – in parallel with low temperature chains which sample accurately from localized regions. We augment replica exchange with an accept-reject phase, which allows us to use approximate posteriors to sample from the true posterior when the predicate is of non-zero measure.

Predicate exchange addresses a shortcoming of probabilistic programming languages, which have vastly expanded the class of probabilistic models that can be expressed, but still restrict the kinds of predicates that can be conditioned on to those which result in a tractable likelihood. In a similar vein to (Wingate et al., 2011) we provide a light-weight implementation that modulates the execution of a stochastic simulation based model to perform inference. This means predicate exchange is easily incorporated into existing probabilistic languages. For a concrete implementation, we build predicate exchange into the OMEGA probabilistic programming language[1] (Tavares et al., 2019),

In summary, we:

1. Formalize the desiderata for predicate relaxation (Section 2.1) and present relaxations of numerical and logical primitive functions.
2. Implement predicate exchange as nonstandard execution of a simulation based model (Section 3).
3. Evaluate our approach on examples including inverse rendering and glycemic forecasting (Section 4).

[1] OMEGA is available at http://github.com/zenna/Omega.jl

2. Predicate Exchange

Given a model (a collection of random variables) $\mathbf{X} = (X_1, X_2, \ldots, X_n)$ and a predicate ℓ which maps a model realization $\mathbf{x} = (x_1, x_2, \ldots, x_n)$ to 0 or 1, predicate exchange samples from the posterior distribution of $\mathbf{X}$ conditioned on ℓ through two steps:

1. **Predicate Relaxation** constructs a soft predicate $\tilde{\ell}$ from ℓ. $\tilde{\ell}$ maps a model realization $\mathbf{x}$ in a realization space $\mathcal{X}$ to a value in a continuous Boolean algebra: the unit interval $[0, 1]$ with continuous logical connectives $\tilde{\wedge}$, $\tilde{\vee}$ and $\tilde{\neg}$. $\tilde{\ell}$ is 1 iff ℓ is 1, but otherwise takes nonzero values denoting the degree to which ℓ is satisfied.
2. **Replica Exchange** is a Markov Chain Monte Carlo procedure that simulates several replicas of the model conditioned on $\tilde{\ell}$ at different temperatures, ultimately in order to condition on ℓ.

In this section we motivate and formalize the relaxation of predicates expressed as compositions of simple functions. In addition, the latter part of this section defines what it means to condition on a soft predicate. Together, these form the foundation of the relaxation and conditional sampling from probabilistic programs in Section 3.

2.1. Predicate Relaxation

Our objective is to construct a soft predicate $\tilde{\ell}$ that introduces approximations into ℓ to make inference more tractable. We refer to $\tilde{\ell}$ as a *relaxation* of ℓ. Informally, these approximations mean that while conditioning on ℓ eliminates a set of values, conditioning on $\tilde{\ell}$ makes them less likely.

A relaxation $\tilde{\ell}$ has a temperature parameter α that controls the fidelity of the approximation. There are three desiderata which govern this approximation. In particular, $\tilde{\ell}$ should (i) converge to ℓ as $\alpha \to 0$, (ii) converge to 1 as $\alpha \to \infty$, and (iii) be consistent with ℓ on 1, i.e., $\ell(\mathbf{x}) = 1$ iff $\tilde{\ell}(\mathbf{x}) = 1$ at all temperatures. Formally:

Definition 1. $\tilde{\ell} : \mathcal{X} \to [0, 1]$ parameterized by $\alpha \in [0, \infty)$ is a relaxation of $\ell : \mathcal{X} \to \{0, 1\}$ if for all $\mathbf{x} \in \mathcal{X}$:

(i) $\lim_{\alpha \to 0} \tilde{\ell}(\mathbf{x}; \alpha) = \ell(\mathbf{x})$.

(ii) $\lim_{\alpha \to \infty} \tilde{\ell}(\mathbf{x}; \alpha) = 1$.

(iii) for all $\alpha < \infty$, $\tilde{\ell}(\mathbf{x}; \alpha) = 1$ iff $\ell(\mathbf{x}) = 1$.

Axioms (i) and (ii) allow us to vary the fidelity to which $\tilde{\ell}$ approximates ℓ, and axiom (iii) prevents $\tilde{\ell}$ from deviating from ℓ on values which satisfy ℓ. These axioms allow us to construct a sampling procedure that (i) samples accurately from local regions, but also (ii) explores large parts of the realization space, and (iii) in non measure-zero cases can sample from the true posterior without approximation.

Similarity Based Satisfiability There is a straightforward construction of $\tilde{\ell}$ that satisfies the relaxation axioms. Let ρ be a metric on $\mathcal{X}$, k_α be a relaxation kernel (described below) which maps distances in $\mathbb{R}$ to similarities in $[0, 1]$, and $A = \{\mathbf{x} \mid \ell(\mathbf{x}) = 1\}$ be the satisfying set. The predicate $\tilde{\ell}_{\text{inf}}(\mathbf{x})$ is a relaxation of ℓ:

$$\tilde{\ell}_{\text{inf}}(\mathbf{x}) = k_\alpha(\rho(\mathbf{x}, A)) \tag{1}$$

The distance $\rho(\mathbf{x}, A) = \inf\{\rho(\mathbf{x}, a) \mid a \in A\}$ is the smallest distance between $\mathbf{x}$ and any element of A.

Relaxation Kernels A relaxation kernel k_α maps a distance in $\mathbb{R}$ to the unit interval, and is parameterized by temperature α. One kernel, which we restrict our attention to, is the squared exponential kernel:

$$k_\alpha(r) = \exp\left(-\frac{r^2}{\alpha}\right) \tag{2}$$

Soft Primitives Unfortunately, $\tilde{\ell}_{\text{inf}}$ is in general incomputable since the distance $\rho(x, A)$ from a point to the satisfying set is incomputable. Instead, to relax ℓ we relax the functions it is composed of, which boil down to a set of numerical and logical primitives. For instance the predicate $(x > y) \vee \neg(x^2 = 2)$ is relaxed by $(x \tilde{>} y) \tilde{\vee} \tilde{\neg}(x^2 \tilde{=} 2)$, where $\tilde{>}$, $\tilde{<}$, $\tilde{=}$, $\tilde{\vee}$, $\tilde{\wedge}$ and $\tilde{\neg}$ are as follows.

Soft equality is straightforward: $x \tilde{=} y$ is defined as $k_\alpha(\rho(x, y))$. A soft inequality such as $x \tilde{>} y$ is a function of the amount by which x must be increased (or y decreased) until $x > y$ is true. This is the distance between x and the interval $[y, \infty]$, where the distance between a point and any interval $[a, b]$ is the smallest distance between x and any element in $[a, b]$, and therefore 0 if $x \in [a, b]$:

$$\rho(x, [a, b]) = \begin{cases} a - x & \text{if } x < a \\ x - b & \text{if } x > b \\ 0 & \text{otherwise} \end{cases} \tag{3}$$

Soft conjunction $\tilde{\wedge}$ and disjunction $\tilde{\vee}$ take the $\min$ and $\max$ respectively, which is standard. Soft negation, on the other hand, introduces complications. In most continuous logics, the negation of $a \in [0, 1]$ is $1 - a$. However, as shown in Figure 1 (b), this violates criteria (iii) of predicate relaxation since there are values which satisfy the hard predicate $\neg(x > 0)$ which do take a value of 1 in $1 - (x \tilde{>} 0)$. Figure 1 illustrates this issue.

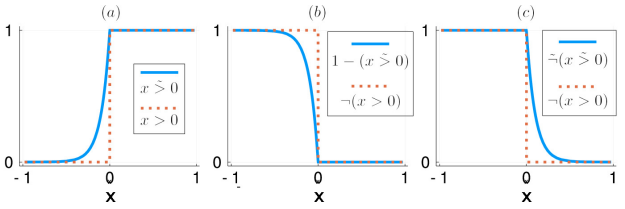

Figure 1. The problem of negation. All figures: soft (blue) and hard (red, dashed) predicates as a function of x. Left: $x \tilde{>} 0$ approximates $x > 0$. Middle: the standard approach to continuous negation $(1 - (x \tilde{>} 0))$ violates relaxation criteria (iii). Right: the desired outcome of soft negation of $x \tilde{>} 0$.

The problem of negation arises because $\tilde{\ell}$ is consistent with ℓ at 1 but not at 0, i.e., $\tilde{\ell}(\mathbf{x}; \alpha) = 1$ iff $\ell(\mathbf{x}) = 1$ at all temperatures α. In other words, $\tilde{\ell}$ is a one-sided approximation. To resolve this, we use *two-sided* soft primitives which yield a pair (a_0, a_1) where $a_0, a_0 \in [0, 1]$. a_1 is consistent with ℓ on 1, just as before, while a_0 is consistent with $\neg\,\ell$ on 1. For example if $x \tilde{>} 0$ evaluates to (a_0, a_1), then a_1 is 1 iff $x > 0$ (Figure 1 (a)) and a_0 is 1 iff $x \leq 0$ (Figure 1 (c)). Two-sided soft equality and inequalities are defined as:

$$\begin{aligned} x \tilde{=} y &= (a_0, k_\alpha(\rho(x, y))) \text{ where } a_0 = \begin{cases} \exp(1/\alpha) & \text{if } x = y \\ 1 & \text{otherwise} \end{cases} \\ x \tilde{>} y &= (k_\alpha(\rho(x, [-\infty, y])), k_\alpha(\rho(x, [y, \infty]))) \\ x \tilde{<} y &= (k_\alpha(\rho(y, [x, \infty])), k_\alpha(\rho(y, [-\infty, x]))) \end{aligned} \tag{4}$$

Two-sided soft conjunction and soft disjunction follow De Morgan's laws, and soft negation simply swaps the elements of (a_0, a_1) to yield (a_1, a_0):

$$\begin{aligned} (a_0, a_1) \tilde{\wedge} (b_0, b_1) &= (\max(a_0, b_0), \min(a_1, b_1)) \\ (a_0, a_1) \tilde{\vee} (b_0, b_1) &= (\min(a_0, b_0), \max(a_1, b_1)) \\ \tilde{\neg}(a_0, a_1) &= (a_1, a_0) \end{aligned} \tag{5}$$

In probability theory, conditioning does not treat the outputs of a predicate symmetrically. Rather, it restricts the model to values which produce 1. Consequently, when used to condition in replica exchange, a_1 and not a_0 is used in the target density.

2.2. Approximate Markov Chain Monte Carlo

We use soft predicates as approximate likelihoods for Markov Chain Monte Carlo sampling. MCMC algorithms require a function f that is proportional to the the target

density. In Bayesian inference this is the posterior, dictated by Bayes' theorem as the product of the likelihood and the prior. Inference using soft predicates has a similar form.

Definition 2. Let $\mathbf{X}$ be a model, ℓ be a predicate that conditions $\mathbf{X}$, and $\mathbf{x}$ be a realization of $\mathbf{X}$. Assuming a prior density p, the approximate posterior f is the product:

$$f(\mathbf{x}) = p(\mathbf{x}) \cdot \tilde{\ell}(\mathbf{x}) \tag{6}$$

For illustration, if $X_{1,2} \sim \mathcal{N}(0,1)$ constitute a model conditioned on $X_1 + X_2 = 0$, the approximate posterior is:

$$f_\alpha(x_1, x_2) = \mathcal{N}_{0,1}(x_1) \cdot \mathcal{N}_{0,1}(x_2) \cdot k_\alpha(\rho(x_1 + x_2, 0)) \tag{7}$$

$\tilde{\ell}$ down-weights parameter values by the degree to which they violate ℓ. This is modulated by the temperature α used in the relaxation kernels which constitute $\tilde{\ell}$. As α tends to infinity $\tilde{\ell}$ has no effect, and the approximate posterior f is equal to the prior p. As α tends to zero, f recovers the true posterior since parameter values which violate the condition are given zero weight. Between these extremes, α trades-off between tractability of inference and the fidelity of the approximation. If α is too high $\tilde{\ell}$ will diverge too greatly from ℓ. If it is too low, convergence will be slow.

Balancing Different Constraints At a fixed, nonzero temperature, the scales of variables affects their influence on the conditional distribution. For instance, consider two uniformly distributed random variables $x \sim \text{Unif}(-1, 1)$ and $y \sim \text{Unif}(-10, 10)$, and predicates p_x, p_y defined as $x \doteq 0$ and $y \doteq 0$ respectively. The expectation of a soft predicate quantifies the degree to which it is true, and is a non-decreasing function of temperature. The prior expectation of p_x is significantly larger than that of p_y. because values far from zero are more likely under y.

This disparity persists if the model is conditioned. Figure 2 shows samples from the model conditioned on $p_x \tilde{\vee} p_y$, and differing histograms of the marginals of p_x and p_y. The disparity decreases with decreasing temperature, and disappears at zero. Still, it is undesirable in practice because it means that the approximation error is distributed unevenly.

To mitigate this issue, we minimize discrepancies between sample averages of constraints. Consider the case of two soft predicates p_1 and p_2. We introduce a weight for all but the first constraint. In this case p_2 in the soft predicate is replaced by $\gamma_2 p_2$. The weight γ_2 should ensure that $\gamma_2 p_2$ has the same magnitude effect as p_1 on the energy function:

$$E_{p_{\alpha,\gamma_2}}[p_1] = E_{p_{\alpha,\gamma_2}}[\gamma_2 p_2]$$

In practice, we find γ_2 that minimizes $||E_{p_{\alpha,\gamma_2}}[p_1] - E_{p_{\alpha,\gamma_2}}[\gamma_2 p_2]||$, using exponential moving averages to approximate the expectations. Figure 2 visualizes the problem of unbalanced constraints, as well as corrections to γ.

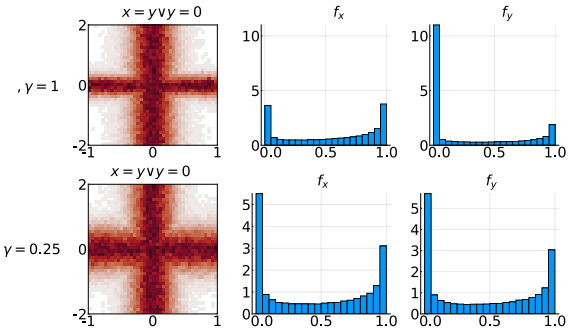

Figure 2. Magnitudes of variables can lead to uneven contributions to error if unadjusted. Top row: (left) samples from model, (middle and right) histograms of marginals. Bottom row is same model with weights adjusted to balance error.

2.3. Replica Exchange

Replica exchange (Swendsen & Wang, 1986) simulates M replicas of a model at different temperatures, and uses a Metropolis-Hastings update to periodically swap the temperatures of chains. Let f_{α_i} denote the approximate posterior function at temperature α_i, then two independent parallel chains simulating targets $f_{\alpha_1}(x)$, $f_{\alpha_2}(y)$ follow a joint target $f_{\alpha_1,\alpha_2}(x, y) = f_{\alpha_1}(x) f_{\alpha_2}(y)$. Replica exchange swaps states between the chains while preserving the joint target. Swapping states is equivalent to swapping predicates, which motivates the name "predicate exchange". Concretely, replica exchange proposes a swap from (x, y) to (y, x), and accepts it with probability $\min(1, A)$, where:

$$A = \frac{f_{\alpha_1,\alpha_2}(y, x)}{f_{\alpha_1,\alpha_2}(x, y)} = \frac{f_{\alpha_1}(y) f_{\alpha_2}(x)}{f_{\alpha_1}(x) f_{\alpha_2}(y)} \tag{8}$$

We modify standard replica exchange in two ways: (i) for exact inference, states which violate the constraint are rejected, and (ii) unlike conventional replica exchange which draws samples only from the zero-temperature chain, we accept states from any chain so long as $f_{\alpha_i}(x) = 1$.

3. Implementation

In this section we describe an implementation of predicate exchange. Our approach resembles (Wingate et al., 2011; Milch et al., 2007) as a language independent layer that can sit on top of existing programming languages and modeling formalisms. Our objective is twofold: (i) to compute the prior term p, approximate likelihood term $\tilde{\ell}$, and approximate posterior term f (Equation 7) from an arbitrary simulator π, and (ii) to perform replica exchange MCMC to sample from the posterior.

We define a simulator π as a program composed of deterministic and stochastic procedures, but where all randomness comes from a set of known random primitives. Primitives correspond to primitive parametric distribution

Example Program 1
1. $x = \text{rand}(n_x, \mathcal{N}, 0, 1)$
2. $y = \text{rand}(n_y, \mathcal{N}, 0, 1)$
3. $\text{cond}(x < y)$
4. **Return:** (x, y)

Example Program 2
$x = \text{rand}(n_x, \mathcal{N}, 0, 1)$
if $x < 0$ **then**
  $\text{cond}(x = -100)$
end if
Return: x

families, such as the uniform or normal distribution. Let $\mathcal{T}$ be a set of primitive types. Each type $\tau \in \mathcal{T}$ must support (i) evaluation of the conditional density $p_\tau(x \mid \theta_1, ..., \theta_n)$, and (ii) sampling from the distribution. Concretely, π is any nullary program that contains the statements:

1. $\text{rand}(n, \tau, \theta_1, ..., \theta_n)$ returns a random sample from $p_\tau(\cdot \mid \theta_1, ..., \theta_n)$. n is a unique name described below.
2. $\text{cond}(y)$ conditions π. It throws an error if $y \in \{0, 1\}$ is 0, and otherwise allows the execution to resume.

Example Programs 1 and 2 illustrate conditioned models.

Names (e.g. n_x) passed to rand are not the same as variable names (e.g. x) in the host programming language. If two distinct names are used, the samples output from rand will be independent or conditionally independent. Care must be taken when a program has loops to avoid inadvertently reusing the same name. A simple solution is to append the the loop counter to the name.

3.1. Tracked Soft Execution

Predicate exchange uses softexecute (Algorithm 1), which formalizes the soft execution of a program π at temperature α in the context of dictionary $\mathbb{D}$. $\mathbb{D}$ is a mutable mapping from a name to a value. In the context of a particular $\mathbb{D}$, the execution of π is deterministic. This allows the execution of π to be modulated by controlling the elements of $\mathbb{D}$.

softexecute computes the prior term p as the product of random choices in the program. That is, let $\pi_{k|x_1,...,x_{k-1}}$ be the k'th random primitive encountered while executing π, x_k be the value it takes, and x denote the set of all values of all random primitives constructed in the simulation of π, $p(x)$ is then the product:

$$p(x) = \prod_{k=1}^{K} p_\tau(x_k \mid \theta_1, ..., \theta_n) \tag{9}$$

The parameters $\theta_1, .., \theta_n$ may be fixed values or depend on values of other random primitives in π.

softexecute executes π but within a context where (i) variables $\ell_{\mathbb{D}}$ and $p_{\mathbb{D}}$ accumulate prior and approximate posterior values, and (ii) the following operators are redefined:

1. $\text{rand}(\tau, n, \theta_1, ...\theta_n)$ returns $\mathbb{D}(n)$ if n is a key in $\mathbb{D}$ (denoted $n \in \mathbb{D}$), and updates $p_{\mathbb{D}}$ according to Equation 9. If $n \notin \mathbb{D}$, the distribution is sampled from and $\mathbb{D}(n)$ is updated with this value.
2. a op b and op a for op $\in \{>, <, =, \wedge, \vee, \neg\}$ are replaced with soft versions $\tilde{\text{op}} \in \{\tilde{>}, \tilde{<}, \tilde{=}, \tilde{\wedge}, \tilde{\vee}, \tilde{\neg}\}$.
3. $\text{cond}(y)$ updates $\tilde{\ell}_{\mathbb{D}}$ with $\tilde{\ell}_{\mathbb{D}} \;\tilde{\wedge}\; y$. y will be a soft Boolean rather than a Boolean due to substitution of primitives with soft primitives as per the previous step.

L	$\mathbb{D}$	$p_{\mathbb{D}}$	$\log(\tilde{\ell}_{\mathbb{D}})$
1	$\emptyset$	1	0
2	$n_x \mapsto 0.9$	$p_{\mathcal{N}}(0.9) = 0.3$	0
3	$n_x, n_y \mapsto 0.9, 0.2$	$0.3 p_{\mathcal{N}}(0.2) = 0.1$	0
4	$n_x, n_y \mapsto 0.9, 0.2$	0.1	-700

Figure 4. softexecute on Program 1. Each row shows $\mathbb{D}$, $p_{\mathbb{D}}$ and $\log(\tilde{\ell}_{\mathbb{D}})$ just prior to executing line L. $p_{\mathcal{N}}$ denotes standard normal pdf. The final $\log(\tilde{\ell}_{\mathbb{D}})$ is $0.9 \;\tilde{<}\; 0.2$, which is -700 at $\alpha = 0.001$.

softexecute returns a value for the approximate posterior as a function of $\mathbb{D}$. Figure 4 vizualises its progression.

Control Flow Programs often have control flow. If a branch condition depends on a soft Boolean, softexecute follows the path taken by the unrelaxed program. That is, if a is a soft Boolean, **if** a **then** b evaluates b iff $a = 1$. One consequence of this is that there may be unexplored paths which would, if explored, produce values that are closer to or within the satisfying set. For illustration, if $x = -0.01$ in Example Program 2, the branch condition succeeds and softexecute will evaluate $x \;\tilde{=}\; -100$. However, with only a small change to x, the branch condition fails, and the path taken has has no conditions. $\tilde{\ell}$ therefore over approximates the change required to satisfy ℓ.

3.2. Replica Exchange

predexchange (Algorithm 2) performs replica exchange using softexecute to compute approximate posterior values. It takes as input an MCMC algorithm, which simulates a Markov Chain by manipulating elements of the $\mathbb{D}$.

predexchange rejects samples which are outside the satisfying set. If the chains converge, resulting samples are distributed according to the true, unrelaxed posterior. This is because axiom (iii) enforces that all chains at all temperatures are equivalent when restricted to the satisfying set, and the accept-reject phase carries out this restriction explicitly. Rejection sampling will fail in cases where the constraint is of measure-zero (which happens when equalities are used), since the probability of proposing a satisfying value falls to zero. In these cases we perform approximate inference by skipping the reject phase and taking samples from the highest temperature chain.

Algorithm 1 Soft Execution: softexecute$(\pi, \alpha, \mathbb{D})$

Input: program π, temperature α, dictionary $\mathbb{D}$
Initialize $\tilde{\ell}_{\mathbb{D}} = 1, p_{\mathbb{D}} = 1$
Simulate π with following subroutines redefined as:
subroutine rand$(n, \tau, \theta_1, ..., \theta_n)$
  if $n \in \mathbb{D}$ **then**
    $x \stackrel{d}{=} \mathbb{D}(n)$
  else
    $x \stackrel{d}{=}$ sample from $p_\tau(x \mid \theta_1, ..., \theta_n)$
    Update dictionary: $\mathbb{D}(n) \stackrel{d}{=} x$
  end if
  $p_{\mathbb{D}} \stackrel{d}{=} p_{\mathbb{D}} \cdot p_\tau(x \mid \theta_1, ..., \theta_m)$
  Return from subroutine: x
end subroutine

subroutine cond(y)
  $\tilde{\ell}_{\mathbb{D}} \stackrel{d}{=} \tilde{\ell}_{\mathbb{D}} \,\tilde{\wedge}\, y$
end subroutine

subroutine op$(x, \dots)$ for op $\in \{>, <, =, \wedge, \vee, \neg\}$
  Return from subroutine: $\tilde{\text{op}}\ (x, \dots)$
end subroutine

Return: $p_{\mathbb{D}} \cdot \tilde{\ell}_{\mathbb{D}}$

Algorithm 2 Predicate Exchange: predexchange

Input: program π, temperatures $\alpha_1, ..., \alpha_m$, nsamples n
Input: mcmc, nsamples between swaps q
Initialize $\mathcal{D} =$ empty collection of dictionarys
Initialize $\mathbb{D}_1^{\text{init}}, ..., \mathbb{D}_m^{\text{init}}$ empty dictionarys
Define $f_{\alpha_i}(\mathbb{D}) =$ softexecute$(\pi, \alpha_i, \mathbb{D})$
repeat
  for $i = 1$ **to** M **do**
    $\mathbb{D}_1, ..., \mathbb{D}_q \stackrel{d}{=} q$ mcmc samples at temp α_i, from $\mathbb{D}_i^{\text{init}}$
    $\mathbb{D}_i^{\text{init}} \stackrel{d}{=} \mathbb{D}_q$
    for $j = 1$ **to** q **do**
      if $f_{\alpha=\infty}(\mathbb{D}_j) \neq 0$ **then**
        append $\mathbb{D}_j$ to $\mathcal{D}$
      end if
    end for
  end for
  for $i = m$ **down to** 2 **do**
    $j \stackrel{d}{=} i - 1$
    $p \stackrel{d}{=} f_{\alpha_i}(\mathbb{D}_j) f_{\alpha_j}(\mathbb{D}_i) / f_{\alpha_i}(\mathbb{D}_i) f_{\alpha_j}(\mathbb{D}_j)$
    if $p >$ random sample in $[0, 1]$ **then**
      swap α_i with α_j
    end if
  end for
until $\mathcal{D}$ has n elements
Return: $\mathcal{D}$

4. Experiments

Experimental Setup Replica exchange requires a within chain MCMC algorithm. For finite dimensional continuous models we use the No U-Turn Sampler (Hoffman & Gelman, 2014), a variant of Hamiltonion Monte Carlo (HMC). We use reverse-mode automatic differentiation (Griewank & Walther, 2008) to compute the negative log gradient of f, which is required for HMC. For other models we use standard Metropolis Hastings by defining proposals on elements in the dictionary. In particular we use the single site Metropolis Hastings (SSMH) (Wingate et al., 2011) which modifies a single random variable at a time.

Replica exchange has a number of hyper-parameters: the number of parallel chains, the corresponding temperatures, the swapping schedule. Several good practices are outlined in (Earl & Deem, 2005). In practice, we simulate four chains with α logarithmically spaced between $\log_{10}(\alpha_1) = 5$ and $\log_{10}(\alpha_M) = -5$, and swap states that are adjacent in temperature (α_1 with α_2, α_2 with α_3, etc) every 10 iterations.

Small Models In Figure 6 we use conditioning to truncate a normal distribution. Figure 5 shows histograms of samples from a uniform prior $[-1, 1]^2$ conditioned on a variety of predicates. While simple, these examples can be challenging due to discontinuities in the approximate posterior.

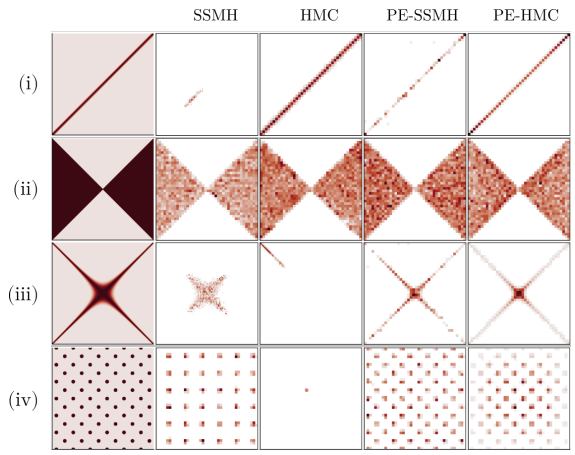

Figure 5. Samples from models using different procedures. Each model is $x, y \sim \text{Unif}(-1, 1)$ conditioned on (i) $x \,\tilde{=}\, y$, (ii) $|x| \,\tilde{>}\, |y|$, (iii) $x^2 \,\tilde{=}\, y^2$ and (iv) $\sin(kx)\cos(kx) \,\tilde{>}\, 0.9999$. Inference procedures are: Single Site Metropolis Hastings (SSMH), Hamiltonian Monte Carlo (HMC), and Predicate-Exchange (PE) using HMC and SSMH within chain.

Inverse Ray Tracing In this example (Figure 7) we sample from a posterior over scenes conditioned on an observed rendering. A scene s is a set of $n \sim \text{poisson}(\lambda = 3)$ spheres. A sphere is parameterized by color, reflectance, emission color, transparency, radius and position, all with a uniform prior. Let r be a ray tracing function that maps scenes

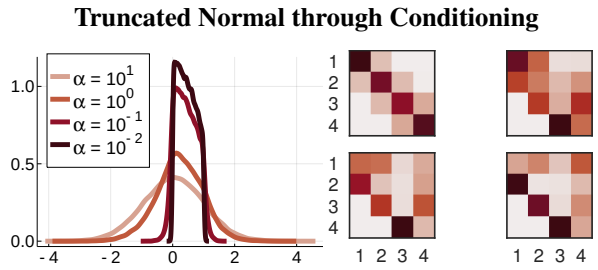

Figure 6. Left: Normalized histogram of samples of Gaussian truncated to $[0, 1]$ through conditioning, at varying temperatures. Right: transitions between different temperatures of replica exchange. For each matrix, value in row i and column j is the fraction of times replica exchange swapped a state at temperature i to temperature j. Temperature is lowest at 4 and increases exponentially to 1. Clockwise, starting top-left: transition counts are accumulated for the first, second, third and fourth quarters of simulation.

to image, i_{obs} be an observed image, and nointersect be a predicate that maps a scene to 1 iff any spheres intersect. The prior s is conditioned on the conjunction of the inverse rendering and the no-intersection constraint:

$$(r(s) = i_{obs}) \wedge \text{nointersect}(s) \tag{10}$$

Glucose Model Type 2 diabetes is a prevalent and costly condition. Keeping blood glucose within normal limits helps prevent the long-term complications of Type 2 diabetes (Brownlee & Hirsch, 2006). Models to predict the trajectories of blood glucose aid in keeping glucose within normal limits (Zeevi et al., 2015). Traditional models have been built from compositions of differential equations (Albers et al., 2017; Levine et al., 2017) whose parameters are estimated separately for each patient. An alternative approach is to use a flexible sequence model like an RNN. The problem with this approach is that an RNN can extrapolate to glucose values incompatible with human physiology. This is especially a problem where we have patients with only a few blood glucose measurements. To build an RNN model that respects physiology, we condition on it.

We compare the independent RNN model to the one with declarative knowledge on a second patient from Physionet (Moody et al., 2001). Figure 8 plots the results performed on more than 300 pairs of patients. We see that the conditional model simulates more realistic glucose dynamics for the patient with only a short observed time-series.

Benchmarks To compare predicate exchange with existing approaches we constructed a problem that scales in difficulty. Let $X_i \sim \mathcal{N}(0, 1)$ in a d-dimensional model $\mathbf{X} = (X_1, \ldots, X_d)$ conditioned on an ϵ-thick ring:

$$\ell_\epsilon(\mathbf{x}) = 1 < |\mathbf{x}| < 1 + \epsilon \tag{11}$$

Table 9 compares the sample average of particle Gibbs (PG),

Figure 7. Inverse rendering with and without no-intersection constraints. Top row: raytraced scenes. Bottom row: red pixels denote the existence of an intersection between spheres at that point. Middle and right scenes are samples from posterior over scenes given observed image on left. The observed scene has no intersections. Without the no-intersection condition (right), intersections occur. Conditioning on no-intersection eliminates intersections (middle).

sequential Monte Carlo (SMC), rejection sampling (RS) and predicate exchange (PE), varying both ϵ and d. The theoretical expectation of all models is 0. Among these methods, predicate exchange is unique in its support for inference with predicates, which makes direct comparison difficult. For all other inference procedures we use predicate relaxation to compute $d = \tilde{\ell}_\epsilon(\mathbf{x})$, and sample from $\mathbf{X}$ conditioned on $\mathcal{N}(d, 1/(2\alpha)) = 1$, where α is the lowest temperature used in predicate exchange. Predicate exchange compares favourably in most scenarios.

5. Related Work

Likelihood-free inference emerged in genetics ecology. Tavaré et al. (1997) filtered samples from a stochastic simulator to only those which matched (according to summary statistics) observed data. Weiss et al. (1998) extended this with a tolerance term, so that simulations sufficiently close to the data were accepted. A variety of approaches in this general regime (Beaumont et al., 2002; Sisson et al., 2007) fall under the heading of Approximate Bayesian Computation (ABC). Marjoram et al. (2003) simulated Markov Chains according to the prior, but applied the same summary statistic based filtering to yield approximate posterior samples. A small tolerance leads to a high rate of rejected simulations, whereas a large tolerance results in an unacceptable approximation error. Proposed solutions are dynamically decreasing the tolerance (Toni et al., 2008), importance reweighting samples based on distance (Wegmann et al., 2009), adapting the tolerance based on distance (Del Moral et al., 2012; Lenormand et al., 2013), and annealing the tolerance (Albert et al., 2015). Predicate exchange targets simulation models and uses distance metrics, but does not

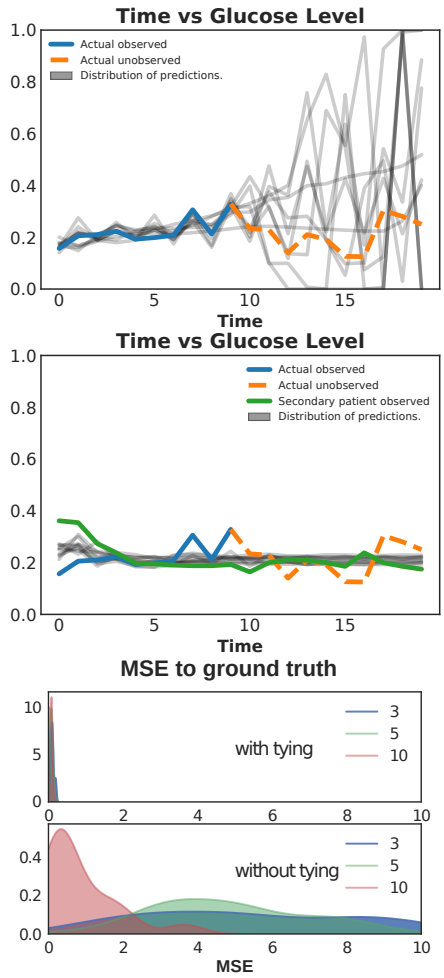

Figure 8. Top: Actual (dotted) and predicted trajectories learned using a partial trajectory. Center: Distribution of predicted trajectories learned using only the first ten data points and a tie with a secondary patient. Bottom: MSE when (above) tie is present, and (below) without tie. Tying expectations has a dramatic influence on prediction error. As more data is observed, the effect decreases.

require summary statistics.

Existing work relaxing programs includes (Chaudhuri & Solar-Lezama, 2010) which extends Gaussian smoothing to programs, and (Ritchie et al., 2015) which defines soft-equality within an HMC sampler. Predicate exchange is not bound to HMC, provides a complete soft algebra, and can sample from the true posterior in non-measure cases.

Probabilistic logics such as ProbLog (Richardson & Domingos, 2006) and Markov logic networks (De Raedt et al., 2007) extend first order logic with probabilities. Probabilistic soft logic (PSL) (Brocheler et al., 2012; Kimmig et al., 2012) uses continuous logic to encode graded beliefs. For example, isfriend(Alice, Bob) $\rightarrow 0.9$ denotes a strong friendship between Alice and Bob. In contrast, predicate exchange uses relaxation solely to make inference more

d, ϵ	$1, 10^{-1}$	$1, 10^{-5}$	$100, 10^{-1}$	$100, 10^{-5}$
PE	0.0054	0.4735	-0.00037	0.00854
SMC	0.06	-0.226	-0.018	0.09
PG	-0.029	0.239	-0.03	0.03
RS	0.003	timedout	timedout	timedout

Figure 9. Comparison of expectations of samples from d dimensional, ϵ-thick ring. Theoretical value is 0 in all cases.

tractable; soft Boolean values only exist within the sampling process and not in the resulting samples themselves. In addition, predicate exchange is motivated by languages for generative models, such as (Milch et al., 2007; Wood et al., 2014; Mansinghka et al., 2014; Goodman et al., 2008), rather than logic based languages.

Several continuous (Levin, 2000) and fuzzy (Klir & Yuan, 1995) logics apply model-theoretic tools to metric structures. Continuous logics replace the Boolean structure $\{T, F\}$, quantifiers $\forall x$ and $\exists x$, and logical connectives with continuous counter-parts. The main technical difference of our continuous logic is its two-sidedness, for negation. The main conceptual difference is that we use continuous logic only to increase the tractability of conventional inference.

6. Discussion

In this work we expanded the class of predicates that probabilistic models can be conditioned on in practice. This pushes predicates exchange into the domain of constraint solvers. Precisely which kinds of predicates are suitable for relaxation in practice remains an open question.

Equality conditions on continuous variables indicate sets of zero measure. This is problematic because the probability of proposing a satisfying state in a Markov chain becomes zero. In these cases predicate exchange samples at a minimum temperature greater than zero, which is approximate.

The problem of unexplored program paths due to control flow is related to the path explosion problem in program analysis (Cadar et al., 2008; Sen et al., 2005). Future work is to adapt program analysis solutions to this problem.

Hard predicates provide a single bit of information. Predicate relaxation effectively increases the quantity of information available to inference procedures to a real value. We anticipate that continuing to extract more kinds of information from models could lead to novel sampling methods.

7. Acknowledgments

ZT and ASL were supported by ONR N00014-17-1-2699. JB was supported by the CDS and the IESL.

References

Albers, D. J., Levine, M., Gluckman, B., Ginsberg, H., Hripcsak, G., and Mamykina, L. Personalized glucose forecasting for type 2 diabetes using data assimilation. *PLoS computational biology*, 13(4):e1005232, 2017.

Albert, C., Künsch, H. R., and Scheidegger, A. A simulated annealing approach to approximate bayes computations. *Statistics and computing*, 25(6):1217–1232, 2015.

Andrieu, C., De Freitas, N., Doucet, A., and Jordan, M. I. An introduction to mcmc for machine learning. *Machine learning*, 50(1-2):5–43, 2003.

Beaumont, M. A., Zhang, W., and Balding, D. J. Approximate bayesian computation in population genetics. *Genetics*, 162(4):2025–2035, 2002.

Brocheler, M., Mihalkova, L., and Getoor, L. Probabilistic similarity logic. *arXiv preprint arXiv:1203.3469*, 2012.

Brownlee, M. and Hirsch, I. B. Glycemic variability: a hemoglobin a1c–independent risk factor for diabetic complications. *Jama*, 295(14):1707–1708, 2006.

Cadar, C., Ganesh, V., Pawlowski, P. M., Dill, D. L., and Engler, D. R. Exe: automatically generating inputs of death. *ACM Transactions on Information and System Security (TISSEC)*, 12(2):10, 2008.

Chaudhuri, S. and Solar-Lezama, A. Smooth interpretation. In *ACM Sigplan Notices*, volume 45, pp. 279–291. ACM, 2010.

De Raedt, L., Kimmig, A., and Toivonen, H. Problog: A probabilistic prolog and its application in link discovery. *International Joint Conferences on Artificial Intelligence*, 2007.

Del Moral, P., Doucet, A., and Jasra, A. An adaptive sequential monte carlo method for approximate bayesian computation. *Statistics and Computing*, 22(5):1009–1020, 2012.

Earl, D. J. and Deem, M. W. Parallel tempering: Theory, applications, and new perspectives. *Physical Chemistry Chemical Physics*, 7(23):3910–3916, 2005.

Goodman, N. D., Mansinghka, V. K., Roy, D., Bonawitz, K., and Tenenbaum, J. B. Church: a language for generative models. In *Proceedings of the Twenty-Fourth Conference on Uncertainty in Artificial Intelligence*, pp. 220–229. AUAI Press, 2008.

Griewank, A. and Walther, A. *Evaluating derivatives: principles and techniques of algorithmic differentiation*, volume 105. Siam, 2008.

Hoffman, M. D. and Gelman, A. The no-u-turn sampler: adaptively setting path lengths in hamiltonian monte carlo. *Journal of Machine Learning Research*, 15(1):1593–1623, 2014.

Jordan, M. I., Ghahramani, Z., Jaakkola, T. S., and Saul, L. K. An introduction to variational methods for graphical models. *Machine learning*, 37(2):183–233, 1999.

Kimmig, A., Bach, S., Broecheler, M., Huang, B., and Getoor, L. A short introduction to probabilistic soft logic. In *Proceedings of the NIPS Workshop on Probabilistic Programming: Foundations and Applications*, pp. 1–4, 2012.

Klir, G. and Yuan, B. *Fuzzy sets and fuzzy logic*, volume 4. Prentice hall New Jersey, 1995.

Kulkarni, T. D., Whitney, W. F., Kohli, P., and Tenenbaum, J. Deep convolutional inverse graphics network. In *Advances in neural information processing systems*, pp. 2539–2547, 2015.

Lenormand, M., Jabot, F., and Deffuant, G. Adaptive approximate bayesian computation for complex models. *Computational Statistics*, 28(6):2777–2796, 2013.

Levin, V. Basic concepts of continuous logics. *Kybernetes*, 29(9/10):1234–1249, 2000.

Levine, M. E., Hripcsak, G., Mamykina, L., Stuart, A., and Albers, D. J. Offline and online data assimilation for real-time blood glucose forecasting in type 2 diabetes. *arXiv preprint arXiv:1709.00163*, 2017.

Mansinghka, V., Selsam, D., and Perov, Y. Venture: a higher-order probabilistic programming platform with programmable inference. *arXiv preprint arXiv:1404.0099*, 2014.

Marjoram, P., Molitor, J., Plagnol, V., and Tavaré, S. Markov chain monte carlo without likelihoods. *Proceedings of the National Academy of Sciences*, 100(26):15324–15328, 2003.

Marschner, S. R. and Greenberg, D. P. *Inverse rendering for computer graphics*. Citeseer, 1998.

Milch, B., Marthi, B., Russell, S., Sontag, D., Ong, D. L., and Kolobov, A. 1 blog: Probabilistic models with unknown objects. *Statistical relational learning*, pp. 373, 2007.

Moody, G. B., Mark, R. G., and Goldberger, A. L. Physionet: a web-based resource for the study of physiologic signals. *IEEE Engineering in Medicine and Biology Magazine*, 20(3):70–75, 2001.

Murata, G. H., Hoffman, R. M., Shah, J. H., Wendel, C. S., and Duckworth, W. C. A probabilistic model for predicting hypoglycemia in type 2 diabetes mellitus: The diabetes outcomes in veterans study (doves). *Archives of internal medicine*, 164(13):1445–1450, 2004.

Ranganath, R., Gerrish, S., and Blei, D. Black box variational inference. In *Artificial Intelligence and Statistics*, pp. 814–822, 2014.

Richardson, M. and Domingos, P. Markov logic networks. *Machine learning*, 62(1-2):107–136, 2006.

Ritchie, D., Lin, S., Goodman, N. D., and Hanrahan, P. Generating design suggestions under tight constraints with gradient-based probabilistic programming. In *Computer Graphics Forum*, volume 34, pp. 515–526. Wiley Online Library, 2015.

Sen, K., Marinov, D., and Agha, G. Cute: a concolic unit testing engine for c. In *ACM SIGSOFT Software Engineering Notes*, volume 30, pp. 263–272. ACM, 2005.

Sisson, S. A., Fan, Y., and Tanaka, M. M. Sequential monte carlo without likelihoods. *Proceedings of the National Academy of Sciences*, 104(6):1760–1765, 2007.

Swendsen, R. H. and Wang, J.-S. Replica monte carlo simulation of spin-glasses. *Physical review letters*, 57 (21):2607, 1986.

Tavaré, S., Balding, D. J., Griffiths, R. C., and Donnelly, P. Inferring coalescence times from dna sequence data. *Genetics*, 145(2):505–518, 1997.

Tavares, Z., Zhang, X., Burroni, J., Minasyan, E., Ranganath, R., and Solar-Lezama, A. The random conditional distribution for higher-order probabilistic inference. *arXiv*, 2019.

Toni, T., Welch, D., Strelkowa, N., Ipsen, A., and Stumpf, M. P. Approximate bayesian computation scheme for parameter inference and model selection in dynamical systems. *Journal of the Royal Society Interface*, 6(31): 187–202, 2008.

Wegmann, D., Leuenberger, C., and Excoffier, L. Efficient approximate bayesian computation coupled with markov chain monte carlo without likelihood. *Genetics*, 2009.

Weiss, G. and von Haeseler, A. Inference of population history using a likelihood approach. *Genetics*, 149(3): 1539–1546, 1998.

Wingate, D., Stuhlmüller, A., and Goodman, N. Lightweight implementations of probabilistic programming languages via transformational compilation. In *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, pp. 770–778, 2011.

Wood, F., Meent, J. W., and Mansinghka, V. A new approach to probabilistic programming inference. In *Artificial Intelligence and Statistics*, pp. 1024–1032, 2014.

Zeevi, D., Korem, T., Zmora, N., Israeli, D., Rothschild, D., Weinberger, A., Ben-Yacov, O., Lador, D., Avnit-Sagi, T., Lotan-Pompan, M., et al. Personalized nutrition by prediction of glycemic responses. *Cell*, 163(5):1079–1094, 2015.