Formulating Detection and Breaking of Parameter Symmetries

Kenny Chiu

December 6, 2019

1 Background

Probabilistic programming is a modern programming paradigm in which inference is automatically performed on probabilistic models that are specified by the user. The probabilistic models are often in the form of a generative Bayesian model and can be conveniently expressed by a graphical model [Rai17]. A key consideration of probabilistic programming is that the choice of the inference algorithm used for the given probabilistic model is left up to the underlying inference engine. By abstracting away the inference step, users have the flexibility to work with multiple models without needing to modify their inference procedure for each model.

However, existing inference algorithms do not work well in all models with one challenge being the possible existence of *symmetries* in the parameterization of a model [NMT13]. A symmetry can be informally described as a transformation to the model parameters that only change the likelihood components in a posterior distribution up to a scaling constant. If a symmetry is present in a model, this poses the problem of parameter nonidentifiability where the results of inference may be of poor quality and difficult to interpret. Another consequence of having parameter symmetries is that different samples of the model parameters may be strongly correlated as a result. In particular, this may reduce the performance of sampling-based inference algorithms.

The problems of parameter symmetries provide motivation for *detecting* and *breaking* them in probabilistic programs. Symmetry detection refers to determining if a symmetry exists in the parameterization of a model, and symmetry breaking refers to dealing with symmetries in some way such that the problems they cause are no longer of concern. The paper by Nishihara et al. [NMT13] introduces methods for automatically detecting various types of parameter symmetries in a probabilistic programming context. We directly build on their work by presenting two different formulations for automating symmetry detection and breaking.

1.1 Definitions and notations

We borrow the notation used by Nishihara et al. [NMT13]. We represent probabilistic models using factor graphs (θ, F) with variables $\theta = (\theta_1, ..., \theta_N)$ and factors $F = (F_1, ..., F_K)$. Let Θ denote the space of variable values. $\theta \in \Theta$ contains all parameters, latent variables, and observations (which are fixed). F represents all functions, operations, constraints, priors and likelihoods that the posterior distribution may factorize into. Given data, the unnormalized posterior distribution can then be expressed as

$$\prod_k F_k(\theta)$$

where factor F_k may not necessarily depend on all of θ. In the context of symmetries, priors are not of interest as it is assumed that the parameters of the prior are chosen by the user. The remainder of this report will refer to non-prior factors when considering F_k. We now formally define symmetries as provided in Nishihara et al.

Definition 1. A symmetry $\sigma : \Theta \to \Theta$ is a measurable function with a measurable inverse that satisfies

$$\prod_k F_k(\theta) \propto \prod_k F_k(\sigma(\theta))$$

and where if θ_n is some observed variable, then the symmetry keeps θ_n fixed. In other words, a symmetry is a transformation on the variables that preserves the product likelihood up to a scaling constant. One may also consider a specific subset of symmetries, referred to as local symmetries.

Definition 2. A local symmetry is a symmetry σ that satisfies

$$F_k(\theta) \propto F_k\left(\sigma(\theta)\right)$$

for all non-prior factors F_k. In contrast to a symmetry, a local symmetry is a transformation on the variables that preserves the likelihood at each factor up to a scaling constant. Nishihara et al. mainly focuses on detecting local symmetries as they tend to be easier to identify than non-local symmetries.

The work by Nishihara et al. discusses several types of symmetries and how they can be detected for a given factor graph. Their work is presented in the context of a probabilistic programming environment where it is assumed that all built-in factors have been annotated with constraints and/or labels that correspond to specific types of symmetries. In this report, we will assume that these annotations are intrinsic properties of the factors F_k and hence are known for a given factor graph. The next two subsections present two of the symmetries discussed in Nishihara et al. and review the approaches and annotations used in their algorithm to detect them.

1.2 Scaling symmetries

A local symmetry that describes the case where variables can be scaled without affecting the overall likelihood is known as a *scaling symmetry*.

Definition 3. A scaling symmetry is a local symmetry σ such that

$$\sigma : (\theta_1, ..., \theta_N) \longmapsto (r_1\theta_1, ..., r_N\theta_N) = (e^{d_1}\theta_1, ..., e^{d_N}\theta_N)$$

where $r_n \in \mathbb{R}_+$ and $d_n = \log r_n$. Note that the definition of scaling symmetries only includes positive scaling. A symmetry that describes the case where the signs of variables can be flipped is a *sign-flip symmetry* which we only briefly allude to in this report.

Examples of the constraints that factors impose on the scaling of the variables are provided in Table 1. For example, a *sum factor* that sums inputs $a + b = c$ preserves its integrity under scaling only if both a, b, and c are scaled by the same amount. This is enforced by the constraint $d_a = d_b = d_c$ that it imposes.

Let $d = (d_1, ..., d_N)$ be the vector of scaling exponents. Consider the matrix $\mathcal{C}$ constructed by stacking all the constraints in a given factor graph, which we refer to as the *constraint matrix*. For example, the sum factor above adds the rows $d_a - d_c = 0$ and $d_b - d_c = 0$ to $\mathcal{C}$. Its null space $\mathcal{N}(\mathcal{C})$ then describes the space of d that satisfies all the constraints. The scaling symmetries of the factor graph can then be expressed by the set $\{\sigma_d : d \in \mathcal{N}(\mathcal{C})\}$.

1.3 Permutation symmetries

The *permutation symmetry* describes the case where the variables can be permuted without changing the overall likelihood. Permutation symmetries are often present in models that have mixture components or latent features and are commonly characterized by the label-switching problem.

Definition 4. A permutation symmetry is a non-local symmetry σ that permutes the components of θ and that satisfies

$$\prod_{\text{label}(k)=c} F_k(\sigma(\theta)) = \prod_{\text{label}(k)=c} F_k(\theta)$$

for all *factor labels* c. A factor label is an identifier of a factor that is shared only between factors of the same type. For example, all binary sum factors would have the same label but a binary sum factor and a ternary sum factor would not.

To detect permutation symmetries, the factors must have factor labels and their arguments (edges) must also be labeled such that two arguments share the same label if and only if the factor is symmetric with respect to those two arguments. For example, the arguments a and b of the sum factor $c = a + b$ would have the same label. Given the labels, Nishihara et al. then reduces the detection problem to a graph

factor	constraints
$c = a + b$	$d_a = d_b = d_c$
$c = a \times b$	$d_c = d_a + d_b$
$x \geq 0$	none

Table 1: Example factors and the constraints they impose on potential scaling symmetries.

automorphism problem where the permutation symmetries of the factor graph are the automorphisms that preserve the factor and argument labels. In the context of graphs, an automorphism is a mapping of a graph onto itself that retains how the vertices and edges are connected. While determining if automorphisms exist for a given graph is a problem not known to be solvable in polynomial time [TODO ref], most graphs can be tested in linear time in practice [TODO ref].

Nishihara et al. also note that this approach is dependent on how the factor graph is structured. For example, the approach will identify the symmetry across a, b and c in the ternary sum factor $a + b + c$ but will only identify the symmetry for a, b and for $a + b$, c in the layered binary sum factors $(a + b) + c$.

1.4 Other symmetries

We give a brief introduction to the other symmetries discussed in Nishihara et al. without a formal definition. As referred to previously, the sign-flip symmetry describes the case where the signs of variables can be flipped. The symmetry is similar to the scaling symmetry with the restriction that $r_n = (-1)^{s_n}$ where $s_n \in \{0, 1\}$.

TODO

1.5 Symmetry breaking

TODO

1.6 Equivalence class

One formulation of symmetry detection and breaking that we present in this report will be based on the idea of *equivalence classes*. This section provides a brief review of the definitions and properties of equivalence classes that are relevant for our purposes.

Definition 5. Let A be a set. For all $a, b, c \in A$, an *equivalence relation* Ξ on A is a binary relation that satisfies the following three properties:

1. **reflexivity**: $a \,\Xi\, a$
2. **symmetry**: if $a \,\Xi\, b$ then $b \,\Xi\, a$
3. **transitivity**: if $a \,\Xi\, b$ and $b \,\Xi\, c$ then $a \,\Xi\, c$

An equivalence relation Ξ partitions A into subsets called equivalence classes. For $a, b \in A$, if $a \,\Xi\, b$ then a and b belong to the same equivalence class. We denote the equivalence class of a as $[a]$. The set of all equivalence classes in A with respect to Ξ is called the *quotient set* of A by Ξ, denoted A/Ξ.

Definition 6. A *section* is a function $f : A/\Xi \to A$ that maps an equivalence class to one of its members. The member $f([a])$ is called the *representative* of $[a]$ with respect to f.

2 Formulation based on equivalence classes

Our first approach to automatic symmetry detection and breaking uses a formulation based on equivalence classes. Symmetry detection can be framed as a problem of finding the set of parameters that are symmetric (an equivalence class) for a given factor graph. Similarly, symmetry breaking can be framed as choosing a single member (representative) from the set of parameters based on some desired criteria. We formalize these ideas mathematically.

Definition 7. Consider a factor graph (θ, F). Define Ξ to be the equivalence relation such that for $\theta, \theta_* \in \Theta$, we have $\theta \,\Xi\, \theta^*$ if and only if there exists a local symmetry σ such that $\sigma(\theta) = \theta^*$.

We show that Ξ is a proper equivalence relation.

Proof. Consider $\theta_1, \theta_2, \theta_3 \in \Theta$.

1. **reflexivity**: let σ be the identity map. Then trivially, $F_k(\theta_1) = F_k(\sigma(\theta_1))$.
2. **symmetry**: suppose $\theta_1 \,\Xi\, \theta_2$. Then there exists some local symmetry σ such that $\sigma(\theta_1) = \theta_2$ and $F_k(\theta_1) \propto F_k(\theta_2)$. By definition of symmety, σ^{-1} exists and $\sigma^{-1}(\theta_2) = \theta_1$. Then
$$F_k(\theta_2) \propto F_k(\theta_1) = F_k(\sigma^{-1}(\theta_2))$$
and hence $\theta_2 \,\Xi\, \theta_1$.
3. **transitivity**: suppose $\theta_1 \,\Xi\, \theta_2$ and $\theta_2 \,\Xi\, \theta_3$. Then there exists local symmetries σ_1, σ_2 such that
$$\begin{aligned} \sigma_1(\theta_1) &= \theta_2 & \qquad F_k(\theta_1) &\propto F_k(\theta_2) \\ \sigma_2(\theta_2) &= \theta_3 & \qquad F_k(\theta_2) &\propto F_k(\theta_3) \end{aligned}$$
Let $\sigma_3 = \sigma_2 \circ \sigma_1$ and so σ_3 is also measurable with a measurable inverse. Then
$$\sigma_3(\theta_1) = \sigma_2(\sigma_1(\theta_1)) = \theta_3$$
and
$$F_k(\theta_1) \propto F_k(\theta_2) \propto F_k(\theta_3) = F_k(\sigma_3(\theta_1))$$
and hence $\theta_1 \,\Xi\, \theta_3$.

□

Notice that Ξ can only be defined in the context of a given factor graph (θ, F). It is therefore mathematically more convenient to consider factor graphs equipped with a Ξ that corresponds to a specific type of symmetry. We denote this as $\mathcal{F} = (\theta, F, \Xi)$. The constraint matrix $\mathcal{C}$ of $\mathcal{F}$ described in Section 1.2 is constructed from the constraints specified by Ξ on each factor F_k.

We now formally define symmetry detection and symmetry breaking in terms of equivalence classes.

Definition 8. A *symmetry detector* $\Delta_{\mathcal{F}} : \Theta \to \Theta/\Xi$ is a measurable function that maps θ to $[\theta]$ with respect to Ξ.

Definition 9. A *symmetry breaker* $\phi_{\mathcal{F}} : \Theta/\Xi \to \Theta$ is a measurable section that maps $[\theta]$ to a representative $\theta^* \in [\theta]$.

Definition 10. An *automatic symmetry breaker* $\Phi_{\mathcal{F}} = \phi_{\mathcal{F}} \circ \Delta_{\mathcal{F}} : \Theta \to \Theta$ is the composition of a symmetry breaker and its corresponding symmetry detector. It maps θ to a representative θ^* of its equivalence class.

The dependence of Δ, ϕ, and Φ on the factor graph $\mathcal{F}$ is made clear by the subscript. When considering two functions of the same type, e.g. $\Phi_{\mathcal{F}_1}$ and $\Phi_{\mathcal{F}_2}$, we will view them as being defined on the same factor graph (θ, F) but with Ξ_1 and Ξ_2 corresponding to different types of symmetries.

This formulation is most useful in the case where the main concern is parameter nonidentifiability in the presence of symmetries. If inference has been made on a posterior distribution but the question remains

whether the inferred value is the "correct" one, it may instead be more interpretable to work with the symmetry-broken posterior distribution

$$\prod_k F_k(\Phi_{\mathcal{F}_M} \circ ... \circ \Phi_{\mathcal{F}_1}(\theta))$$

where $\Phi_{\mathcal{F}_1}, ..., \Phi_{\mathcal{F}_M}$ correspond to M different types of symmetries that are to be broken. The corresponding $\phi_{\mathcal{F}_1}, ..., \phi_{\mathcal{F}_M}$ are chosen to select representatives from the respective equivalence classes based on some desired criteria.

The following two subsections show examples of how this equivalence class formulation is used in the context of scaling symmetries and permutation symmetries.

2.1 Scaling symmetries

We describe the scaling symmetries in Section 1.2 under the formulation based on equivalence classes.

Definition 11. Let Σ_d be the diagonal matrix with $(e^{d_1}, ..., e^{d_N})$ on the diagonal. A scaling symmetry σ_d can be written as

$$\sigma_d(\theta) = \Sigma_d \theta$$

where the right-hand side is taken as a matrix-vector product. The matrix Σ_d is positive definite and so the inverse exists as required by the definition of symmetry. We then define the *scaling symmetry detector* to be

$$\Delta_{\mathcal{F}}(\theta) = \{\Sigma_d \theta : d \in \mathcal{N}(\mathcal{C})\} = [\theta]$$

where $\mathcal{C}$ is the constraint matrix of $\mathcal{F}$.

We show that this detector correctly maps the variables θ to their equivalence classes $[\theta]$ with respect to Ξ. That is, the detector maps two variables θ, $\theta^* \in \Theta$ to the same equivalence class if and only if there exists a scaling symmetry σ_d such that $\sigma_d(\theta) = \theta^*$.

Proof. Sufficiency: Suppose that $\theta^* = \Sigma_{d^*}\theta$ for some $d^* \in \mathcal{N}(\mathcal{C})$. Then

$$\begin{aligned}
\Delta_{\mathcal{F}}(\theta^*) &= \{\Sigma_d \theta^* : d \in \mathcal{N}(\mathcal{C})\} \\
&= \{\Sigma_d \Sigma_{d^*} \theta : d \in \mathcal{N}(\mathcal{C})\} \\
&= \{\Sigma_{d+d^*} \theta : d \in \mathcal{N}(\mathcal{C})\} \\
&= \{\Sigma_d \theta : d \in \mathcal{N}(\mathcal{C})\} \\
&= \Delta_{\mathcal{F}}(\theta)
\end{aligned}$$

where the above follows because Σ_d and Σ_{d^*} are diagonal and $d + d^* \in \mathcal{N}(\mathcal{C})$.

Necessity: We show this by the contrapositive. Suppose that $\theta^* \neq \Sigma_d \theta$ for all $d \in \mathcal{N}(\mathcal{C})$. Then $\theta^* \notin \Delta_{\mathcal{F}}(\theta)$ by definition of the symmetry detector. Also, by definition of equivalence class, $\theta^* \in [\theta^*] = \Delta_{\mathcal{F}}(\theta^*)$. Hence $\Delta_{\mathcal{F}}(\theta) \neq \Delta_{\mathcal{F}}(\theta^*)$ and so the detector maps θ, θ^* to different equivalence classes. □

It is more difficult to provide a concrete example of a *scaling symmetry breaker* $\phi_{\mathcal{F}}$ for two reasons. The first reason is that the equivalence class is described by using the input θ as a reference point. If $[\theta_1]$ and $[\theta_2]$ describe the same equivalence class, the breaker must be able to map both to the representative θ^*. The second reason is that the equivalence class is nonlinear in the space of diagonal matrices Σ_d (it is described by the linear space $\mathcal{N}(\mathcal{C})$ rather). This means that for a given θ, operations such as scaling may not return a member of the same equivalence class. These two reasons make it challenging to mathematically define a scaling symmetry breaker that correctly extracts the representative for a given equivalence class. An example of a possible breaker worth exploring is the one that returns the minimum norm, i.e.,

$$\phi_{\mathcal{F}}([\theta]) = \arg\min_{\theta \in [\theta]} \|\theta\|$$

However, this mathematically convenient notation hides the question of how to computationally solve for the minimum norm when $[\theta]$ may be an uncountable nonlinear space. Consideration will also need to be given in how to deal with multiple members having the same norm.

2.2 Permutation symmetries

While not a local symmetry, the permutation symmetry discussed in Section 1.3 can also be described in terms of equivalence classes. In this case, the definition of Ξ is modified to consider the existence of a non-local symmetry between two members of an equivalence class.

Definition 12. Let Π be a permutation matrix (a matrix where rows of the identity matrix are permuted). A permutation symmetry σ can be written as

$$\sigma(\theta) = \Pi\theta$$

where again the right-hand side is taken to be a matrix-vector product. Permutation matrices are orthogonal and so the inverse exists and is given by Π^T. We say that $\Pi \in \mathcal{F}$ if the rows that correspond to non-permutable variables have 1 on the diagonal. The *permutation symmetry detector* is then defined to be

$$\Delta_{\mathcal{F}}(\theta) = \{\Pi\theta : \Pi \in \mathcal{F}\} = [\theta]$$

Under this definition, it is assumed that there are no restrictions on the permutations of the permutable variables. If variables are permutable with only certain others, multiple permutation symmetry detectors are needed where each detects the symmetries in a subset of permutable variables.

Notice that if $\Pi, \Pi^* \in \mathcal{F}$, then the matrix product $\Pi\Pi^*$ is itself a permutation matrix by properties of permutation matrices. Furthermore, $\Pi\Pi^* \in \mathcal{F}$ as both matrices have 1 on the diagonal in non-permutable rows and thus so does the product. We use this fact to again show that $\Delta_{\mathcal{F}}(\theta) = \Delta_{\mathcal{F}}(\theta^*)$ if and only if there exists a permutation symmetry σ such that $\sigma(\theta) = \theta^*$.

Proof. Sufficiency: Suppose that $\theta^* = \Pi^*\theta$ for some $\Pi^* \in \mathcal{F}$. Then

$$\begin{aligned}\Delta_{\mathcal{F}}(\theta^*) &= \{\Pi\theta^* : \Pi \in \mathcal{F}\} \\ &= \{\Pi\Pi^*\theta : \Pi \in \mathcal{F}\} \\ &= \{\Pi\theta : \Pi \in \mathcal{F}\} \\ &= \Delta_{\mathcal{F}}(\theta)\end{aligned}$$

where the above follows from the aforementioned fact.

Necessity: The proof for the converse follows the same argument for the scaling symmetry. Hence this permutation symmetry detector correctly maps the variables to their equivalent classes. □

An example of a *permutation symmetry breaker* is easy to describe. For convenience, suppose that the permutable variables in θ have distinct values. Then an example permutation breaker $\phi_{\mathcal{F}}$ is one such that $\phi_{\mathcal{F}}([\theta]) = \theta^*$ where if θ_i and θ_j are permutable, then $\theta_i^* < \theta_j^*$. In the context of a probabilistic programming, this reduces to applying a sorting algorithm to certain entries of θ.

One distinction between permutation equivalence classes and scaling equivalence classes is that the permutation equivalence classes are always finite. A class can have at most $N!$ members corresponding to the possible permutations of the variables. This makes the permutation symmetry breaker computationally simpler compared to the scaling symmetry breaker in that in the worst case, the breaker can just iterate over the equivalence class to pick a representative.

2.3 Other symmetries

We end this section with a brief comment about describing the other symmetries discussed in Nishihara et al. under the equivalence class formulation. The formulation for the sign-flip symmetry would likely closely resemble that of the scaling symmetry where the symmetries are given by the null space of the constraint matrix. An added benefit is that the space of symmetries is also finite due to having only two possible values for each exponent. On the other hand, we expect the translation symmetry to be difficult to describe under the equivalence formulation. The dependence on θ in the symmetry itself makes it challenging to define detection as a function in general notation.

3 Formulation based on reparameterization

The formulation based on equivalence classes is useful when the main concern of having symmetries is parameter nonidentifiability. In this case, it is assumed that the inference algorithm is able to run as normal and that the symmetries can be broken post-inference. However, there may also be cases where the presence of symmetries impact the performance of the inference algorithm. In this situation, the symmetry should be broken before running inference. We present a formulation of symmetry breaking based on reparameterizing the model which is done pre-inference. The formulation directly builds on the symmetry detection approaches proposed in Nishihara et al. by changing the factor graph in such a way that the approaches detect no symmetries.

3.1 Scaling symmetries

The approach proposed by Nishihara et al. for detecting scaling symmetries is based on constructing and finding the null space to the constraint matrix $\mathcal{C}$. In other words, a scaling symmetry is present if the constraint matrix is underdetermined. We look to change the factor graph such that the constraint matrix becomes a determined system. We present this approach from two perspectives.

The first perspective makes the constraint matrix determined by deleting columns. This corresponds to reducing the number of variables in the factor graph by either removing or merging factors. How this can be done will largely depend on the factor graph and the source of the scaling symmetry. For example, the ternary sum of common distributions from a single distribution family is typically specified by another well-known distribution. If a scaling symmetry is introduced through a sum factor that takes in multiple Gaussian inputs, the model can be reparameterized such that the sum and the Gaussian inputs are replaced by a single Gaussian factor representing the sum. This breaks the scaling symmetry originating from the original sum factor in the factor graph.

Rather than making the constraint matrix smaller, the second perspective adds new rows to the constraint matrix. This corresponds to imposing new constraints through additional factors. TODO

3.2 Permutation symmetries

Nishihara et al. reduces detection of permutation symmetries to a problem of finding the automorphism of a labeled graph. An automorphism may exist due to the symmetric arguments of a factor being annotated with the same label. The symmetry in the arguments can be removed however by imposing an additional constraint based on ordering. For example, the sum factor $c = a + b$ is no longer symmetric with respect to a and b under the constraint $a < b$. This is equivalent to reparameterizing the factor graph such that the inputs to the factor are order statistics. Under this new parameterization, the inputs no longer receive the same label and so the only automorphism of the factor graph is the factor graph itself.

4 Open questions and research directions

A Exercises

References

[NMT13] R. Nishihara, T. Minka, and D. Tarlow. *Detecting Parameter Symmetries in Probabilistic Models.* 2013. arXiv: 1312.5386 [stat.ML].

[Rai17] T. Rainforth. "Automating Inference, Learning, and Design using Probabilistic Programming". PhD thesis. University of Oxford, 2017. URL: http://www.robots.ox.ac.uk/~twgr/assets/pdf/rainforth2017thesis.pdf.