entropy

MDPI

Article

Automating Model Comparison in Factor Graphs

Bart van Erp [1,*], Wouter W. L. Nuijten [1], Thijs van de Laar [1] and Bert de Vries [1,2]

[1] Department of Electrical Engineering, Eindhoven University of Technology, 5612 AP Eindhoven, The Netherlands
[2] GN Hearing, 5612 AB Eindhoven, The Netherlands
* Correspondence: b.v.erp@tue.nl

Abstract: Bayesian state and parameter estimation are automated effectively in a variety of probabilistic programming languages. The process of model comparison on the other hand, which still requires error-prone and time-consuming manual derivations, is often overlooked despite its importance. This paper efficiently automates Bayesian model averaging, selection, and combination by message passing on a Forney-style factor graph with a custom mixture node. Parameter and state inference, and model comparison can then be executed simultaneously using message passing with scale factors. This approach shortens the model design cycle and allows for the straightforward extension to hierarchical and temporal model priors to accommodate for modeling complicated time-varying processes.

Keywords: factor graphs; message passing; model averaging; model combination; model selection; probabilistic inference; scale factors

Citation: van Erp, B.; Nuijten, W.W.L.; van de Laar, T.; de Vries, B. Automating Model Comparison in Factor Graphs. *Entropy* **2023**, *25*, 1138. https://doi.org/10.3390/e25081138

Academic Editor: Patrick Shafto

Received: 9 June 2023
Revised: 21 July 2023
Accepted: 27 July 2023
Published: 29 July 2023

1. Introduction

The famous aphorism of George Box states that "all models are wrong, but some are useful" [1]. It is the task of statisticians and data analysts to find a model which is most useful for a given problem. The build, compute, critique and repeat cycle [2], also known as Box's loop [3], is an iterative approach for finding the most useful model. Any efforts in shortening this design cycle increase the chances of developing more useful models, which in turn might yield more reliable predictions, more profitable returns or more efficient operations for the problem at hand.

In this paper, we choose to adopt the Bayesian formalism, and therefore we specify all tasks in Box's loop as principled probabilistic inference tasks. In addition to the well-known parameter and state inference tasks, the critique step in the design cycle is also phrased as an inference task, known as Bayesian model comparison, which automatically embodies Occam's razor (Ch. 28.1, [4]). As opposed to just selecting a single model in the critique step, for different models, we better quantify our confidence about which model is best, especially when data are limited (Ch. 18.5.1, [5]). The uncertainty arising from prior beliefs $p(m)$ over a set of models m and limited observations can be naturally included through the use of Bayes' theorem:

$$p(m \mid D) = \frac{p(D \mid m)\, p(m)}{p(D)}, \tag{1}$$

which describes the posterior probabilities $p(m \mid D)$ as a function of model evidences $p(D \mid m)$ and where $p(D) = \sum_m p(D \mid m)p(m)$. Starting from Bayes' rule, we can obtain different comparison methods from the literature, such as Bayesian model averaging [6], selection, and combination [7], which we formally introduce in Section 5. We use Bayesian model comparison as an umbrella term for these three methods throughout this paper.

The task of state and parameter estimation was automated in a variety of tools, e.g., [8–14]. Bayesian model comparison, however, is often regarded as a separate task, whereas it submits to the same Bayesian formalism as state and parameter estimation.

A reason for overlooking the model comparison stage in a modeling task is that the computation of model evidence $p(D \mid m)$ is, in most cases, not automated and therefore still requires error-prone and time-consuming manual derivations, in spite of its importance and the potential data representation improvement that can be achieved by, for example, including a Bayesian model combination stage in the modeling process [7].

This paper aims to automate the Bayesian model comparison task and is positioned between the mixture model and 'gates' approaches of [15,16], respectively, which we will describe in more detail in Section 2. Specifically, we specify the model comparison tasks as a mixture model, similarly to that in [15], on a factor graph with a custom mixture node for which we derive automatable message-passing update rules, which perform both parameter and state estimation, and model comparison. These update rules generalize the model comparison to arbitrary models submitting to exact inference, as the operations of the mixture node are ignorant about the adjacent subgraphs. Additionally, we derive three common model comparison methods from the literature (Bayesian model averaging, selection, and combination) using the custom mixture node.

In short, this paper derives automated Bayesian model comparison using message passing in a factor graph. After positioning our paper in Section 2 and after reviewing factor graphs and message passing-based probabilistic inference in Section 3, we make the following contributions:

1. We show that the Bayesian model comparison can be performed through message passing on a graph, where the individual model performance results are captured in a single factor node as described in Section 4.1.
2. We specify a universal mixture node and derive a set of custom message-passing update rules in Section 4.2. Performing probabilistic inference with this node in conjunction with scale factors yields different Bayesian model comparison methods.
3. Bayesian model averaging, selection, and combination are recovered and consequently automated in Sections 5.1–5.3 by imposing a specific structure or local constraints on the model selection variable m.

We verify our presented approach in Section 6.1. We illustrate its use for models with both continuous and discrete random variables in Section 6.2.1, after which we continue with an example of voice activity detection in Section 6.2.2, where we add temporal structure on m. Section 7 discusses our approach, and Section 8 concludes the paper.

2. Related Work

This section discusses related work and aims at providing the clear positioning of this paper for our contributions that follow in the upcoming sections.

The task of model comparison is widely represented in the literature [17], for example, concerning hypothesis testing [18,19]. Bayesian model averaging [6] can be interpreted as the simplest form of model comparison that uses the Bayesian formalism to retain the first level of uncertainty in the model selection process [20]. Bayesian model averaging has proven to be an effective and principled approach that converges with infinite data to the single best model in the set of candidate models [21–23]. When the true underlying model is not part of this set, the data is often better represented by ad hoc methods [24], such as ensemble methods. In [7], the idea of Bayesian model comparison is introduced, which basically performs Bayesian model averaging between mixture models comprising the candidate models with different weights. Another ensemble method is proposed in [23,25], which uses (hierarchical) stacking [26] to construct predictive densities whose weights are data dependent.

Automating the model design cycle [2] under the Bayesian formalism has been the goal of many probabilistic programming languages [8–14]. This paper focuses on message passing-based approaches, which leverage the conditional independencies in the model structure for performing probabilistic inference, e.g., [27–30], which will be formally introduced in Section 3.2. Contrary to alternative sampling-based approaches, message passing excels in modularity, speed and efficiency, especially when models submit to closed-form

(variational) message computations. Throughout this paper, we follow the spirit of [31], which shows that many probabilistic inference algorithms, such as (loopy) belief propagation [32,33], variational message passing [30,34], expectation maximization [35], and expectation propagation [36] can all be phrased as a constrained Bethe free energy [37] minimization procedure. Specifically, in Section 5, we aim to phrase different Bayesian model comparison methods as automatable message-passing algorithms. Not only does this have the potential to shorten the design cycle but also to develop novel model comparison schemes.

The connection between (automatable) state and parameter inference, versus model comparison was explored recently by [15,22], who frame the problem of model comparison as a "mixture model estimation" task that is obtained by combining the individual models into a mixture model with weights representing the model selection variable. The exposition in [15,22] is based on relatively simple examples that do not easily generalize to more complex models for the model selection variable and for the individual cluster components. In the current paper, we aim to generalize the mixture model estimation approach by an automatable message passing-based inference framework. Specifically, we build on the results of the recently developed scale factors (Ch. 6, [38,39]), which we introduce in Section 3.3. These scale factors support the efficient tracking of local summaries of model evidences, thus enabling model comparison in generic mixture models; see Sections 4 and 5.

The approach we present in the current paper is also similar to the concept of 'gates', introduced in [16]. Gates are factor nodes that switch between mixture components with which we can derive automatable message-passing procedures. Mathematically, a gate represents a factor node of the form $f(s, m) = \prod_{k=1}^{K} f_k(s_k)^{m_k}$, where the model selection variable m is a one-of-K coded vector defined as $m_k \in \{0, 1\}$ subject to $\sum_{k=1}^{K} m_k = 1$. The variables are $s = \bigcup_{k=1}^{K} s_k$. Despite the universality of the Gates approach, the inference procedures in [16] focus on variational inference [30,34,40] and expectation propagation [36]. The mixture selection variable m is then updated based on "evidence-like quantities". In the variational message-passing case, these quantities resemble local Bethe free energy contributions, which only take into account the performance around the gate factor node, disregarding the performance contributions of other parts in the model. Because of the local contributions, the message-passing algorithm can be very sensitive to its initialization, which has the potential to yield suboptimal inference results.

In the current paper, we extend gates to models submitting to exact inference using scale factors, which allows for generalizing and automating the mixture models of [15,22]. With these advances we can automate well-known Bayesian model comparison methods using message passing, enabling the development of novel comparison methods.

3. Background Material

This section aims to provide a concise review of factor graphs and message-passing algorithms, as we deem these concepts essential to appreciate our core contributions, which we present in Sections 4 and 5. This review is intentionally not exhaustive; instead, we provide references to works that help to obtain a deeper understanding about the material covered here. In Section 3.1, we introduce factor graphs as a way to visualize factorizable (probabilistic) models. Section 3.2 then describes how probabilistic inference can be efficiently performed through message passing, utilizing the inherent factorization of the model. The model evidence can be tracked locally with message passing, using scale factors as described in Section 3.3. Finally, Section 3.4 introduces the variational free energy as a bound on the model evidence.

3.1. Forney-Style Factor Graphs

A factor graph is a specific type of probabilistic graphical model. Here we use the Forney-style factor graph (FFG) framework as introduced in [41] with notational conventions adopted from [27] to visualize our probabilistic models. An FFG can be used to represent a factorized function

$$f(s) = \prod_{a \in \mathcal{V}} f_a(s_a), \tag{2}$$

where s collects all variables in the function. The subset $s_a \subseteq s$ contains all argument variables of a single factor f_a. FFGs visualize the factorization of such a function as a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where nodes $\mathcal{V}$ and edges $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$ represent factors and variables, respectively. An edge connects to a node only if the variable associated with the edge is an argument of the factor associated with the node. Nodes are indexed by the variables a, b, and c, where edges are indexed by i and j unless stated otherwise. The set of edges connected to node $a \in \mathcal{V}$ is denoted by $\mathcal{E}(a)$, and the set of nodes connected to edge $i \in \mathcal{E}$ is referred to as $\mathcal{V}(i)$. As an example, consider the model $f(s_1, s_2, s_3, s_4)$ with factorization

$$f(s_1, s_2, s_3, s_4) = f_a(s_1) f_b(s_1, s_2) f_c(s_3) f_d(s_2, s_3, s_4) \tag{3}$$

The FFG representation of (3) is shown in Figure 1. For a more thorough review of factor graphs, we refer the interested reader to [27,28].

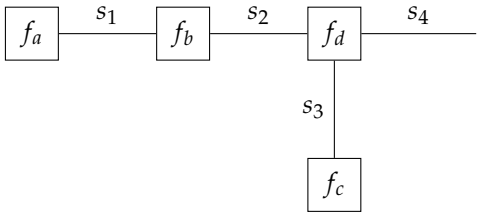


Figure 1. A Forney-style factor graph representation of the factorized function in (3).

3.2. Sum-Product Message Passing

Consider the normalized probabilistic model

$$p(y, s) = \prod_{a \in \mathcal{V}} f_a(y_a, s_a), \tag{4}$$

with observed and latent sets of variables y and s, respectively. Note here that the subset $y_a \subseteq y$ can be empty, for example, when dealing with prior distributions. Upon observing the realizations $\hat{y}$, the corresponding model $p(y = \hat{y}, s)$ becomes unnormalized. Probabilistic inference in this model then concerns the computation of the posterior distribution over the latent variables $p(s \,|\, y = \hat{y})$ and of the model evidence $p(y = \hat{y})$ as $p(y = \hat{y}, s) = p(s \,|\, y = \hat{y}) p(y = \hat{y})$. Consider the global integration over all variables in (4), except for s_j as $\int p(y = \hat{y}, s) \, \mathrm{d}s_{\setminus j}$. (Integrals are taken over the support over the variables. If a variable is discrete valued, integral operators are replaced with summation operators. For a consistent exposition of our work, we use integral operators throughout the paper.) This large global integration can be performed through a set of smaller local computations as a result of the assumed factorization in (4). These smaller local computations can be considered to summarize the part of the graph that is being integrated over and are termed messages, which graphically can be interpreted to propagate over the edges in the graph. These messages are denoted by μ and can be locally computed on the graph. The sum–product message $\vec{\mu}_{s_j}(s_j)$ flowing out of the node $f_a(y_a = \hat{y}_a, s_a)$ with incoming messages $\vec{\mu}_{s_i}(s_i)$ is given by [29]

$$\vec{\mu}_{s_j}(s_j) = \int f_a(y_a = \hat{y}_a, s_a) \prod_{i \neq j} \vec{\mu}_{s_i}(s_i) \, \mathrm{d}s_{a \setminus j}. \tag{5}$$

We represent edges in the graph by arbitrarily directed arrows in order to distinguish between forward and backward messages propagating in or against the direction of an edge s_j as $\vec{\mu}_{s_j}(s_j)$ and $\overleftarrow{\mu}_{s_j}(s_j)$, respectively. Following this approach, the global integration reduces to the product of the messages of the variable of interest as $\int p(y = \hat{y}, s) \mathrm{d}s_{\setminus j} = \vec{\mu}_{s_j}(s_j) \overleftarrow{\mu}_{s_j}(s_j)$ for acyclic models.

Posterior distributions on edges and around nodes can then be computed according to

$$p(s_j \mid y = \hat{y}) = \frac{\vec{\mu}_{s_j}(s_j)\overleftarrow{\mu}_{s_j}(s_j)}{\int \vec{\mu}_{s_j}(s_j)\overleftarrow{\mu}_{s_j}(s_j)\,\mathrm{d}s_j} \tag{6}$$

and

$$p(s_a \mid y = \hat{y}) = \frac{f_a(y_a = \hat{y}_a, s_a)\prod\limits_{i\in\mathcal{E}(a)}\vec{\mu}_{s_i}(s_i)}{\int f_a(y_a = \hat{y}_a, s_a)\prod\limits_{i\in\mathcal{E}(a)}\vec{\mu}_{s_i}(s_i)\,\mathrm{d}s_a}, \tag{7}$$

respectively [31].

Derivations of the message-passing update rule in (5) by phrasing inference as a constrained Bethe free energy minimization procedure are presented in [31]. Through a similar procedure, one can obtain alternative message-passing algorithms, such as variational message passing [30,34,40], expectation propagation [36], expectation maximization [35] and hybrid algorithms.

3.3. Scale Factors

The previously discussed integration $\int p(y = \hat{y}, s)\,\mathrm{d}s_{\setminus j}$ can be represented differently as

$$\int p(y = \hat{y}, s)\,\mathrm{d}s_{\setminus j} = p(y = \hat{y})\int p(s \mid y = \hat{y})\,\mathrm{d}s_{\setminus j} = p(y = \hat{y})p(s_j \mid y = \hat{y}), \tag{8}$$

where $p(s_j \mid y = \hat{y})$ is the marginal distribution of s_j. The implications of this result are significant: the product of two colliding sum–product messages $\vec{\mu}_{s_j}(s_j)\overleftarrow{\mu}_{s_j}(s_j)$ in an acyclic graph results in the scaled marginal distribution $p(y = \hat{y})p(s_j \mid y = \hat{y})$. Because of the normalization property of $p(s_j \mid y = \hat{y})$, it is possible to obtain both the normalized posterior $p(s_j \mid y = \hat{y})$ as the model evidence $p(y = \hat{y})$ on any edge and around any node in the graph.

Theorem 1. *Consider an acyclic Forney-style factor graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$. The model evidence of the corresponding model $p(y = \hat{y}, s)$ can be computed at any edge in the graph as $\int \vec{\mu}_{s_j}(s_j)\overleftarrow{\mu}_{s_j}(s_j)\,\mathrm{d}s_j$ for all $j \in \mathcal{E}$ and at any node in the graph as $\int f_a(y_a = \hat{y}_a, s_a)\prod_{i\in\mathcal{E}(a)}\vec{\mu}_{s_i}(s_i)\,\mathrm{d}s_a$ for all $a \in \mathcal{V}$.*

Proof. See Appendix A.1. □

What enables this local computation of the model evidence is the scaling of the messages resulting from the equality in (5). As a result, the messages $\vec{\mu}_{s_j}(s_j)$ can be decomposed as

$$\vec{\mu}_{s_j}(s_j) = \vec{\beta}_{s_j}\vec{p}_{s_j}(s_j), \tag{9}$$

where $\vec{p}_{s_j}(s_j)$ denotes the probability distribution representing the normalized functional form of the message $\vec{\mu}_{s_j}(s_j)$. The term $\vec{\beta}_{s_j}$ denotes the scaling of the message $\vec{\mu}_{s_j}(s_j)$, also known as the scale factor (Ch. 6, [38,39]). Scale factors can be interpreted as local summaries of the model evidence that are passed along the graph.

3.4. Variational Free Energy

In practice, however, the computation of the model evidence and, therefore, the posterior distribution is intractable. Variational inference provides a generalized view that supports probabilistic inference in these types of models by approximating the exact posterior $p(s \mid y = \hat{y})$ with a variational posterior $q(s)$ that is constrained to be within a family of distributions $q \in \mathcal{Q}$. Variational inference optimizes (the parameters of) the variational distribution $q(s)$ by minimizing the variational free energy (VFE) of a single model, defined as

$$\mathrm{F}[q] = \mathbb{E}_{q(s)}\left[\ln\frac{q(s)}{p(y = \hat{y}, s)}\right] = \mathrm{KL}\left[q(s) \,\|\, p(s \mid y = \hat{y})\right] - \ln p(y = \hat{y}), \tag{10}$$

through, for example, coordinate or stochastic gradient descent.

The variational free energy can serve as a bound to the model evidence in (1) for model comparison (Ch. 10.1.4, [42–44]). It is important to emphasize that the VFE not only encompasses the model evidence but also the Kullback–Leibler (KL) divergence between the variational and exact posterior distributions obtained from the inference procedure.

4. Universal Mixture Modeling

This section derives a custom factor node that allows for performing model comparison as an automatable message-passing procedure in Section 5. In Section 4.1, we specify a variational optimization objective for multiple models at once, where the optimization of the model selection variable can be rephrased as a probabilistic inference procedure on a separate graph with a factor node encapsulating the model-specific performance metrics. Section 4.2 further specifies this node and derives custom message-passing update rules that allow for jointly computing (1) and for performing state and parameter inference.

Before continuing our discussion, let us first describe the notational conventions adopted throughout this section. In Section 3, only a single model is considered. Here, we cover K normalized models, selected by the model selection variable m, which comprises a 1-of-K binary vector with elements $m_k \in \{0, 1\}$ constrained by $\sum_{k=1}^{K} m_k = 1$. The individual models $p(y_k, s_k \mid m_k = 1)$ are indexed by k, where y_k and s_k collect the observed and latent variables in that model.

4.1. A Variational Free Energy Decomposition for Mixture Models

Consider the normalized joint model

$$p(y, s, m) = p(m) \prod_{k=1}^{K} p(y_k, s_k \mid m_k = 1)^{m_k} \tag{11}$$

specifying a mixture model over the individual models $p(y_k, s_k \mid m_k = 1)$, with a prior $p(m)$ on the model selection variable m and where $y = \bigcup_{k=1}^{K} y_k$ and $s = \bigcup_{k=1}^{K} s_k$. Based on this joint model, let us define its variational free energy F as

$$\begin{aligned} \mathrm{F}[q] &= \mathbb{E}_{q(s,m)}\left[\ln \frac{q(s,m)}{p(y = \hat{y}, s, m)}\right], \\ &= \mathbb{E}_{q(m)}\left[\ln \frac{q(m)}{p(m)}\right] + \mathbb{E}_{q(m)}\left[\prod_{k=1}^{K} \left(\mathrm{F}_k[q]\right)^{m_k}\right], \end{aligned} \tag{12}$$

in which the joint variational posterior $q(s, m)$ factorizes as $q(s, m) = q(m) \prod_{k=1}^{K} q(s_k \mid m_k = 1)^{m_k}$, and where F_k denotes the variational free energy of the k-th model. This decomposition is obtained by noting that m is a 1-of-K binary vector. Furthermore, derivations of this decomposition are provided in Appendix B.1.

This definition also appeared in a similar form in (Ch.10.1.4, [42]) and in the reinforcement learning and active inference community as an approach to policy selection (Section 2.1, [45]). From this definition, it can be noted that the VFE for mixture models can also be written as

$$\mathrm{F}[q] = \mathbb{E}_{q(m)}\left[\ln \frac{q(m)}{p(m) f_m(m)}\right] \tag{13}$$

where

$$f_m(m) = \prod_{k=1}^{K} \exp(-\mathrm{F}_k[q])^{m_k}, \tag{14}$$

as shown in Appendix B.1. This observation implies that the obtained VFEs of the individual submodels can be combined into a single factor node f_m, representing a scaled categorical distribution, which terminates the edge corresponding to m as shown in Figure 2. The specification of f_m allows for performing inference in the overcoupling model following existing inference procedures, similar to that for the individual submodels. This is

in line with the validity of Bayes' theorem in (1) for both state and parameter inference, and model comparison. Importantly, the computation of the VFE in acyclic models can be automated [31]. Therefore, model comparison itself can also be automated. For cyclic models, one can resort to approximating the VFE with the Bethe free energy [31,37].

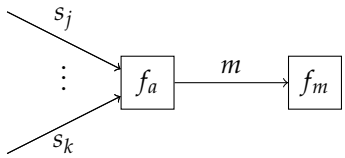


Figure 2. Subgraph containing model selection variable m. The node f_m terminates the subgraph and is defined in (14).

In practice, the prior model $p(m)$ might have hierarchical or temporal dynamics, including additional latent variables. These can be incorporated without loss of generality due to the factorizable structure of the joint model, supported by the modularity of factor graphs and the corresponding message-passing algorithms, as shown in Figure 2.

4.2. A Factor Graph Approach to Universal Mixture Modeling: A General Recipe

In this subsection, we present the general recipe for computing the posterior distributions over the variables s and model selection variable m in universal mixture models. Section 4.3 provides an illustrative example that aids the exposition in this section. The order in which these two section are read is a matter of personal preference.

In many practical applications, distinct models $p(y_k, s_k \mid m_k = 1)$ partly overlap in both structure and variables. These models may, for example, just differ in terms of priors or likelihood functions. Let $f_o(y_o, s_o)$ be the product of factors which are present in all different factorizable models $p(y_k, s_k \mid m_k = 1)$, with overlapping variables $s_o = \bigcap_{k=1}^{K} s_k$ and $y_o = \bigcap_{k=1}^{K} y_k$. Based on this description, we define a universal mixture model as in [15] encompassing all individual models as

$$\begin{aligned} p(y, s, m) &= p(m) \prod_{k=1}^{K} p(y_k, s_k \mid m_k = 1)^{m_k} \\ &= p(m) f_o(y_o, s_o) \prod_{k=1}^{K} \left(\frac{p(y_k, s_k \mid m_k = 1)}{f_o(y_o, s_o)} \right)^{m_k}, \end{aligned} \tag{15}$$

with model selection variable m. Here, the overlapping factors are factored out from the mixture components. Figure 3 shows a visualization of the transformation from K distinct models into a single mixture model. With the transformation from the different models into a single mixture model presented in Figure 3, it becomes possible to include the model selection variable m into the same probabilistic model.

In these universal mixture models, we are often interested in computing the posterior distributions of (1) the overlapping variables s_o marginalized over the distinct models and of (2) the model selection variable m. Given the posterior distributions $q(s_o \mid m_k = 1)$ over variables s_o in a single model m_k, we can compute the joint posterior distribution over all overlapping variables $q(s_o)$ as

$$q(s_o) = \mathbb{E}_{q(m)} \left[\prod_{k=1}^{K} q(s_o \mid m_k = 1)^{m_k} \right], \tag{16}$$

marginalized over the different models m. In the generic case, this computation follows a three-step procedure. First, the posterior distributions $q(s_k \mid m_k = 1)$ are computed in the individual submodels through an inference algorithm of choice. Then, based on the computed VFE $\mathrm{F}_k[q]$ of the individual models, the variational posterior $q(m)$ can be calculated. Finally, the joint posterior distribution $q(s_o)$ can be computed using (16).

Here, we restrict ourselves to acyclic submodels. We show that the previously described inference procedure for computing the joint posterior distributions can be performed jointly with the process of model comparison through message passing with scale factors. In order to arrive at this point, we combine the different models into a single mixture model inspired by [15,22]. Our specification of the mixture model, however, is more general compared to that of [15,22], as it does not constrain the hierarchical depth of the overlapping or distinct models and also works for nested mixture models.

Table 1 introduces the novel mixture node, which acts as a switching mechanism between different models, based on the selection variable m. It connects the model selection variable m and the overlapping variables $s_j \mid m_k = 1$ for the different models m_k, to the variable s_j marginalized over m. Here, the variables s_j connect the overlapping to the non-overlapping factors.

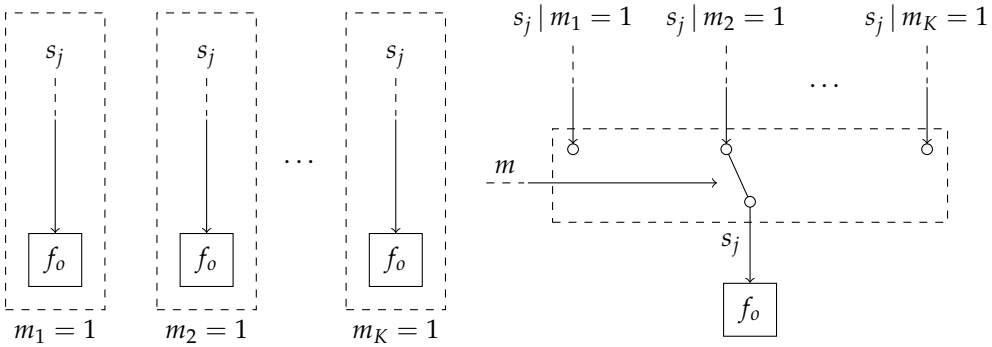


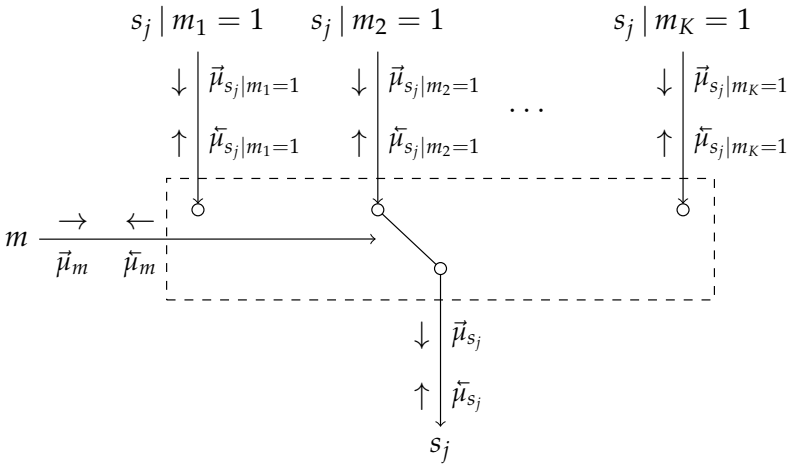
Figure 3. (**left**) Overview of the traditional process of model comparison. Here, inference is performed in a set of K models, after which the models are compared. These models may partially overlap in both variables as in structure. Specifically, in this example, the variables s_j connect the overlapping factors f_o to the non-overlapping factors. The notation $s_j \mid m_k = 1$ denotes the variable s_j in the k-th model. (**right**) Our approach to model comparison based on mixture modeling. The different models are combined into a single graph representing a mixture model, where the model selection variable m specifies the component assignment. The variable s_j without conditioning implies that it has been marginalized over the different models m.

The messages in Table 1 are derived in Appendices B.2 and B.3, and can be intuitively understood as follows. The message $\overleftarrow{\mu}_m(m)$ represents the unnormalized distribution over the model evidence corresponding to the individual models. Based on the scale factors of the incoming messages, the model evidence can be computed. The message $\overleftarrow{\mu}_{s_j \mid m_k=1}(s_j)$ equals the incoming message from the likelihood $\overleftarrow{\mu}_{s_j}(s_j)$. It will update $s_j \mid m_k = 1$ as if the k-th model is active. The message $\vec{\mu}_{s_j}(s_j)$ represents a mixture distribution over the incoming messages $\vec{\mu}_{s_j}(s_j)$, where the weightings are determined by the message $\vec{\mu}_m(m)$ and the scale factors of the messages $\vec{\mu}_{s_j \mid m_k=1}(s_j)$. This message can be propagated as a regular message over the overlapping model segment yielding the marginal posterior distributions over all variables in the overlapping model segment.

Theorem 2. *Consider multiple acyclic FFGs. Given the message $\vec{\mu}_{s_j}(s_j)$ in Table 1 that is marginalized over the different m models, propagating this message through the factor $f_a(y_a, s_a)$, which overlaps for all models with $s_j \in s_a$, yields again messages which are marginalized over the different models.*

Proof. See Appendix A.2. □

Table 1. Table containing (top) the Forney-style factor graph representation of the mixture node. (bottom) The derived outgoing messages for the mixture node. It can be noted that the backward message towards m resembles a scaled categorical distribution and that the forward message towards s_j represents a mixture distribution. Derivations of the messages $\overleftarrow{\mu}_m(m)$ and $\vec{\mu}_{s_j}(s_j)$ are presented in Appendices B.2 and B.3, respectively.

Factor Node


Messages	**Functional form**
$\overleftarrow{\mu}_m(m)$	$\prod_{k=1}^{K}\left(\int \vec{\mu}_{s_j\|m_k=1}(s_j)\overleftarrow{\mu}_{s_j}(s_j)\mathrm{d}s_j\right)^{m_k}$
$\vec{\mu}_{s_j}(s_j)$	$\sum_{k=1}^{K}\vec{\mu}_m(m_k=1)\vec{\mu}_{s_j\|m_k=1}(s_j)$
$\overleftarrow{\mu}_{s_j\|m_k=1}(s_j)$	$\overleftarrow{\mu}_{s_j}(s_j)$

4.3. A Factor Graph Approach to Universal Mixture Modeling: An Illustrative Example

Consider the two probabilistic models

$$p(y, s \mid m_1 = 1) = p(y \mid s)\, p(s \mid m_1 = 1), \tag{17}$$
$$p(y, s \mid m_2 = 1) = p(y \mid s)\, p(s \mid m_2 = 1), \tag{18}$$

which share the same likelihood model with a single observed and latent variable y and s, respectively. The model selection variable m is subject to the prior

$$p(m) = \mathrm{Ber}(m \mid \pi) = \pi^{m_1}(1-\pi)^{m_2}, \tag{19}$$

with π denoting the success probability. This allows for the specification of the mixture model

$$p(y, s, m) = p(m)\, p(y \mid s) \prod_{k=1}^{2} p(s \mid m_k = 1)^{m_k}, \tag{20}$$

which we visualize in Figure 4.

Suppose we are interested in computing the posterior probabilities $p(s \mid y = \hat{y})$, marginalized over the distinct models, and $p(m \mid y = \hat{y})$. The model evidence of both models can be computed using scale factors locally on the edge corresponding to s as

$$p(y = \hat{y} \mid m_1 = 1) = \int \vec{\mu}_{s|m_1=1}(s)\overleftarrow{\mu}_s(s)\,\mathrm{d}s = \int p(s \mid m_1 = 1)\, p(y = \hat{y} \mid s)\,\mathrm{d}s,$$
$$p(y = \hat{y} \mid m_2 = 1) = \int \vec{\mu}_{s|m_2=1}(s)\overleftarrow{\mu}_s(s)\,\mathrm{d}s = \int p(s \mid m_2 = 1)\, p(y = \hat{y} \mid s)\,\mathrm{d}s,$$

which takes place inside the mixture node for computing $\overleftarrow{\mu}_m(m)$. Together with the forward message over edge m, we obtain the posterior

$$p(m \mid y = \hat{y}) = \frac{\vec{\mu}_m(m)\overleftarrow{\mu}_m(m)}{\sum_{k=1}^{2} \vec{\mu}_m(m_k = 1)\overleftarrow{\mu}_m(m_k = 1)} = \frac{p(m)p(y = \hat{y} \mid m)}{p(y = \hat{y})}. \tag{21}$$

The posterior distribution over s for the first model can be computed as

$$p(s \mid y = \hat{y}, m_1 = 1) = \frac{\vec{\mu}_{s|m_1=1}(s)\overleftarrow{\mu}_s(s)}{p(y = \hat{y} \mid m_1 = 1)}.$$

From (16), we can then compute the posterior distribution over s marginalized over both models as

$$\begin{aligned} p(s \mid y = \hat{y}) &= p(m_1 = 1 \mid y = \hat{y})p(s \mid y = \hat{y}, m_1 = 1) \\ &\quad + p(m_2 = 1 \mid y = \hat{y})p(s \mid y = \hat{y}, m_2 = 1), \\ &= \frac{p(m_1 = 1)p(y = \hat{y} \mid m_1 = 1)}{p(y = \hat{y})} \frac{\vec{\mu}_{s|m_1=1}(s)\overleftarrow{\mu}_s(s)}{p(y = \hat{y} \mid m_1 = 1)} \\ &\quad + \frac{p(m_2 = 1)p(y = \hat{y} \mid m_2 = 1)}{p(y = \hat{y})} \frac{\vec{\mu}_{s|m_2=1}(s)\overleftarrow{\mu}_s(s)}{p(y = \hat{y} \mid m_2 = 1)}, \\ &= \frac{\Big(p(m_1 = 1)\vec{\mu}_{s|m_1=1}(s) + p(m_2 = 1)\vec{\mu}_{s|m_2=1}(s)\Big)\overleftarrow{\mu}_s(s)}{p(y = \hat{y})}. \end{aligned}$$

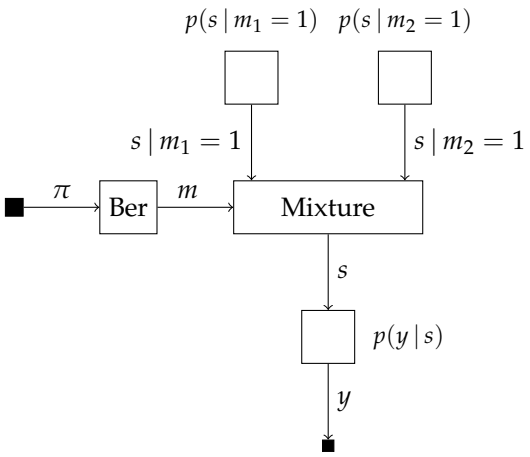
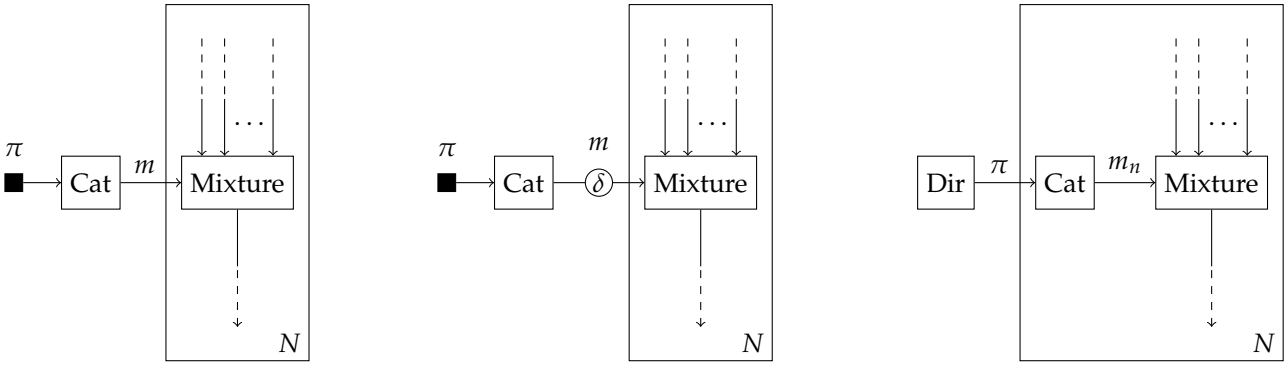


Figure 4. Factor graph visualization of (20) in the example sketched in Section 4.3.

5. Model Comparison Methods

In this section, we introduce three Bayesian model comparison methods from the literature: model averaging [6], selection and combination [7]. For each of these methods, we describe how to automate them using message passing with the mixture node in Table 1. The factor graph approach here aids the intuitive understanding of the different approaches, as their distinctions are sometimes obscure in the literature. As we will show, each method describes an inference procedure on a slightly different model for the model selection variable m, possibly with different variational constraints as visualized in Figure 5.



(**a**) Model averaging (**b**) Model selection (**c**) Model combination

Figure 5. Schematic overview of (**a**) Bayesian model averaging, (**b**) selection and (**c**) combination as specified in Sections 5.1–5.3. This overview explicitly visualizes the structural differences between the prior distributions and form constraints imposed on the model selection variable m. The edges crossing the plates are implicitly connected through equality nodes.

5.1. Bayesian Model Averaging

Bayesian model averaging (BMA) can be considered the simplest form of model comparison and is therefore mentioned in many works, e.g., (Ch. 14.1, [6,42]). BMA completes the model specification by specifying a categorical prior distribution over the models m as

$$p(m) = \text{Cat}(m \mid \pi), \tag{22}$$

where π denotes the vector of event probabilities. BMA then aims at computing the posterior distribution over the models $q(m)$. Given a set of possible models, or hypotheses, with BMA, the posterior distribution $q(m)$ converges with infinite data to a Kronecker delta function that selects the single model which is the most likely given the observed set of data [15,21]. Figure 5a provides a visual representation of Bayesian model averaging.

5.2. Bayesian Model Selection

Bayesian model selection (BMS) is a further specification of BMA as illustrated in Figure 5b, which selects the model out of a group of models that is the maximum a posteriori (MAP) estimate of m, e.g., (Ch. 5.3, [46]). Where BMA returns a posterior probability over the models m, BMS only returns the most probable model. In addition to the specification of the model prior of (22), BMS can be interpreted to enforce a form constraint [31] on the variable m. Specifically, we constrain the posterior distribution $q(m)$ to be a Kronecker delta function $\delta(\cdot)$, centered around the MAP estimate of m as

$$q(m) = \delta(m - e_k), \qquad \text{s.t. } k = \arg\max_k \vec{\mu}_m(m_k = 1)\overleftarrow{\mu}_m(m_k = 1), \tag{23}$$

where e_k represents the k-th Euclidean standard basis vector. Figure 5b visualizes this constraint by the encircled δ on the edge corresponding to the variable m. This form constraint will effectively interrupt the flow of the messages μ_m and instead propagate the computed marginal distribution $q(m)$ back to the connected nodes (Theorem 3, [31]). As a result $q(m)$ will be substituted for $\vec{\mu}_m(m)$ in the message $\vec{\mu}_{s_j}(s_j)$ in Table 1, which performs a selection of the incoming messages for the outgoing message as $\vec{\mu}_{s_j}(s_j) = \vec{\mu}_{s_j|m_k=1}(s_j)$.

5.3. Bayesian Model Combination

Contrary to what some consider its naming to imply, BMA does not find the best possible weighted set of models that explains the data and is therefore often subject to misinterpretation [21]. Instead, it performs a soft selection of the most probable model from the set of candidate models [7,21]. With infinite data, BMA converges to the single best model of the group of possible models [15]. In the case that the true model is inside

the subset of models to evaluate, this will correctly identify the true model. However, often, the true underlying model is not within this subset and, therefore, a suboptimal model is selected. In this case, there might actually exist a specific weighted combination of models that represents the observed data better in terms of model evidence than the single best model [21].

Bayesian model combination (BMC) [7] was introduced to find the best possible weighted set of models, whilst retaining uncertainty over this weighting. The founding work of [7] presents two approaches for BMC: (1) by performing an extensive search over a discretized subspace of model weightings, and (2) by sampling from a Dirichlet distribution that extends the regular categorical model prior. Here, we illustrate the latter approach using a Dirichlet prior on π because inference in this model can be executed efficiently using message passing.

Contrary to the previous subsection, every (set of) observation(s) is now assumed to be modeled by a distinct model m_n from the set of candidate models, where n indexes the observation. Each variable m_n comprises a 1-of-K binary vector with elements $m_{nk} \in \{0, 1\}$ constrained by $\sum_{k=1}^{K} m_{nk} = 1$. We specify the prior distribution

$$p(m_n \mid \pi) = \mathrm{Cat}(m_n \mid \pi), \tag{24}$$

where the event probabilities π now appears as a random variable, which is modeled by

$$p(\pi) = \mathrm{Dir}(\pi \mid \alpha), \tag{25}$$

where α is the concentration parameters. Intuitively, the variable π is shared among all observations, whereas m_n is specific to a single observation as shown in Figure 5c.

Probabilistic Inference for Bayesian Model Combination

The exact inference in this model is intractable because the posterior over π resembles a mixture of Dirichlet distributions with a number of components that scales exponentially with the number of observations. As a result, previous works in the literature presented approximate algorithms for performing probabilistic inference in this model, such as sampling [7]. Here, we present two alternative approaches for performing approximate inference in this model.

The first approach concerns constraining the posterior distributions over m_n to be Kronecker delta functions $\delta(\cdot)$, similar to that shown in Section 5.2 as

$$q(m_n) = \delta(m_n - e_k), \qquad \text{s.t. } k = \arg\max_k \vec{\mu}_{m_n}(m_{nk} = 1)\overleftarrow{\mu}_{m_n}(m_{nk} = 1). \tag{26}$$

Here, we choose the approximate posterior $q(m_n)$ to be centered around the MAP estimate of m_n; however, alternative centers can also be chosen, for example, by sampling from $\vec{\mu}_{m_n}(m_{nk} = 1)\overleftarrow{\mu}_{m_n}(m_{nk} = 1)$. Using this constraint, the backward message $\overleftarrow{\mu}_{\pi}(\pi)$ towards π can be computed analytically (Appendix A.5, [47]). Batch or offline processing can be performed by an iterative message-passing procedure, similar to variational message passing [30,34,40], which requires the initialization of messages $\vec{\mu}_{m_n}(m_n)$ or marginals $q(m_n)$ in order to break circular dependencies between messages and marginals in the model. However, this approach also lends itself toward an online setting with streaming observations. In the online setting, however, the results are heavily influenced by prior $p(\pi)$ if chosen uninformatively as we detail in Section 6.1. In Section 6.1, we also describe an approach to cope with this initialization problem.

An alternative approach to performing approximate inference in an offline manner is obtained by variational message passing [30,34,40]. The true posterior distribution $p(\pi, m_1, \ldots, m_N \mid D)$ is, in this case, approximated by the variational posterior distribution $q(\pi, m_1, \ldots, m_N)$, being subject to naive mean-field factorization as

$$p(\pi, m_1, \ldots, m_N \mid D) \approx q(\pi, m_1, \ldots, m_N) = q(\pi) \prod_{n=1}^{N} q(m_n), \tag{27}$$

where the individual variational distributions are constrained to have the functional forms

$$q(\pi) = \mathrm{Dir}(\pi \mid \tilde{\alpha}), \tag{28a}$$

$$q(m_n) = \mathrm{Cat}(m_n \mid \tilde{\pi}_n), \tag{28b}$$

where the $\tilde{}$ accent is used to indicate the parameters of the variational posterior distributions. Variational message passing minimizes the variational free energy by iterating the computation of variational messages and posteriors until convergence. The corresponding variational message-passing update rules are derived in (Appendix A.5, [47]).

6. Experiments

In this section, a set of experiments is presented for the previously presented message passing-based model comparison techniques. Section 6.1 verifies the basic operations of the inference procedures for model averaging, selection and combination for data generated from a known mixture distribution. In Section 6.2, the model comparison approaches are validated on application-based examples.

All experiments were performed using the scientific programming language `Julia` [48] with the state-of-the-art probabilistic programming package `RxInfer.jl` [9]. The mixture node specified in Section 4.2 was integrated in its message-passing engine `ReactiveMP.jl` [49,50]. Aside from the results presented in the upcoming subsections, interactive `Pluto.jl` notebooks are available online (all experiments are publicly available at https://github.com/biaslab/AutomatingModelComparison, (accessed on 9 June 2023)), allowing the reader to change hyperparameters in real time.

6.1. Verification Experiments

For verification of the mixture node in Table 1, $N = \{1, 5, 10, 100, 1000\}$ observations y_n are generated from the mixture distribution

$$p(y_n) = 0.2\,\mathcal{N}(y_n \mid -3, 1+\sigma^2) + 0.5\,\mathcal{N}(y_n \mid 0, 1+\sigma^2) + 0.3\,\mathcal{N}(y_n \mid 4, 1+\sigma^2), \tag{29}$$

where $\mathcal{N}(y_n \mid \mu, \sigma^2)$ represents a normal distribution with mean μ and variance σ^2. σ^2 represents the additional observation noise variance. For the obtained data, we construct the probabilistic model

$$p(x_n \mid m) = \mathcal{N}(x_n \mid -3, 1)^{m_1}\mathcal{N}(x_n \mid 0, 1)^{m_2}\mathcal{N}(x_n \mid 4, 1)^{m_3}, \tag{30a}$$

$$p(y_n \mid x_n) = \mathcal{N}(y_n \mid x_n, \sigma^2), \tag{30b}$$

which is completed by the structures imposed on m as introduced in Section 5. Depending on the comparison method as outlined in Sections 5.1–5.3, we add an uninformative categorical prior on z or an uninformative Dirichlet prior on the event probabilities π that model z. The aim is to infer the marginal (approximate) posterior distributions over component assignment variable z, for model averaging and selection, and over event probabilities π, for model combination.

The Bayesian model combination preliminary experiments show that the results relied significantly on the initial prior $p(\pi)$ in the online setting. Choosing this term to be uninformative, i.e., $\alpha_k << 1 \,\forall\, k$ and $\alpha_i = \alpha_j \,\forall\, i, j$, led to a posterior distribution which became dominated by the inferred cluster assignment of the first observation. As a result, the predictive class probability approached a delta distribution, centered around the class label of the first observation, leading to all consecutive observations being assigned to the same cluster. This observation is as expected, as the concentration parameters $\tilde{\alpha}$ of the posterior distribution $q(\pi)$ after the first model assignment $m_{1k} = 1$ were updated as $\tilde{\alpha} = \alpha + m_1$, with prior concentration parameters α. When the entries of α are small, this update will have a significant effect of the updated prior distribution over π and consecutively over the prior belief over the model assignment $\vec{\mu}_{m_n}(m_n) = \mathrm{Cat}\left(m_n \mid \alpha / \sum_{k=1}^{K} \alpha\right)$. To remedy this undesirable behavior, the prior $p(\pi)$ was chosen to prefer uniformly distributed class

labels, i.e., $\alpha_k >> 1 \, \forall \, k$ and $\alpha_i = \alpha_j \, \forall \, i, j$. Although this prior yields the same forward message $\vec{\mu}_{m_n}(m_n)$, consecutive forward messages will be less affected by the selected models m_n. After the inference procedure was completed, the informativeness of this prior was removed using Bayesian model reduction [43,44], where the approximate posterior over π was recomputed based on an alternative uninformative prior.

Figure 6 shows the inferred posterior distributions of z or the predictive distributions for z obtained from the posterior distributions $q(\pi)$, for an observation noise variance $\sigma^2 = 5$. From the results, it can be observed that Bayesian model averaging converges with increasing data to a single cluster as expected. This selected cluster corresponds to the cluster inferred by Bayesian model selection, which also corresponds to the cluster with the highest mixing weight in (29). Contrary to Bayesian model selection, the alternative event probabilities obtained with Bayesian model averaging are non-zero. Both Bayesian model combination approaches do not converge to a single cluster assignment as expected. Instead, they better recover the data-generating mixing weights in the data-generating distribution. It can be seen that the variational approach to model combination is better capable of retrieving the original mixing weights, despite the high noise variance of $\sigma^2 = 5$. The online model combination approach is less capable of retrieving the original mixing weights. This is also as expected since the online approach performs an approximate filtering procedure, contrary to the approximate smoothing procedure of the variational approach. For smaller values of the noise variance, we observe in our experiments that the online model combination strategy approaches the variational strategy.

6.2. Validation Experiments

Aside from verifying the correctness of the message-passing implementations of Section 5 using the mixture node, this section further illustrates its usefulness in a set of validation experiments, covering real-world problems.

6.2.1. Mixed Models

In order to illustrate an application of the mixture node from Table 1, we show how it can be used in a mixed model, where it connects continuous to discrete variables. Consider the hypothetical situation, where we wish to fit a mixture with fixed components but unknown mixing coefficients to some set of observations. To highlight the generality of the mixture node, the mixture components are chosen to reflect shifted product distributions, where the possible shifts are limited to a discrete set of values. The assumed probabilistic model of a single observation y is given by

$$p(a) = \mathcal{N}(a \,|\, 0.5, 1), \tag{31a}$$

$$p(b) = \mathcal{N}(b \,|\, 0, 1), \tag{31b}$$

$$p(c \mid z) = \delta(c + 0.2)^{z_1} \delta(c + 1.8)^{z_2} \delta(c - 0.9)^{z_3}, \tag{31c}$$

$$p(z) = \mathrm{Cat}(z \,|\, 1_3/3), \tag{31d}$$

$$p(y \,|\, a, b, c) = \delta(y - (ab + c)). \tag{31e}$$

The variables a and b are latent variables defining the product distribution. c specifies the shift introduced on this distribution, which is picked by the selector variable z, comprising a 1-of-3 binary vector with elements $z_k \in \{0, 1\}$ constrained by $\sum_{k=1}^{3} z_k = 1$. 1_K denotes a vector of ones of length K. The goal is to infer the posterior distribution of z and to, therefore, fit this exotic mixture model to some set of data.

We perform offline probabilistic inference in this model using Bayesian model averaging and Bayesian model combination. For the latter approach, we extend the prior on z with a Dirichlet distribution following Section 5.3 and by assuming a variational mean-field factorization. The shifted product distributions do not yield tractable closed-form messages; therefore, these distributions are approximated following [51]. Figure 7 shows the obtained data fit on a data set of 1500 observations drawn from a standard normal distribution. This distribution does not reflect the used model in (31) on purpose to illustrate its behavior

when the true underlying model is not one of the components. As expected, model averaging converges to the most dominant component, whereas model combination attempts to improve the fit by combining the different components with fixed shifts.

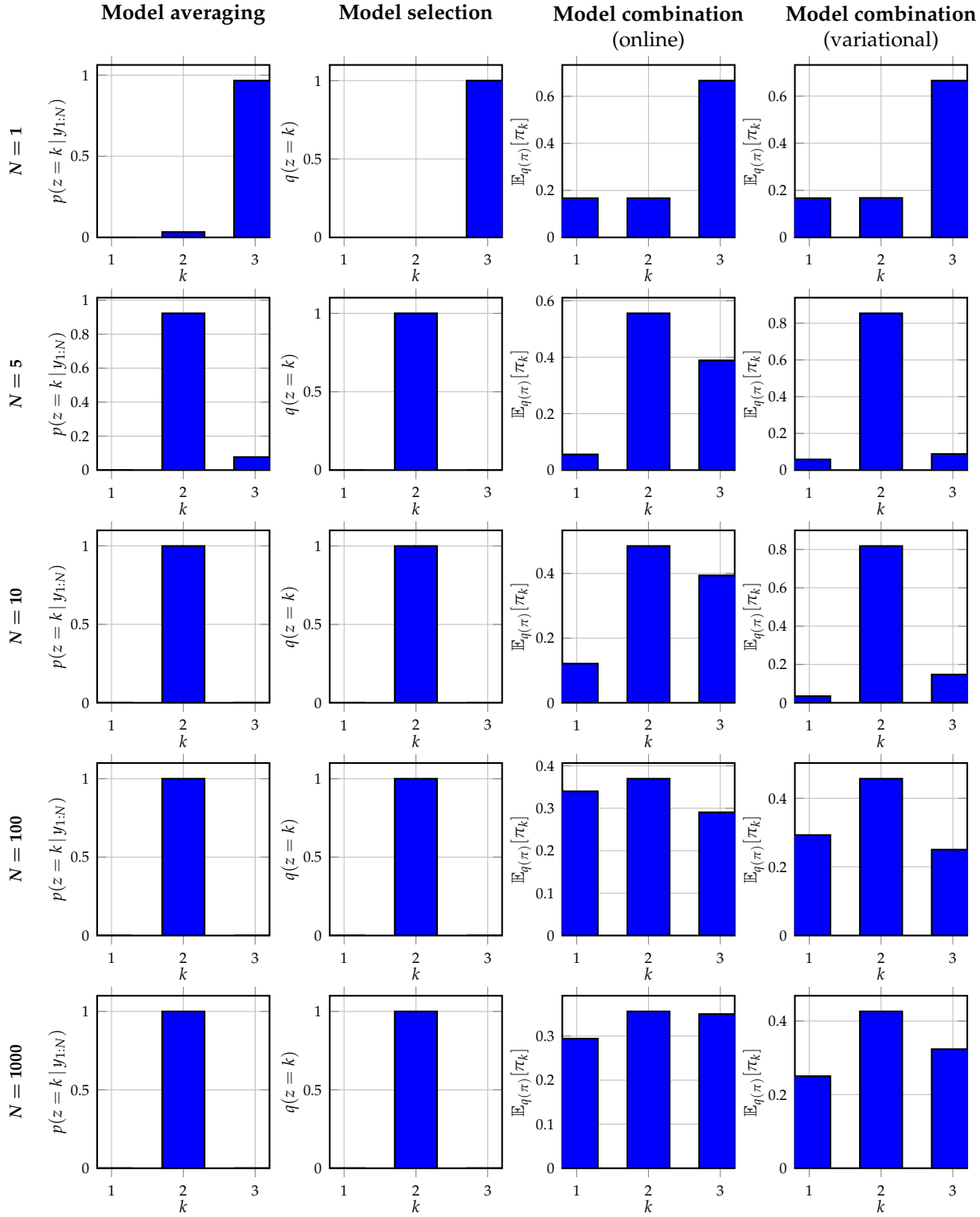


Figure 6. Visualization of the verification experiments as specified in Section 6.1. The individual plots show the (predictive) posterior distributions for the assignment variable in (29) for $N = \{1, 5, 10, 100, 1000\}$ observations as computed using the different methods outlined in Sections 5.1–5.3.

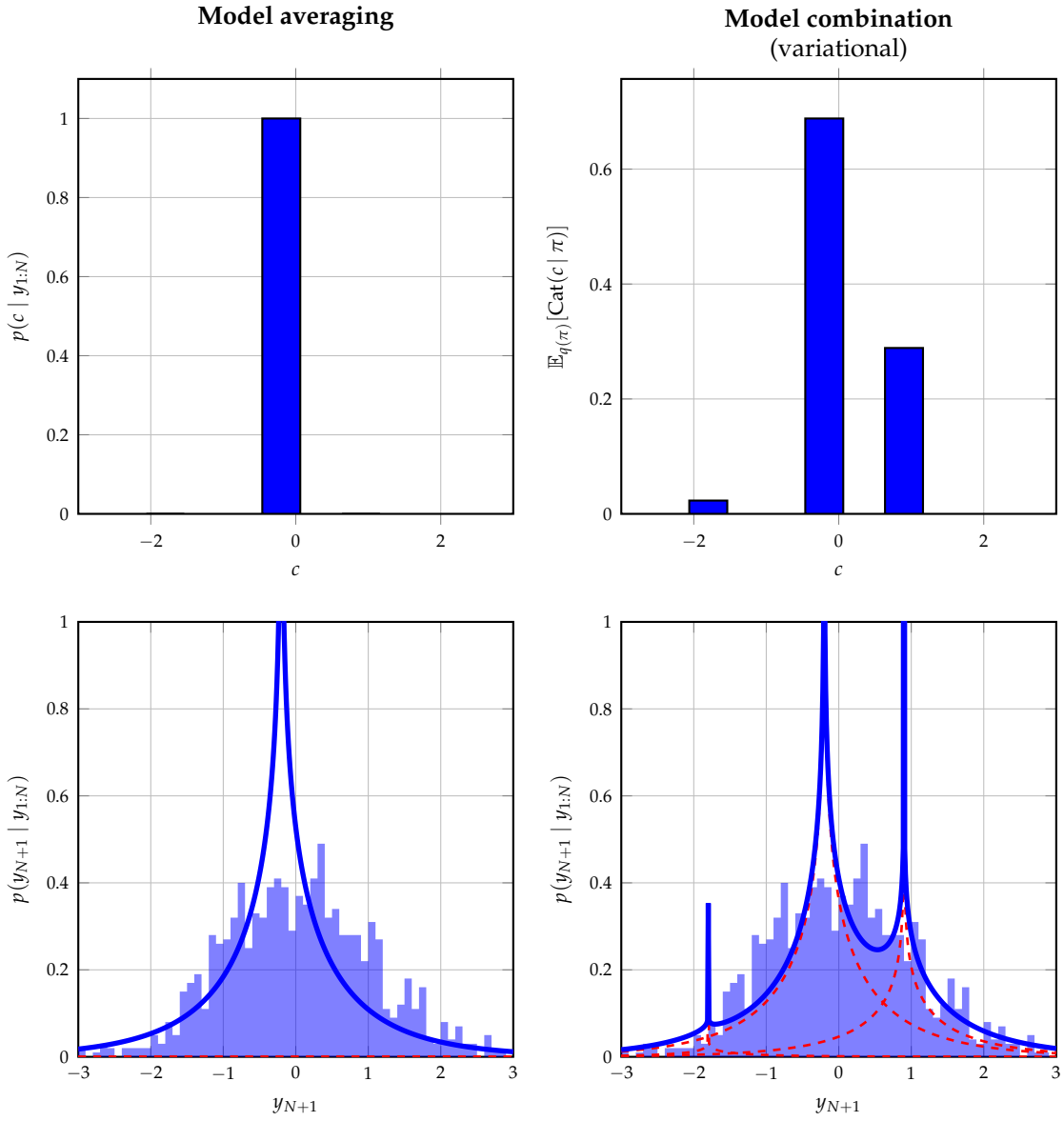


Figure 7. Inference results of the mixed model as described in Section 6.2.1. The inference procedure is performed by (**left**) Bayesian model averaging and (**right**) Bayesian model combination under a variational mean-field factorization. (**top**) The posterior estimate for the shift c. (**bottom**) The predictive posterior distribution for new observations in blue with underlying components in red.

6.2.2. Voice Activity Detection

In this section, we illustrate a message-passing approach to voice activity detection in speech that is corrupted by additive white Gaussian noise using the mixture node from Table 1. We model speech signal s_t as a first-order auto-regressive process as

$$p(s_t \mid s_{t-1}) = \mathcal{N}(s_t \mid \rho s_{t-1}, \sigma^2), \tag{32}$$

with auto-regressive parameter ρ and process noise variance σ^2. The absence of speech is modeled by independent and identically distributed variables n_t, which are enforced to be close to 0 as

$$p(n_t) = \mathcal{N}(n_t \mid 0, 0.01). \tag{33}$$

We model our observations by the mixture distribution, where we include the corruption from the additive white Gaussian noise as

$$p(y_t \mid s_t, n_t, z_t) = \mathcal{N}(y_t \mid s_t, 0.5)^{z_{t1}} \mathcal{N}(y_t \mid n_t, 0.5)^{z_{t2}}. \tag{34}$$

Here, z_t indicates the voice activity of the observed signal as a 1-of-2 binary vector with elements $z_{tk} \in \{0, 1\}$ constrained by $\sum_{k=1}^{2} z_{tk} = 1$. Because periods of speech are often preceded by more speech, we add temporal dynamics to z_t as

$$p(z_t \mid z_{t-1}) = \text{Cat}(z_t \mid \text{T} z_{t-1}), \tag{35}$$

where the transition matrix is specified as $\text{T} = [0.99999, 10^{-5}; 10^{-5}, 0.99999]$.

Figure 8 shows the clean and corrupted audio signals. The audio is sampled with a sampling frequency of 16 kHz. The corrupted signal is used for inferring z_t, which is presented in the bottom plot. Despite the corruption inflicted on the audio signal, this presented simple model is capable of detecting voice effectively as illustrated in the bottom plot of Figure 8.

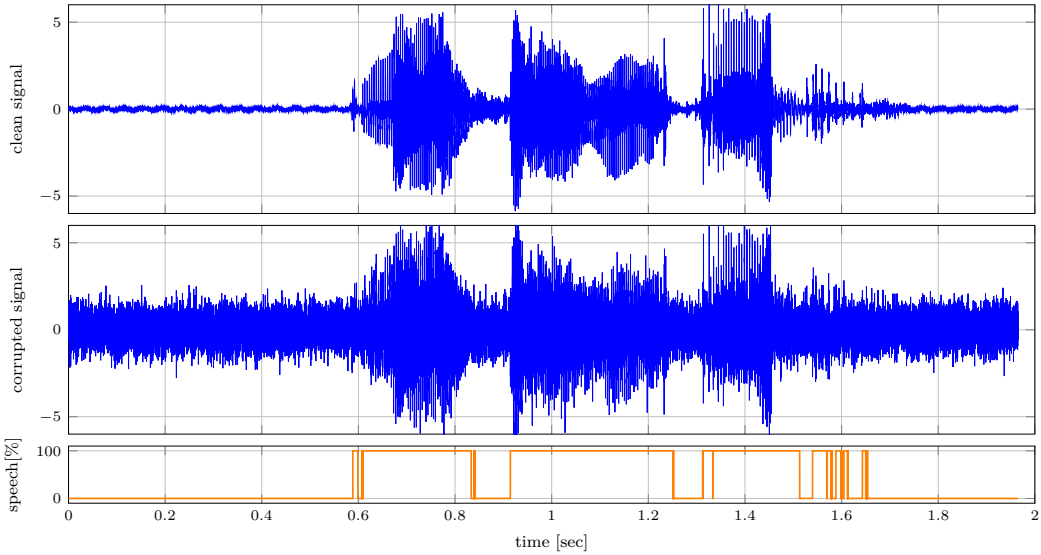


Figure 8. Results of the voice activity detection experiment as specified in Section 6.2.2. The figure shows (**top**) the clean signal, (**middle**) the clean signal corrupted by additive white Gaussian noise and (**bottom**) the inferred speech probability.

7. Discussion

The unifying view between probabilistic inference and model comparison as presented by this paper allows us to leverage the efficient message-passing schemes for both tasks. Interestingly, this view allows for the use of belief propagation [32], variational message passing [30,34,40] and other message passing-based algorithms around the subgraph connected to the model selection variable m. This insight gives rise to a novel class of model comparison algorithms, where the prior on the model selection variable is no longer constrained to be a categorical distribution but where we now can straightforwardly introduce hierarchical and/or temporal dynamics. Furthermore, a consequence of the automatability of the message-passing algorithms is that these model comparison algorithms can easily and efficiently be implemented, without the need for error-prone and time-consuming manual derivations.

Although this paper solely focused on message passing-based probabilistic inference, we envision interesting directions for alternative probabilistic programming packages, such as Stan [14], Pyro [11], Turing [10], UltraNest [12], and PyMC [13]. Currently, only the PyMC framework allows for model comparison through their `compare()` function. However, often these packages allow for estimating the (log-)evidence through sampling, or for computing the evidence lower bound (ELBO), which resembles the negative VFE of (10), which is optimized using stochastic variational inference [52]. An interesting direction of future research would be to use these estimates to construct the factor node $f(m)$ in (14),

with which novel model comparison algorithms can be designed, for example, where the model selection variables becomes observation dependent as in [25].

The presented approach is especially convenient when the model allows for the use of scale factors (Ch.6, [38,39]). In this way, we can efficiently compute the model evidence as shown in [39]. The introduced mixture node in Table 1 consecutively enables a simple model specification as illustrated in the Supplementary Materials source code of our experiments (all experiments are publicly available at https://github.com/biaslab/AutomatingModelComparison, (accessed on 9 June 2023)).

A limitation of the scale factors is that they can only be efficiently computed when the model submits to exact inference [39]. Extensions of the scale factors towards a variational setting would allow the use of the mixture node with a bigger variety of models. If this limitation is resolved, then the introduced approach can be combined with more complicated models, such as, for example, Bayesian neural networks, whose performance is measured by the variational free energy, see, e.g., [53,54]. This provides a novel solution to multi-task machine learning problems, where the number of tasks is not known beforehand [55]. Each Bayesian neural network can then be trained for a specific task, and additional components or networks can be added if appropriate.

The mixture nodes presented in this paper can also be nested on top of each other. As a result, hierarchical mixture models can be realized, which can quickly increase the complexity of the nested model. The question quickly arises as to where to stop. An answer to this question is provided by Bayesian model reduction [43,44]. Bayesian model reduction allows for the efficient computation of the model evidence when parts of the hierarchical model are pruned. This approach allows for the pruning of hierarchical models in an effort to bound the complexity of the entire model.

8. Conclusions

This paper bridges the gap between probabilistic inference for states and parameters, and for model comparison, allowing for the simultaneous automation of both tasks. It is shown that model comparison can be performed by message passing on a graph terminated by a node that captures the performance results of the different submodels as motivated from a variational free energy perspective. In the case where the model submits to exact inference, we can efficiently implement model comparison using our newly proposed mixture node, which leverages the efficiently computed scale factors. Based on this node description, we show how to automate Bayesian model averaging, selection, and combination by changing the (hierarchical) prior and posterior constraints on the selection variable.

Supplementary Materials: Source code and experiments are available at https://github.com/biaslab/AutomatingModelComparison (accessed on 9 June 2023).

Author Contributions: Conceptualization, B.v.E.; methodology, B.v.E., W.W.L.N. and T.v.d.L.; software, B.v.E.; validation, B.v.E.; formal analysis, B.v.E. and T.v.d.L.; investigation, B.v.E., W.W.L.N. and T.v.d.L.; writing—original draft preparation, B.v.E.; writing—review and editing, B.v.E., W.W.L.N., T.v.d.L. and B.d.V.; visualization, B.v.E.; supervision, B.d.V. All authors have read and agreed to the published version of the manuscript.

Funding: This work was partly financed by GN Advanced Science, which is the research department of GN Hearing A/S.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Not applicable.

Acknowledgments: The authors would like to thank the BIASlab team members for various insightful discussions related to this work.

Conflicts of Interest: The authors declare no conflict of interest.

Appendix A. Proofs

Appendix A.1. Proof of Theorem 1

This proof follows a similar recipe as that in Appendix D.3 of [31]. Consider the induced subgraph in Figure A1. The node-local and edge-specific marginals $q_a(s_a)$ and $q_j(s_j)$, respectively, obtained by belief propagation or sum–product message passing as the fixed points of (5), are given by

$$q_a(s_a) = \frac{1}{Z_a} f_a(y = \hat{y}, s_a) \prod_{i \in \mathcal{E}(a)} \vec{\mu}_{s_i}(s_i), \tag{A1a}$$

$$q_j(s_j) = \frac{1}{Z_j} \vec{\mu}_{s_j}(s_j) \overleftarrow{\mu}_{s_j}(s_j), \tag{A1b}$$

as shown in Theorem 1 in [31], with node-local and edge-specific normalization constants

$$Z_a = \int f_a(y = \hat{y}, s_a) \prod_{i \in \mathcal{E}(a)} \vec{\mu}_{s_i}(s_i) \, \mathrm{d}s_a, \tag{A2a}$$

$$Z_j = \int \vec{\mu}_{s_j}(s_j) \overleftarrow{\mu}_{s_j}(s_j) \, \mathrm{d}s_j. \tag{A2b}$$

As shown in Appendix D.3 in [31], the normalization constants are equal to $Z_a = Z_j$. As this holds for all variables $s_j \in s_a$, we can deduce that $Z_i = Z_j \, \forall \, i, j \in \mathcal{E}(a)$ also holds. Similarly, this property remains valid for adjacent nodes, allowing us to write

$$Z_a = Z_j \qquad \text{s.t. } a \in \mathcal{V}, j \in \mathcal{E}. \tag{A3}$$

This property stipulates that the normalization constants of all (joint) marginal distributions are equal if the solutions correspond to the fixed points of (5) under sum–product message passing. As the normalization constant equals the model evidence, we can compute the model evidence on any edge and around any node in our graph. The fixed-point assumption is only violated in the case of the cyclic graph, where we perform loopy belief propagation [33].

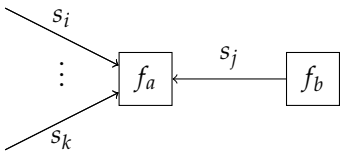


Figure A1. Visualization of a subgraph.

Appendix A.2. Proof of Theorem 2

The arbitrary outward message $\vec{\mu}_{s_k}(s_k)$ from the node $f_a(y = \hat{y}, s_a)$ can be computed using the sum–product message-passing update rule in (5). Substitution of the definition of $\vec{\mu}_{s_j}(s_j)$ of (A9) in this update rules yields

$$\begin{aligned}
\vec{\mu}_{s_k}(s_k) &= \int \vec{\mu}_{s_j}(s_j) \prod_{\substack{i \neq j \\ i \neq k}} \vec{\mu}_{s_i}(s_i) f_a(y = \hat{Y}, s_a) \, \mathrm{d}s_{a \setminus k}, \\
&= \int \left(\sum_{k=1}^{K} \vec{\mu}_m(m_k = 1) \vec{\mu}_{s_j | m_k = 1}(s_j) \right) \prod_{\substack{i \neq j \\ i \neq k}} \vec{\mu}_{s_i}(s_i) f_a(y = \hat{y}, s_a) \, \mathrm{d}s_{a \setminus k}, \\
&= \sum_{k=1}^{K} \vec{\mu}_m(m_k = 1) \int \vec{\mu}_{s_j | m_k = 1}(s_j) \prod_{\substack{i \neq j \\ i \neq k}} \vec{\mu}_{s_i}(s_i) f_a(y = \hat{y}, s_a) \, \mathrm{d}s_{a \setminus k}, \\
&= \sum_{k=1}^{K} \vec{\mu}_m(m_k = 1) \vec{\mu}_{s_k | m_k = 1}(s_k),
\end{aligned} \tag{A4}$$

where we identify the same form as compared to $\vec{\mu}_{s_j}(s_j)$.

Appendix B. Derivations

Appendix B.1. Derivation of Variational Free Energy Decomposition for Mixture Models

The following is the derivation of (12):

$$\begin{aligned}
\mathrm{F}[q] &= \mathbb{E}_{q(s,m)}\left[\ln \frac{q(s,m)}{p(y=\hat{y},s,m)}\right] \\
&= \mathbb{E}_{q(s,m)}\left[\ln \frac{q(m)\prod_{k=1}^{K} q(s_k \mid m_k=1)^{m_k}}{p(m)\prod_{k=1}^{K} p(y_k=\hat{y}_k,s_k \mid m_k=1)^{m_k}}\right] \\
&= \mathbb{E}_{q(s,m)}\left[\ln \frac{q(m)}{p(m)}\right] + \mathbb{E}_{q(s,m)}\left[\sum_{k=1}^{K} m_k \ln\left(\frac{q(s_k \mid m_k=1)}{p(y_k=\hat{y}_k,s_k \mid m_k=1)}\right)\right] \\
&= \mathbb{E}_{q(s,m)}\left[\ln \frac{q(m)}{p(m)}\right] + \mathbb{E}_{q(s,m)}\left[\prod_{k=1}^{K}\left(\ln \frac{q(s_k \mid m_k=1)}{p(y_k=\hat{y}_k,s_k \mid m_k=1)}\right)^{m_k}\right] \\
&= \mathbb{E}_{q(m)}\left[\ln \frac{q(m)}{p(m)}\right] + \mathbb{E}_{q(m)}\left[\prod_{k=1}^{K} \mathbb{E}_{q(s_k \mid m_k=1)}\left[\ln \frac{q(s_k \mid m_k=1)}{p(y_k=\hat{y}_k,s_k \mid m_k=1)}\right]^{m_k}\right] \\
&= \mathbb{E}_{q(m)}\left[\ln \frac{q(m)}{p(m)}\right] + \mathbb{E}_{q(m)}\left[\prod_{k=1}^{K} \left(\mathrm{F}_k[q]\right)^{m_k}\right]
\end{aligned}$$

The step from the third to fourth line is the result of the variable m being one-hot coded. As a result, only a single m_k equals 1, and all others are constrained to be 0. Therefore, we can obtain the identity

$$\prod_{k=1}^{K} a_k^{m_k} = \sum_{k=1}^{K} m_k a_k, \qquad \text{s.t. } \sum_{k=1}^{K} m_k = 1 \text{ and } m_k \in \{0,1\}\,\forall\, k$$

which one might recognize as the different representations of the probability mass function of a categorical distribution.

The factor $f_m(m)$ in (14) can be derived from the above result as

$$\begin{aligned}
\mathbb{E}_{q(m)}\left[\prod_{k=1}^{K} \left(\mathrm{F}_k[q]\right)^{m_k}\right] &= \mathbb{E}_{q(m)}\left[-\ln\left(\exp\left(-\prod_{k=1}^{K} \left(\mathrm{F}_k[q]\right)^{m_k}\right)\right)\right] \\
&= \mathbb{E}_{q(m)}\left[\ln \frac{1}{\exp\left(-\prod_{k=1}^{K} (\mathrm{F}_k[q])^{m_k}\right)}\right] \\
&= \mathbb{E}_{q(m)}\left[\ln \frac{1}{\prod_{k=1}^{K} \exp(-\mathrm{F}_k[q])^{m_k}}\right]
\end{aligned}$$

The step from the second to third line again uses the fact that the variable m is one-hot coded. As a result, we can obtain the identity

$$\exp\left(-\prod_{k=1}^{K} \alpha_k^{m_k}\right) = \prod_{k=1}^{K} \exp(-\alpha_k)^{m_k}. \qquad \text{s.t. } \sum_{k=1}^{K} m_k = 1 \text{ and } m_k \in \{0,1\}\,\forall\, k$$

One can validate this expression by considering a realization of m and expanding the identity.

Appendix B.2. Derivation of Message $\overleftarrow{\mu}_m(m)$

Consider the variable $s_j \in s_o$ in the context of Table 1, where we wish to compute the backwards message $\overleftarrow{\mu}_m(m)$ towards m. As shown in [45], the posterior $q(m)$ can be obtained through the functional optimization of (12). Following the approach stipulated in [31], the solution for $q(m)$ follows the form of (6) as

$$q(m) \propto \vec{\mu}_m(m)\overleftarrow{\mu}_m(m) = \vec{\mu}_m(m)f(m), \tag{A5}$$

where $\vec{\mu}_m(m)$ denotes the message towards m, not originating from $f(m)$.

Under the assumptions of acyclicity and tractability,

$$\exp(-\mathrm{F}_k[q]) = Z_k = \int \vec{\mu}_{s_j|m_k=1}(s_j)\overleftarrow{\mu}_{s_j|m_k=1}(s_j)\,\mathrm{d}s_j \tag{A6}$$

holds, from which we obtain the message

$$\overleftarrow{\mu}_m(m) = f(m) = \prod_{k=1}^{K}\left(\int \vec{\mu}_{s_j|m_k=1}(s_j)\overleftarrow{\mu}_{s_j|m_k=1}(s_j)\,\mathrm{d}s_j\right)^{m_k}, \tag{A7}$$

where the substitution of (A6) into the definition of $f(m)$ in (14) yields the message $\overleftarrow{\mu}_m(m)$ in Table 1. Here, we leverage scale factors inside the mixture node to compute the normalization constants of the different models.

Appendix B.3. Derivation of Message $\vec{\mu}_{s_j}(s_j)$

Consider again the variable $s_j \in s_o$, whose posterior distribution $q(s_j)$ we now wish to compute. The substitution of $q(s_j \mid m_k = 1)$ in (6) and $\overleftarrow{\mu}_m(m)$ in (A7) into (16) yields

$$\begin{aligned}
q(s_j) &= \mathbb{E}_{q(m)}\left[\prod_{k=1}^{K} q(s_j \mid m_k)^{m_k}\right], \\
&= \sum_{k=1}^{K} q(m_k)q(s_j \mid m_k), \\
&= \sum_{k=1}^{K} \frac{\vec{\mu}_m(m_k=1)\overleftarrow{\mu}_m(m_k=1)}{\sum_{k=1}^{K}\vec{\mu}_m(m_k=1)\overleftarrow{\mu}_m(m_k=1)} \frac{\vec{\mu}_{s_j|m_k}(s_j)\overleftarrow{\mu}_{s_j|m_k=1}(s_j)}{\int \vec{\mu}_{s_j|m_k=1}(s_j)\overleftarrow{\mu}_{s_j|m_k=1}(s_j)\,\mathrm{d}s_j}, \\
&= \frac{1}{\sum_{k=1}^{K}\vec{\mu}_m(m_k=1)\overleftarrow{\mu}_m(m_k=1)} \sum_{k=1}^{K} \vec{\mu}_m(m_k=1)\overleftarrow{\mu}_m(m_k=1)\frac{\vec{\mu}_{s_j|m_k=1}(s_j)\overleftarrow{\mu}_{s_j|m_k=1}(s_j)}{\overleftarrow{\mu}_m(m_k=1)}, \\
&= \frac{1}{Z_m}\sum_{k=1}^{K} \vec{\mu}_m(m_k=1)\vec{\mu}_{s_j|m_k=1}(s_j)\overleftarrow{\mu}_{s_j|m_k=1}(s_j),
\end{aligned} \tag{A8}$$

where $Z_m = \sum_{k=1}^{K}\vec{\mu}_m(m_k=1)\overleftarrow{\mu}_m(m_k=1)$. Furthermore, as a result of the assumption $s_j \in s_o$ being located in the overlapping model section, one of the sum–product messages towards s_j is independent of the model m_k, i.e., either $\vec{\mu}_{s_j|m_k=1}(s_j) = \vec{\mu}_{s_j}(s_j)$ or $\overleftarrow{\mu}_{s_j|m_k=1}(s_j) = \overleftarrow{\mu}_{s_j}(s_j)$ holds. In the situation sketched in Figure 3, we assume the latter. In this case, we obtain the identity

$$Z_m q(s_j) = \underbrace{\left(\sum_{k=1}^{K}\vec{\mu}_m(m_k=1)\vec{\mu}_{s_j|m_k=1}(s_j)\right)}_{\vec{\mu}_{s_j}(s_j)}\overleftarrow{\mu}_{s_j}(s_j), \tag{A9}$$

from which the message $\vec{\mu}_{s_j}(s_j)$ in Table 1 can be identified.

References

1. Box, G.E.P. Robustness in the Strategy of Scientific Model Building. In *Robustness in Statistics*; Launer, R.L., Wilkinson, G.N., Eds.; Academic Press: Cambridge, MA, USA, 1979; pp. 201–236.
2. Blei, D.M. Build, Compute, Critique, Repeat: Data Analysis with Latent Variable Models. *Annu. Rev. Stat. Its Appl.* **2014**, *1*, 203–232. [CrossRef]
3. Box, G.E.P. Science and Statistics. *J. Am. Stat. Assoc.* **1976**, *71*, 791–799. [CrossRef]
4. MacKay, D.J.C. *Information Theory, Inference, and Learning Algorithms*; Cambridge University Press: Cambridge, UK, 2003.
5. Koller, D.; Friedman, N. *Probabilistic Graphical Models: Principles and Techniques*; Adaptive computation and machine learning; MIT Press: Cambridge, MA, USA, 2009.
6. Hoeting, J.A.; Madigan, D.; Raftery, A.E.; Volinsky, C.T. Bayesian Model Averaging: A Tutorial. *Stat. Sci.* **1999**, *14*, 382–401.
7. Monteith, K.; Carroll, J.L.; Seppi, K.; Martinez, T. Turning Bayesian model averaging into Bayesian model combination. In Proceedings of the 2011 International Joint Conference on Neural Networks, San Jose, CA, USA, 31 July–5 August 2011; pp. 2657–2663. [CrossRef]
8. Cox, M.; van de Laar, T.; de Vries, B. A factor graph approach to automated design of Bayesian signal processing algorithms. *Int. J. Approx. Reason.* **2019**, *104*, 185–204. [CrossRef]
9. Bagaev, D.; Podusenko, A.; De Vries, B. RxInfer: A Julia package for reactive real-timeBayesian inference. *J. Open Source Softw.* **2023**, *8*, 5161. [CrossRef]
10. Ge, H.; Xu, K.; Ghahramani, Z. Turing: A Language for Flexible Probabilistic Inference. In Proceedings of the Twenty-First International Conference on Artificial Intelligence and Statistics, PMLR, Playa Blanca, Spain, 9–11 April 2018; pp. 1682–1690.
11. Bingham, E.; Chen, J.P.; Jankowiak, M.; Obermeyer, F.; Pradhan, N.; Karaletsos, T.; Singh, R.; Szerlip, P.; Horsfall, P.; Goodman, N.D. Pyro: Deep Universal Probabilistic Programming. *J. Mach. Learn. Res.* **2018**, *20*, 973–978. [CrossRef]
12. Buchner, J. UltraNest—A robust, general purpose Bayesian inference engine. *arXiv* **2021**. [CrossRef]
13. Salvatier, J.; Wiecki, T.; Fonnesbeck, C. Probabilistic Programming in Python using PyMC. *arXiv* **2015**, . arXiv.1507.08050. [CrossRef]
14. Carpenter, B.; Gelman, A.; Hoffman, M.D.; Lee, D.; Goodrich, B.; Betancourt, M.; Brubaker, M.; Guo, J.; Li, P.; Riddell, A. Stan: A Probabilistic Programming Language. *J. Stat. Softw.* **2017**, *76*, 1–32. [CrossRef]
15. Kamary, K.; Mengersen, K.; Robert, C.P.; Rousseau, J. Testing hypotheses via a mixture estimation model. *arXiv* **2018**, arXiv:1412.2044.
16. Minka, T.; Winn, J. Gates. In *Advances in Neural Information Processing Systems 21*; Curran Associates, Inc.: Red Hook, NY, USA, 2009; pp. 1073–1080.
17. Fragoso, T.M.; Neto, F.L. Bayesian model averaging: A systematic review and conceptual classification. *Int. Stat. Rev.* **2018**, *86*, 1–28. [CrossRef]
18. Stephan, K.E.; Penny, W.D.; Daunizeau, J.; Moran, R.J.; Friston, K.J. Bayesian model selection for group studies. *NeuroImage* **2009**, *46*, 1004–1017. [CrossRef]
19. Rigoux, L.; Stephan, K.; Friston, K.; Daunizeau, J. Bayesian model selection for group studies—Revisited. *NeuroImage* **2014**, *84*, 971–985. [CrossRef]
20. Schmitt, M.; Radev, S.T.; Bürkner, P.C. Meta-Uncertainty in Bayesian Model Comparison. *arXiv* **2023**, arXiv:2210.07278.
21. Minka, T.P. Bayesian Model Averaging Is Not Model Combination. 2000. Available online: http://www.stat.cmu.edu/minka/papers/bma.html (accessed on 9 June 2023).
22. Keller, M.; Kamary, K. Bayesian model averaging via mixture model estimation. *arXiv* **2018**, arXiv:1711.10016.
23. Yao, Y.; Vehtari, A.; Simpson, D.; Gelman, A. Using Stacking to Average Bayesian Predictive Distributions (with Discussion). *Bayesian Anal.* **2018**, *13*, 917–1007. [CrossRef]
24. Domingos, P. Bayesian Averaging of Classifiers and the Overfitting Problem. In Proceedings of the Seventeenth International Conference on Machine Learning, ICML '00, San Francisco, CA, USA, 29 June–2 July 2000; Morgan Kaufmann Publishers Inc.: San Francisco, CA, USA, 2000; pp. 223–230.
25. Yao, Y.; Pirš, G.; Vehtari, A.; Gelman, A. Bayesian Hierarchical Stacking: Some Models Are (Somewhere) Useful. *Bayesian Anal.* **2022**, *17*, 1043–1071. [CrossRef]
26. Wolpert, D.H. Stacked generalization. *Neural Netw.* **1992**, *5*, 241–259. [CrossRef]
27. Loeliger, H.A. An introduction to factor graphs. *IEEE Signal Process. Mag.* **2004**, *21*, 28–41. . [CrossRef]
28. Loeliger, H.A.; Dauwels, J.; Hu, J.; Korl, S.; Ping, L.; Kschischang, F.R. The Factor Graph Approach to Model-Based Signal Processing. *Proc. IEEE* **2007**, *95*, 1295–1322. [CrossRef]
29. Kschischang, F.; Frey, B.; Loeliger, H.A. Factor graphs and the sum-product algorithm. *IEEE Trans. Inf. Theory* **2001**, *47*, 498–519. [CrossRef]
30. Dauwels, J. On Variational Message Passing on Factor Graphs. In Proceedings of the 2007 IEEE International Symposium on Information Theory, Nice, France, 24–29 June 2007; pp. 2546–2550. [CrossRef]
31. Şenöz, I.; van de Laar, T.; Bagaev, D.; de Vries, B. Variational Message Passing and Local Constraint Manipulation in Factor Graphs. *Entropy* **2021**, *23*, 807. [CrossRef]
32. Pearl, J. Reverend Bayes on Inference Engines: A Distributed Hierarchical Approach. In Proceedings of the American Association for Artificial Intelligence National Conference on AI, Pittsburgh, PA, USA, 18–20 August 1982; pp. 133–136.

33. Murphy, K.; Weiss, Y.; Jordan, M.I. Loopy Belief Propagation for Approximate Inference: An Empirical Study. In Proceedings of the Fifteenth Conference on Uncertainty in Artificial Intelligence, Stockholm, Sweden, 30 July–1 August 1999.
34. Winn, J.M. Variational Message Passing and Its Applications. Ph.D. Thesis, University of Cambridge, Cambridge, UK, 2004.
35. Dauwels, J.; Korl, S.; Loeliger, H.A. Expectation maximization as message passing. In Proceedings of the International Symposium on Information Theory (ISIT), Adelaide, Australia, 4–9 September 2005; IEEE: Piscataway, NJ, USA, 2005; pp. 583–586. [CrossRef]
36. Minka, T.P. Expectation Propagation for Approximate Bayesian Inference. In Proceedings of the Seventeenth Conference on Uncertainty in Artificial Intelligence, Seattle, WA, USA, 2–5 August 2001; Morgan Kaufmann Publishers Inc.: San Francisco, CA, USA, 2001; pp. 362–369.
37. Yedidia, J.S.; Freeman, W.T.; Weiss, Y. Generalized Belief Propagation. *Adv. Neural Inf. Process. Syst.* **2000**, *13*, 689–695.
38. Reller, C. State-Space Methods in Statistical Signal Processing: New Ideas and Applications. Ph.D. Thesis, ETH Zurich, Zurich, Switzerland, 2013.
39. Nguyen, H.M.; van Erp, B.; Şenöz, İ.; de Vries, B. Efficient Model Evidence Computation in Tree-structured Factor Graphs. In Proceedings of the 2022 IEEE Workshop on Signal Processing Systems (SiPS), Rennes, France, 2–4 November 2022; pp. 1–6. [CrossRef]
40. Winn, J.; Bishop, C.M. Variational Message Passing. *J. Mach. Learn. Res.* **2005**, *6*, 661–694.
41. Forney, G. Codes on graphs: Normal realizations. *IEEE Trans. Inf. Theory* **2001**, *47*, 520–548. [CrossRef]
42. Bishop, C.M. *Pattern Recognition and Machine Learning*; Information science and statistics; Springer: New York, NY, USA, 2006.
43. Friston, K.; Penny, W. Post hoc Bayesian model selection. *NeuroImage* **2011**, *56*, 2089–2099. [CrossRef]
44. Friston, K.; Parr, T.; Zeidman, P. Bayesian model reduction. *arXiv* **2019**, arXiv:1805.07092.
45. Parr, T.; Friston, K.J. Generalised free energy and active inference. *Biol. Cybern.* **2019**, *113*, 495–513. [CrossRef]
46. Murphy, K.P. *Machine Learning: A Probabilistic Perspective*; Adaptive computation and machine learning series; MIT Press: Cambridge, MA, USA, 2012.
47. van de Laar, T. Automated Design of Bayesian Signal Processing Algorithms. Ph.D. Thesis, Technische Universiteit Eindhoven, Eindhoven, The Netherlands, 2019.
48. Bezanson, J.; Edelman, A.; Karpinski, S.; Shah, V.B. Julia: A Fresh Approach to Numerical Computing. *SIAM Rev.* **2017**, *59*, 65–98. [CrossRef]
49. Bagaev, D.; de Vries, B. Reactive Message Passing for Scalable Bayesian Inference. *Sci. Program.* **2023**, *2023*, 6601690. [CrossRef]
50. Bagaev, D.; van Erp, B.; Podusenko, A.; de Vries, B. ReactiveMP.jl: A Julia package for reactive variational Bayesian inference. *Softw. Impacts* **2022**, *12*, 100299. [CrossRef]
51. Cui, G.; Yu, X.; Iommelli, S.; Kong, L. Exact Distribution for the Product of Two Correlated Gaussian Random Variables. *IEEE Signal Process. Lett.* **2016**, *23*, 1662–1666. [CrossRef]
52. Hoffman, M.D.; Blei, D.M.; Wang, C.; Paisley, J. Stochastic Variational Inference. *J. Mach. Learn. Res.* **2013**, *14*, 1303–1347.
53. Blundell, C.; Cornebise, J.; Kavukcuoglu, K.; Wierstra, D. Weight Uncertainty in Neural Networks. *arXiv* **2015**, arXiv:1505.05424.
54. Haussmann, M.; Hamprecht, F.A.; Kandemir, M. Sampling-Free Variational Inference of Bayesian Neural Networks by Variance Backpropagation. *arXiv* **2019**. [CrossRef]
55. Ruder, S. An Overview of Multi-Task Learning in Deep Neural Networks. *arXiv* **2017**, arXiv:1706.05098.