Eindhoven University of Technology

MSc thesis

Machine Learning Framework for Bayesian Signal Processing

Author:
Timur Bagautdinov

Supervisor at EE department:
Prof. dr.B. de Vries

Supervisor at CS department:
Dr. M. Pechenizkiy

August 23, 2013

Abstract

In this thesis, we present a Bayesian machine learning framework for signal processing, based on variational message passing. The developed framework and its open-source software implementation allow building generative models using a set of building blocks, and doing inference on them using variational message passing. The capabilities of the framework are being demonstrated on several applications, such as speech denoising and image segmentation.

The main contribution of this work is the newly developed framework that incorporates means for model specification, probabilistic inference and model comparison, and focuses on solving signal processing tasks. The framework is capable of working with multivariate continuous data, and is expandable enough to solve complex real-world problems.

Contents

List of Figures

List of Tables

List of Acronyms

i.i.d. Independent and Identically Distributed

PDF Probability Density Function

FFG Forney Factor Graph

KL Kullback-Leibler

DAG Directed Acyclic Graph

VMP Variational Message Passing

MAP Maximum A-Posteriori

ML Maximum Likelihood

EM Expectation Maximization

EP Expectation Propagation

MCMC Markov Chain Monte Carlo

PSNR Peak Signal-to-Noise Ratio

MSE Mean Squared Error

DFT Discrete Fourier Transform

SNR Signal-to-Noise Ratio

PSD Power Spectral Density

List of Notations

$x \sim p(\boldsymbol{\theta})$	A random variable x has distribution p parameterized with $\boldsymbol{\theta}$
$\langle x \rangle$	The expectation of a random variable x
$\langle x \rangle_{p(x)}$	The expectation of a random variable x w.r.t. the distribution $p(x)$
x	Scalar variable x
$\boldsymbol{x}, \boldsymbol{X}$	Non-scalar variable (vector, matrix, set)
$\boldsymbol{V}$	A set of observed variables
$\boldsymbol{H}$	A set of hidden variables
$\boldsymbol{A}^T$	Transpose of a matrix $\boldsymbol{A}$
$\boldsymbol{A}^{-1}$	Inverse of a matrix $\boldsymbol{A}$
$\lvert\boldsymbol{A}\rvert$	Determinant of a matrix $\lvert\boldsymbol{A}\rvert$

1 Introduction

1.1 Motivation

Generative modeling [1, Ch.1.3] is a methodology for finding patterns and regularities in the observed data. The main goal of a researcher within this methodology is to find a specific structure of the model that would allow explaining the data generation process and, possibly, predicting future observations. The generative process itself is usually considered to consist of a number of unknown sources or, in other words, *hidden variables*, and connections of those variables with each other, as well as with the observed data (or *observed variables*), encode the patterns one is trying to unveil. In terms of probability theory, these connections can be thought of as a *joint distribution* over observed and hidden variables.

It is often the case, that there is already some prior knowledge that one has about the underlying process. For instance, one might already know whether the observed or hidden variables have continuous or discrete distributions, whether the values of those variables are only allowed negative or positive, or which variables are (in)dependent. This knowledge can allow to enforce a certain structure of the joint distribution, to some extent, allowing to incorporate domain expert knowledge into models. However, even in case this kind of prior knowledge is available, the values of the hidden variables still remain unknown. A widely adopted approach to get the estimates of those is known as *expectation-maximization* (EM), and is based on maximization of the likelihood of posterior functions (see Section 2.3.2). The method, however, is only providing *point estimates* of the hidden variables, rather than complete distributions over those, meaning that they are not capable of calculating the variance of those values (which can be quite useful to understand how the value is spread). This method is also quite prone to overfitting, particularly because it does not penalize the model for its complexity [3].

This is why, *fully Bayesian* approaches, in which the values of model parameters themselves are specified as probability distributions, are being widely adopted [1]. Unfortunately, it turns out that for non-trivial models analytically tractable solutions do not exist [3]. Thus, approximation methods are necessary, and one of the most promising class of methods is a deterministic approximation scheme known as variational method, sometimes referred to as *Variational Bayes* [32].

Such approximation methods are rather generic and make Bayesian methods applicable for various target areas, such as data mining [12], image processing [27] and speech processing [8], [5]. However, when applying these methods, authors mostly derive inference equations from scratch, which can be quite tedious from the mathematical point of view and prone to errors. This can be considered as highly non-effective, since the further developments of variational methods [32] showed, that for a large class of models the variational method can be generalized, and the non-trivial part of deriving equations can be omitted.

We believe, that this shortcoming is caused by the fact that there are very few comprehensive Bayesian machine learning frameworks that would be directly applicable to real-world problems, and that would eliminate the necessity of users to derive yet another inference algorithm for any new model they come up with.

1.2 Goal and problem formulation

The main goal of this work is to create a machine learning framework that would provide a *unified* way to describe generative models specifically for signal processing, doing inference on these models, and comparing them.

Signal processing, is, of course, a very broad area, and there are no highly specific requirements that it imposes on the framework. Important aspects that should be, in one way or another, taken into account is that *continuous multivariate* and *time-varying* data are very common in signal processing tasks [31]. A motivational example, which we consider as a case study in this work later on, could be noise reduction for speech enhancement. By itself it is a very active area of research [20], [9], however, a common shortcoming among existing algorithms is that they incorporate prior knowledge about the underlying process quite implicitly. This often leads to heuristical choices for the values of model parameters [9], which, of course, can be non-optimal.

Note, that by *framework* we imply an overall methodology when approaching the problem of interest, that includes the combination of mathematical treatments and notations for:

- model specification, i.e. a (possibly graphical) notation for describing probabilistic models;

- inference on the specified model, that is, reasoning on the specified model;
- model selection, that is, estimating and comparing performance of several models.

Of course, in the context of modern machine learning, it is also extremely important to have a working software implementation of the framework, since it makes it possible to apply the framework with little programming effort to the real-world datasets. It is worth mentioning, that when developing the software, as a by-product, we have created a kind of Domain-Specific Language (DSL) for generative model specification. We do not elaborate on that aspect in this work, since it is not the main focus of the thesis, however, the idea of probabilistic programming and native probabilistic computing is an interesting and highly active area of research too [18].

1.3 Approach

The main idea behind the framework is to provide a set of pre-designed building blocks, and allow doing inference and parameter learning on models defined using these blocks. Such an approach lets users to be involved more in the specification of the model and tuning it to the needs of their application area, rather than in deriving inference equations for each new model they come up with.

We are taking the Bayesian approach based on variational inference [32]. These are discussed in more detail in Section 2.4.1, but what is important is that Bayesian methods themselves provide a unified formal way for model specification, namely, using random variables and probability distributions. Variational methods, in their turn, make inference process tractable, and, moreover, provide a build-in way for model comparison (see Section 2.6). Moreover, these are convenient ways of graphical representation of Bayesian probabilistic models, such as *factor graphs* (see Section 2.2.2), which naturally adopt the inference methods, and are quite straightforward to modify, making it fairly easy to introduce changes into model, which, along with the provided model selection criteria, can allow fast model prototyping.

1.4 Main contributions

We consider our main contribution developing a machine learning framework and its open-source implementation, based on variational message passing that allows performing probabilistic modeling, inference and model comparison.

The output of the development, apart from the framework itself and its implementation, involved the following:

- Providing a practical description of the existing methods for Bayesian inference, with a focus on variational approximations.
- Validating the correctness of the framework by comparing its performance to expectation-maximization on such tasks as learning parameters of Gaussian mixtures and image segmentation.
- Performing a case-study using the developed framework by applying it to basically unsolved problem, namely, noise reduction for speech enhancement.

We believe that the results of what we call *validation*, namely, comparing the performance of the approach implemented in the framework, with a baseline, maximum likelihood expectation-maximization (Section 2.3.2), on basic tasks, can also be of interest on their own, since they demonstrate how the variational message passing performs on standard Gaussian mixture learning tasks and image segmentation. Existing work [27] [4] performs similar experiments, however, they are using slightly different models and classical variational inference, rather than variational message passing.

Furthermore, we would like to mention that the purpose of a case-study was not to beat existing state-of-the-art approaches, but rather demonstrate, that the framework can be used to tackle highly complex problems, for which other existing frameworks at the current stage, for various reasons, are incapable of solving.

1.5 Outline

The presentation of the material of this thesis is targeted at people with the basic knowledge of the probability theory and moderate background in statistical machine learning.

The rest of this thesis is organized as follows. In Section 2 we provide the background on existing probabilistic modeling approaches, including graphical models and inference methods. It is required to give the reader a comprehensive explanation of the theoretical foundations of the framework, with a strong focus on inference using variational message passing. Then in Section 3 we describe the developed framework, shortly comparing it to existing alternatives and describing some implementation details. In Section 4, we demonstrate how the developed framework can be applied to several basic models, and validate that the inference indeed works comparable and sometimes outperforms the baseline (expectation-maximization). Then, in Section 5 we describe a case study using developed framework to the area of interest, namely, to noise reduction for speech signals, based on the generative model described in [8]. Finally, in Section 6 we discuss the contributions of this work and possible directions for the future work.

2 Background

The framework includes three major parts: a way to specify probabilistic models, an algorithm to do reasoning (inference) on these models and a method to compare model performance. In this section, we provide the necessary theoretical background for those.

We start by a short introduction into probabilistic inference in general, and then describe several graphical model notations, which we use as a convenient way for model specification. We then give a short overview of existing approaches to inference, elaborating more on the approach that we adopted within the framework, variational methods. After that, we briefly discuss possible approaches to do inference in online settings. Even though we have not introduced any of those methods in our framework, they can be quite valuable when solving signal processing tasks that are time-varying in their essence. Finally, we focus on the issue of model selection, again, with a bias towards variational methods.

Note that not all the methods described here are directly applied in our framework, but rather the background is given here to set some important concepts and to define keywords used along all the thesis, such as, factor graphs, conjugate-exponential family, or message passing algorithms. Furthermore, it provides some reasoning about the approaches used in the framework. For a more complete and deep discussion of Bayesian methods in machine learning, one can refer to [3], [1].

2.1 Probabilistic Inference

The main principle of probabilistic reasoning when modeling a process of interest, is identifying all of the process components as random variables $X_1, \ldots, X_N$ and specifying their interaction via probability model (distribution) $p(X_1, \ldots, X_N)$. The reasoning itself, which is often referred to as *inference*, is then done by applying the Bayes theorem and conditional probability rules to compute probabilities of variables of interest (or a combination of those). In this work, due to the target application area - signal processing, we are focusing mostly on continuous variables and distributions.

2.1.1 Bayes Rule

In practically any settings, the idea of the scientific reasoning can be formulated as follows: given the observed data $\boldsymbol{V}$ and some prior knowledge about the data generation process, determine the value of the unobserved (hidden) variables $\boldsymbol{H}$ of the process. Within the probabilistic framework, this can be expressed using the Bayes rule [1]:

$$p(\boldsymbol{H}|\boldsymbol{V}) = \frac{p(\boldsymbol{V}|\boldsymbol{H})p(\boldsymbol{H})}{p(\boldsymbol{V})} = \frac{p(\boldsymbol{V}|\boldsymbol{H})p(\boldsymbol{H})}{\int p(\boldsymbol{V}|\boldsymbol{H})p(\boldsymbol{H})d\boldsymbol{H}}. \tag{2.1}$$

where

- $p(\boldsymbol{H})$ is the *prior distribution*, it represents the initial beliefs about the hidden variables $\boldsymbol{H}$: how they are distributed and how they are related to each other;
- $p(\boldsymbol{V}|\boldsymbol{H})$ is called *likelihood*, it specifies how the observed data is being generated, given the hidden values, in other words, it represents the *generative model* of the process;
- $p(\boldsymbol{H}|\boldsymbol{V})$ is called the *posterior distribution*, it represents the beliefs about $\boldsymbol{H}$ *after* observing the data $\boldsymbol{V}$;
- $p(\boldsymbol{V})$ is called *marginal likelihood* or the *model evidence*, it is the same for all possible $\boldsymbol{H}$, so it is often simply considered as a constant factor.

2.1.2 Parameters and Hyperparameters

Note that sometimes, it is more natural to talk about the hidden variables $\boldsymbol{H}$ as of *parameters* of the model, especially when they correspond to parameters of some probability distribution.

For example, let's assume that we have a set of i.i.d. observations $\boldsymbol{X} = \{x_1, \ldots, x_n\}$, that are (presumably) distributed according to a Gaussian $\mathcal{N}(\mu, \gamma^{-1})$, where μ is the *mean* parameter, and γ is the *precision* parameter, the inverse of the *variance* (for a description of the distributions used in this work, refer e.g. to Appendix A, or to a more detailed one [1, Ch. 8.3, p.163]). The joint distribution over observations and parameters is then:

$$p(\boldsymbol{X}, \mu, \gamma) = \prod_n \mathcal{N}(x_n|\mu, \gamma^{-1}) \tag{2.2}$$

Here, in terms of hidden and observed variables, $\boldsymbol{X}$ is a set of observed variables and μ,γ are hidden variables, but it is also quite natural to call them *parameters*.

Furthermore, if we make assumptions on how μ and γ are distributed themselves, i.e. if we introduce prior distributions $p(\mu|\boldsymbol{\theta}_\mu)$ and $p(\gamma|\boldsymbol{\theta}_\gamma)$ over μ and γ correspondingly, the joint distribution over all the model variables will look as follows:

$$p(\boldsymbol{X}, \mu, \gamma, \boldsymbol{\theta}_\mu, \boldsymbol{\theta}_\gamma) = p(\mu|\boldsymbol{\theta}_\mu)p(\gamma|\boldsymbol{\theta}_\gamma)\prod_i \mathcal{N}(x_n|\mu, \gamma^{-1}) \tag{2.3}$$

Now, the set of hidden variables is extended with $\boldsymbol{\theta}_\mu$ and $\boldsymbol{\theta}_\gamma$, which are basically parameters of prior distributions for the parameters μ and γ. These are often referred to as *hyperparameters*. Moreover, this process of introducing priors over parameters can be continued even further, leading to so-called *hierarchical* Bayesian models [21], which underneath are quite similar to *deep learning*, which is currently one of the most promising machine learning approaches [13].

2.1.3 Generative vs Discriminative Learning

It is also worth mentioning that in the context of machine learning, the process of modeling posterior distribution using the generative model and prior is called *generative learning*, while the alternative approach models the posterior distribution *directly* is referred to as *discriminative learning*. In other words, it may be stated that generative approach models *both* hidden and observed variables, whereas discriminative focuses only on observed ones, and, broadly speaking, is less generic.

However, it does not mean that generative methods provide better performance: when providing prior beliefs, one will most definitely introduce some assumptions and approximations, which can be wrong or not precise enough. Thus, the question of which approach is better for a particular task in terms of performance is not straightforward and will most definitely depend on the application settings [15]. In this work, we are focusing on the generative modeling, since we believe that in the area of signal processing the domain knowledge .

2.2 Graphical Models

Broadly speaking, a *graphical model* is a notation for representing dependencies and independences between random variables for probability distributions. Typically, they merge together a *certain form* of probability distributions that are being modeled and their vivid representation using graph structures. For instance, Bayesian Networks, described below, imply a distribution that encodes causal relationships between variables, which are then represented as a directed graph. Another type of graphical models, factor graphs, are capable of representing any function that can be written as a product of several multipliers using a laconic graph structure. In this work, we are interested in graphical models since they are very convenient when specification of a complex model is needed, especially when the description using conventional notation is getting crowded.

However, graphical models turn out to be useful not only for describing probabilistic models, but also very naturally correspond to efficient inference algorithms on those models, so-called *message-passing algorithms*, such as sum-product algorithm (Section 2.3.1), and variational message passing (Section 2.4.1). Below, we give a very short overview of several popular graphical notations.

2.2.1 Bayesian Networks

A *Bayesian network* is a probabilistic graphical model that is used to represent random variables and their conditional dependencies using a directed acyclic graph (DAG). Nodes of this graph represent random variables, and directed edges are used to describe causal relationships between variables: the starting node is referred to as a parent node, and the destination node is called a child node.

The edge structure in the graph in fact represents conditional independences. Each variable, conditioned on its parents, is independent from all its ancestors. Thus, the general form of the probability distribution for a Bayesian network with variables $X_1, \ldots, X_N$ will be:

$$p(X_1, \ldots, X_N) = \prod_i p(X_i | \mathrm{pa}(X_i)). \tag{2.4}$$

where $\mathrm{pa}(X_i)$ represents the set of parents of the variable X_i.

An example of a Bayesian network is given in the Figure 1, it denotes N i.i.d. observations x_n sampled from a Gaussian distribution $\mathcal{N}(\mu, \gamma^{-1})$.

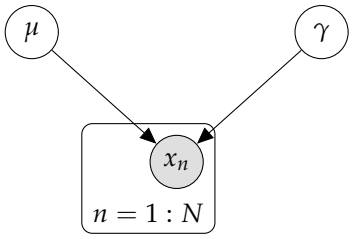


Figure 1: Bayesian network of a sequence of Gaussian observations.

Note that observations x_n are depicted using so-called *plate* notation [1, Ch.9.1], a box around the node that denotes that the nodes lying inside are repeated and indexed by $1 \leq n \leq N$.

2.2.2 Forney Factor Graphs

A *factorization* of a function is a representation of a function as a product of several multipliers, or *factors*, e.g. see (2.5), as an example of factorization of $f(w, x, y, z)$ into three factors f_1,f_2 and f_3. *Factor graphs* correspond to a specific type of a graphical model that is used to express factorizations of functions. Factor graphs were originally introduced by Frey et al. in [7], and they are quite expressive: such popular graphical models as Markov networks and Bayesian networks can be expressed via factor graphs [2]. There are several types of factor graph notations that have similar expressive power. In this section, we will discuss one of the most recent notations, Forney factor graphs (FFG) [17]. We find them more convenient for showing what factor graphs actually are, and, moreover, for formulating a very basic inference algorithm (sum-product algorithm, see Section 2.3.1).

We will now demonstrate on a small example how one can construct a FFG given a factorization of a function. Let's consider a factorization of a multivariate function $f(w, x, y, z)$:

$$f(w, x, y, z) = f_1(w, x, y) f_2(y, z) f_3(z). \tag{2.5}$$

The FFG expressing this factorization is provided in the Figure 2. As one can see, the graph comprises three different types of elements:

- nodes, represented by f_1, f_2 and f_3;
- edges, represented by y and z;
- half-edges, represented by w and x.

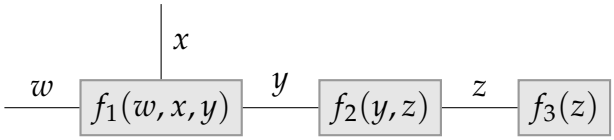


Figure 2: Forney factor graph for a factorization of $f(w, x, y, z)$.

In general, a factor graph can be constructed out of a factorization according to the following three rules:

- for each factor, there is a node in the graph;
- for each variable appearing in two factors, there is an edge in the graph;
- for each variable appearing in one factor, there is a half-edge in the graph.

Note that these rules imply that factorizations are restricted to those with no more than two factors sharing a variable. This restriction can be easily resolved though, by introducing additional variables and an equality constraint.

Let's assume that we have a factorization:

$$f(x, y, z) = f_1(x, y) f_2(y, z) f_3(y), \tag{2.6}$$

where the variable y is shared by three factors f_1, f_2 and f_3. To overcome this, we can introduce auxiliary variables y' and y'', and an equality constraint $y = y' = y''$. The factor that implements this equality constraint is:

$$f_=(y, y', y'') = \delta(y - y')\delta(y - y''), \tag{2.7}$$

where δ is Dirac delta function (B.1). The FFG of the modified factorization is presented in the Figure 3.

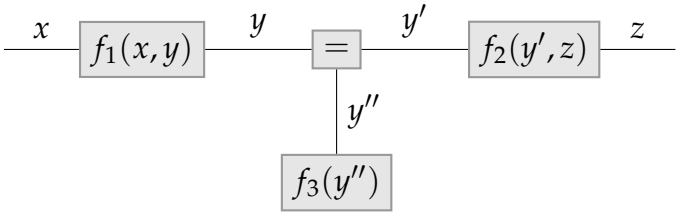


Figure 3: Forney factor graph for a modified factorization of $f(x, y, z)$.

2.2.3 Factor Graphs

Forney factor graphs are a quite illustrative for showing basic concepts, and are also widely used for state-space modeling [17]. However, in this work, we make use mostly of the traditional factor graph notation [7], due to the fact that depicting certain kind of models, for example, mixture models, is not well-defined for FFGs, and requires additional research effort.

The only major difference between the traditional notation and FFGs is that there are two types of nodes on the graph instead of one: one for variables, and one for factor nodes. An example demonstrating traditional factor graph corresponding to a Gaussian-distributed variable is given in the Figure 4. Small black boxes represent factor nodes (which include distributions), in this case we have specified that variable x is distributed as a Gaussian with mean and precision μ and γ. μ is also distributed as a Gaussian, with parameters m and β, that has constant parameters (priors on m and β). Identically for γ that is distributed according to gamma distribution with constant parameters. Circles represent variables, and filled circles indicate that the variable x is observed.

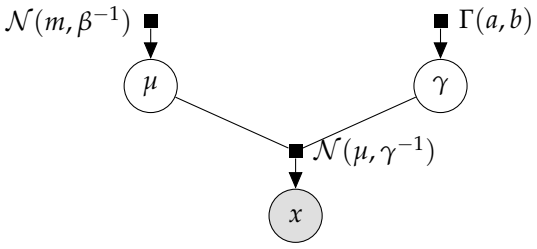


Figure 4: Factor graph for a Gaussian-distributed variable.

2.3 Inference Methods

Let's assume now that we have a probabilistic model, either defined by a Bayesian network or a factor graph, or directly by a joint probability distribution over hidden and observed variables. In order to answer meaningful questions, one will most probably be interested in computing a *posterior probability* over a *particular hidden variable* (or a group of those). In this section, we discuss various approaches to this process, which is called *inference*.

Inference algorithms can be classified with respect to various criteria. One of them is whether the algorithm is *exact*, or whether it is some kind of *approximation*. For some models, namely, for models containing only discrete variables or Gaussian-distributed variables [17], it is possible to derive a closed-form analytical solution, that leads to an algorithm known as *sum-product algorithm* (see Section 2.3.1). In case one wants to use a more involved model, for instance, if one is modeling a sequence of i.i.d. Gaussian samples (e.g. as in Section 2.1.2), and wants to specify a prior distribution for the precision parameter, it is necessary to use a non-Gaussian distribution (simply because precision is a non-negative quantity, whereas Gaussian distribution does not impose that restriction). Unfortunately, for such models exact methods do not exist [32], and approximation methods are needed. These are discussed in Section 2.3.3.

Another way of looking on inference methods is comparing the way they treat model parameters. So-called *fully Bayesian methods* allow models, in which parameters are defined using full probability distributions. Sum-product algorithm and approximation algorithms described in the following sections all belong to this class. On the other hand, non-fully Bayesian methods, such as EM methods described in Section 2.3.2, treat parameters as point estimates, which, of course, are less informative compared to full distributions.

2.3.1 Sum-Product Algorithm

Given a joint distribution $p(X_1, \ldots, X_N)$, the goal of *marginal inference* is to compute marginal distributions over subsets of variables, possibly conditioned on other subsets. For continuous models, this concerns computation of integrals of the joint distribution.

It is possible to compute the exact marginals only for certain classes

of models: typically the exact marginal inference algorithms are derived specifically for discrete models [1]. An important class of continuous distributions for which an efficient algorithm exist is Gaussian. The reason for that is that the algorithm described here requires the distribution to be closed under *marginalization* (integration) and *multiplication* operations.

A distribution family is closed under marginalization, if for any distribution from this family, after integrating out one of the variables, it stays in the *same family*, meaning that it has the same functional from as the original. For example, if it was a multivariate Gaussian distribution:

$$p(x, y) = \mathcal{N}(\begin{bmatrix} x \\ y \end{bmatrix}) \tag{2.8}$$

then, after integrating out y, the result is still Gaussian:

$$p(x) = \int p(x, y) = \mathcal{N}(x) \tag{2.9}$$

Similarly, the distribution family is closed under multiplication, if the multiplication any two distributions (to be more precise, their PDFs) from this family, again, has the same functional form.

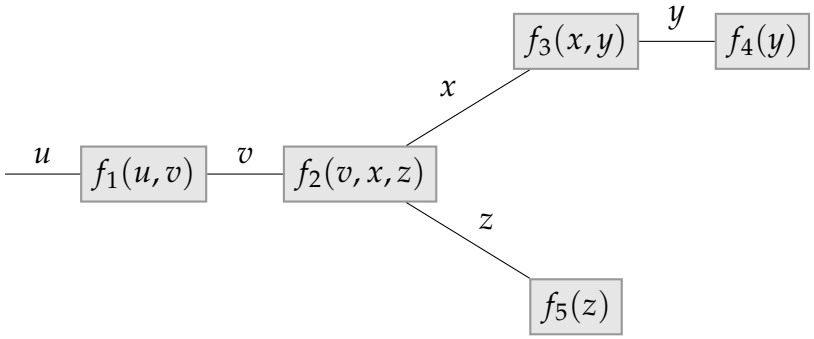


Figure 5: Forney factor graph for $f(u, v, x, y, z)$.

Sum-product algorithm, also known as *belief propagation* algorithm, is usually formulated for factor graphs (see Section 2.2.2). Let's consider the following probability mass distribution corresponding to the Forney factor graph presented in the Figure 5:

$$f(u, v, x, y, z) = f_1(u, v) f_2(v, x, z) f_3(x, y) f_4(y) f_5(z), \tag{2.10}$$

and assume that we are interested in the distribution of a specific variable $p(u)$, which can be found using the following marginalization:

$$p(u) = \int_{v,x,y,z} f(u,v,x,y,z) = \int_{v,x,y,z} f_1(u,v) f_2(v,x,z) f_3(x,y) f_4(y) f_5(z). \tag{2.11}$$

If, for now, we assume that variables v, x, y, z are discrete, (2.11) can be computed simply by summing up over all possible values of those variables. However, if the number of possible values is large, this type of inference is becoming computationally heavy. The sum-product algorithm uses the factor graph *structure* in order to make inference more efficient - it substitutes the computation of the global marginal (2.11) by a sequence of smaller local marginals. In other words, we eliminate variables in some order by pushing an integration sign in (2.11) to the right.

For this particular example, we might first integrate out the variable y. Only two factors f_3 and f_4 have y as a parameter, so we can rewrite (2.11) as follows:

$$p(u) = \int_{v,x,y,z} f(u,v,x,y,z) = \int_{v,x,z} f_1(u,v) f_2(v,x,z) f_5(z) \mu_1(x), \tag{2.12}$$

where:

$$\mu_1(x) = \int_y f_3(x,y) f_4(y). \tag{2.13}$$

This integral $\mu_1(x)$ is called a *message*, that is sent from the factor f_3 to the factor f_2 along an edge x. Similarly, we could have defined a message from f_4 to f_3 by computing $\int_y f_4(y)$ first, but in this case it does not make much difference since it will be a constant (one, if f_4 is a well-defined probability distribution). We can apply identical technique to eliminate two variables at once, x and z:

$$p(u) = \int_{v,x,y,z} f(u,v,x,y,z) = \int_{v,x,z} f_1(u,v) f_2(v,x,z) \mu_2(v), \tag{2.14}$$

where μ_2 is a message from f_2 to f_1:

$$\mu_2(v) = \int_{x,z} f_2(v,x,z) f_5(z) \mu_1(x). \tag{2.15}$$

Note, that here we are using the message μ_1 that we have computed previously.

And finally we can get the required marginal distribution:

$$p(u) = \int_v f_1(u, v)\mu_2(v). \tag{2.16}$$

As mentioned above, the local integrals $\mu_1(x)$ and $\mu_2(v)$ can be called *messages*, and the whole process can be seen as a sequence of messages, passing from one node of the factor graph to another. In fact, these several steps for computing $p(u)$ described above can be generalized as the *sum-product rule*: the message from a node f along some edge x is formed as the product of f and all incoming messages along all edges except x, integrated over all involved variables except x itself.

For a cycle-free graph, the direct application of this rule leads to a message-passing algorithm: the sum-product algorithm. Again, messages from nodes like f_4 and f_5 that depend only on a single variable, can be considered as constants. For graphs with cycles, the computation can be done iteratively, according to some message-passing schedule that is specified in advance, with incoming messages initialized to identities. Note that for discrete variables integration should be substituted by summation, this is where the word "sum" in the name of the algorithm is coming from.

The main drawback of the sum-product algorithm described here is a limited class of applicable models: the algorithm is only proven to converge for cycle-free graphs, and for continuous variables only Gaussian distributions are known to be supported [17].

2.3.2 Expectation-Maximization

Expectation-Maximization (EM) is a generic iterative method for finding maximum likelihood (ML) or maximum a-posteriori (MAP) *point estimates* for models with hidden variables [1].

To understand how it works, let's consider a probabilistic model with observed variables denoted as $\boldsymbol{V}$ and hidden variables as $\boldsymbol{H}$. The joint distribution $p(\boldsymbol{V}, \boldsymbol{H}|\boldsymbol{\theta})$ is parametrized by the vector $\boldsymbol{\theta}$. The classical version of the EM algorithm seeks for a specific set of parameters that maximize

the likelihood function $p(\boldsymbol{V}|\boldsymbol{\theta})$:

$$\boldsymbol{\theta}^{ML} = \arg\max_{\boldsymbol{\theta}} p(\boldsymbol{V}|\boldsymbol{\theta}) = \arg\max_{\boldsymbol{\theta}} \int_{\boldsymbol{H}} p(\boldsymbol{V}, \boldsymbol{H}|\boldsymbol{\theta}) \tag{2.17}$$

It can be proven (see Appendix C.1), that for any distribution $q(\boldsymbol{H})$ over hidden variables it holds that:

$$\ln p(\boldsymbol{V}|\boldsymbol{\theta}) = \mathcal{L}(q, \boldsymbol{\theta}) + \mathrm{KL}(q||p) \tag{2.18}$$

where:

$$\mathcal{L}(q, \boldsymbol{\theta}) = \int_{\boldsymbol{H}} q(\boldsymbol{H}) \ln \frac{p(\boldsymbol{V}, \boldsymbol{H}|\boldsymbol{\theta})}{q(\boldsymbol{H})} \tag{2.19}$$

$$\mathrm{KL}(q||p) = -\int_{\boldsymbol{H}} q(\boldsymbol{H}) \ln \frac{p(\boldsymbol{H}|\boldsymbol{V}, \boldsymbol{\theta})}{q(\boldsymbol{H})} \tag{2.20}$$

$\mathrm{KL}(q||p)$ is the Kullback-Leibler (KL) divergence between a newly introduced distribution $q(\boldsymbol{H})$ (*variational* distribution) and the posterior distribution $p(\boldsymbol{H}|\boldsymbol{V}, \boldsymbol{\theta})$. Note that it can only be non-negative, being zero if and only if $q(\boldsymbol{H}) = p(\boldsymbol{H}|\boldsymbol{V}, \boldsymbol{\theta})$ [1, Ch.8.2.1, p.161]. Thus, from (2.18) $\mathcal{L}(q, \boldsymbol{\theta})$ is a lower bound on the log-likelihood:

$$\mathcal{L}(q, \boldsymbol{\theta}) \leq \ln p(\boldsymbol{V}|\boldsymbol{\theta}), \tag{2.21}$$

The EM algorithm consists of two steps that follow (2.18). Let t be the index of the current iteration. Then during the t-th *E-step*, $\boldsymbol{\theta}$ is set to a fixed value: $\boldsymbol{\theta}_{t-1}$, and $\mathcal{L}(q, \boldsymbol{\theta}_{t-1})$ is maximized w.r.t. $q(\boldsymbol{H})$, resulting in a new estimate for $q(\boldsymbol{H})$. Note that this means that for $t = 1$ it is necessary to have some initial guess $\boldsymbol{\theta}^0$. During the following *M-step*, the distribution q is fixed to the value q computed on the E-step, and $\mathcal{L}(q, \boldsymbol{\theta})$ is maximized w.r.t. $\boldsymbol{\theta}$, resulting in a new estimate $\boldsymbol{\theta}_t$. This procedure repeats until either the parameters or the value of the lower bound converge. It can be shown [3] that the two steps indeed maximize the lower bound and lead to convergence, although not necessarily to the global maximum.

The classical formulation of EM algorithm can also be modified to maximize the a-posteriori distribution (MAP):

$$\boldsymbol{\theta}^{MAP} = \arg\max_{\boldsymbol{\theta}} p(\boldsymbol{\theta}) p(\boldsymbol{V}|\boldsymbol{\theta}) \tag{2.22}$$

In this case the M-step slightly differs [3] because of the presence of the prior distribution. Note that in general maximum likelihood is a special case of the maximum a-posteriori with a uniform prior.

2.3.3 Approximate Inference

As discussed earlier, exact inference algorithms only exist for a somewhat limited class of models. For more complex models, such as ones involving distributions that are not closed under multiplication, exact inference is analytically non-tractable. This is why a lot of advances in probabilistic inference relate to approximation methods, which can be divided into two major classes: sampling methods and deterministic approximations.

Sampling methods, such as Markov Chain Monte Carlo (MCMC), and, in particular, Gibbs sampling, represent modeled distributions using a set of samples. As the number of samples grow, an approximation becomes more and more precise. One can think of these samples as of a histogram that is used instead of the distribution itself. However, to reach reasonable precision, these methods might be very expensive, both in terms of computation time and memory (for an introduction into sampling methods see, e.g. [1, Ch.27]).

Deterministic approximations overcome some problems of the sampling methods. Instead of using data samples, deterministic methods approximate otherwise intractable distributions using flexible enough models for which inference can be done efficiently. Two of the most widely adopted methods are *expectation-propagation* (EP), [23] and *Variational Bayes* (VB), [32].

In this work, we focus on Variational Bayes, or, more precisely, on its generalization, *variational message passing* (VMP), since it provides a very nice feature that the expectation-propagation approach is lacking: a build-in method for model comparison. Expectation-propagation, which is a generalization of a sum-product algorithm, nevertheless, is also a very promising method [23], and, ultimately, these two methods, EP and VMP, can be generalized further [22] and used together within a single framework.

2.4 Variational Inference

Variational inference or *Variational Bayes* (VB) is an approximate deterministic approach to inference [32]. The main idea behind VB is to approximate a posterior distribution by a simpler distribution assuming that given the observed data, some hidden variable can be considered as independent.

Particularly, a posterior distribution over hidden variables $p(\boldsymbol{H}|\boldsymbol{V})$ is approximated by a *variational distribution* $q(\boldsymbol{H})$ that belongs to a restricted family of distributions, that is simpler than the true posterior:

$$p(\boldsymbol{H}|\boldsymbol{V}) \approx q(\boldsymbol{H}). \tag{2.23}$$

Note that the choice of the distribution family for $q(\boldsymbol{H})$ should be justified: $q(\boldsymbol{H})$ should be flexible enough to be able to approximate the true posterior. A very common measure of dissimilarity for probability distributions is the Kullback-Leibler divergence, that we have already seen when discussing EM algorithm (Section 2.3.2):

$$\mathrm{KL}(q||p) = \int_x q(x) \ln \frac{q(x)}{p(x)}. \tag{2.24}$$

When the distributions are equal ($q = p$) the measure is zero. It is worth mentioning that this is not a symmetric measure, i.e. $\mathrm{KL}(q||p) \neq \mathrm{KL}(p||q)$ - the two variants might favor different distributions when minimized [32]. In variational inference, $\mathrm{KL}(q||p)$ is preferred since it leads to significant computation simplifications.

In order to find the variational $q(\boldsymbol{H})$ that would approximate $p(\boldsymbol{H}|\boldsymbol{V})$ we can minimize the KL divergence between these two distributions:

$$\mathrm{KL}(q||p) = \int_{\boldsymbol{H}} q(\boldsymbol{H}) \ln \frac{q(\boldsymbol{H})}{p(\boldsymbol{H}|\boldsymbol{V})}. \tag{2.25}$$

Unfortunately, this equation cannot be solved unless we know the true posterior, which is a distribution that is too complex to evaluate in the first place. Luckily, it is possible to substitute it using the rule of conditional probability: $p(\boldsymbol{H}|\boldsymbol{V}) = \frac{p(\boldsymbol{H},\boldsymbol{V})}{p(\boldsymbol{V})}$. Then, the KL divergence will then can be rewritten as:

$$\mathrm{KL}(q||p) = -\mathcal{L}(q) + \ln P(\boldsymbol{V}), \tag{2.26}$$

where:

$$\mathcal{L}(q) = \int_{\boldsymbol{H}} q(\boldsymbol{H}) \ln \frac{q(\boldsymbol{H})}{p(\boldsymbol{H}, \boldsymbol{V})} = \langle \ln p(\boldsymbol{H}, \boldsymbol{V}) \rangle_{q(\boldsymbol{H})} + \mathrm{H}(q), \tag{2.27}$$

where $\langle \rangle_{q(\boldsymbol{H})}$ denotes expectation w.r.t. the distribution $q(\boldsymbol{H})$, in this case:

$$\langle \ln p(\boldsymbol{H}, \boldsymbol{V}) \rangle_{q(\boldsymbol{H})} = \int_{\boldsymbol{H}} q(\boldsymbol{H}) \ln p(\boldsymbol{H}, \boldsymbol{V}), \tag{2.28}$$

and $\mathrm{H}(q)$ denotes the entropy of $q(\boldsymbol{H})$, which is defined as:

$$\mathrm{H}(q) = \int_{\boldsymbol{H}} q(\boldsymbol{H}) \ln q(\boldsymbol{H}). \tag{2.29}$$

From (2.26), one can see that since $\ln P(\boldsymbol{V})$ does not depend on $q(\boldsymbol{H})$, it is possible to substitute KL minimization by $\mathcal{L}(q)$ maximization. Note that $\mathcal{L}(q)$ is actually a *lower bound* on the likelihood $\ln P(\boldsymbol{V})$, which will be useful for model comparison. We refer to this lower bound as *variational lower bound*, in some literature it is also sometimes called *variational free energy*.

In order to make the minimization problem tractable, it is necessary to select an appropriate variational distribution $q(\boldsymbol{H})$. The default choice that simplifies computations dramatically is using a fully factorized distribution:

$$q(\boldsymbol{H}) = \prod_i q_i(H_i). \tag{2.30}$$

Note that this factorization does not mean that variables H_i cannot have any dependencies. Since $q(\boldsymbol{H})$ is the approximate *posterior*, it means that any possible dependencies that remain between these groups *after conditioning on* $\boldsymbol{V}$, will not be captured.

Now we can substitute the factorized form of $q(\boldsymbol{H})$ into (2.27), and get the following expression for the $\mathcal{L}(q)$:

$$\mathcal{L}(q) = \int_{\boldsymbol{H}} \prod_i q_i(H_i) \ln p(\boldsymbol{H}, \boldsymbol{V}) + \sum_i \mathrm{H}(q_i). \tag{2.31}$$

We now will show that it is possible to optimize $\mathcal{L}(q)$ incrementally, by doing maximization with respect to separate variables H_i. For that, we

separate terms for a single factor $q_j(H_i)$:

$$\mathcal{L}(q) = \int_{H_j} q_j(H_j) \langle \ln p(\boldsymbol{H}, \boldsymbol{V}) \rangle_{\sim q_i} + \mathrm{H}(q_j) + \sum_{i \neq j} \mathrm{H}(q_i), \tag{2.32}$$

where $\langle \rangle_{\sim q_j}$ denotes the expectation with respect to all factors *except* $q_j(\boldsymbol{H}_j)$. Now, we can introduce the following auxiliary distribution:

$$q_j^*(H_j) = \frac{1}{Z} \exp\Big(\langle \ln p(\boldsymbol{H}, \boldsymbol{V}) \rangle_{\sim q_j} \Big). \tag{2.33}$$

After this, the lower bound $\mathcal{L}(q)$ can be written as follows:

$$\mathcal{L}(q) = -\mathrm{KL}(q_j || q_j^*) - \ln Z + \sum_{i \neq j} \mathrm{H}(q_i). \tag{2.34}$$

Since only the first summand in (2.34) depends on q_j, it is clear that maximization of $\mathcal{L}(q)$ with respect to a single factor q_j is equivalent to the minimization of $\mathrm{KL}(q_j || q_j^*)$. KL divergence is minimized when its two argument distributions, thus to maximize $\mathcal{L}(q)$, one should set $q_j = q_j^*$. The same holds for any other factor, and hence $\mathcal{L}(q)$ can be iteratively maximized by updating factors one by one until the convergence is reached. Note that the equation for the optimal q_j (2.33) can be rewritten in log-domain for convenience:

$$\ln q_j^*(H_j) = \langle \ln p(\boldsymbol{H}, \boldsymbol{V}) \rangle_{\sim q_j} + \text{const.} \tag{2.35}$$

2.4.1 Variational Message Passing

Let's now apply variational inference to an arbitrary Bayesian network (Section 2.2.1) that corresponds to a distribution over a set of variables $\boldsymbol{X}$, and let $\boldsymbol{H}$ and $\boldsymbol{V}$ represent the subsets of hidden and observed respectively.

Again, to approximate the posterior, we introduce a *factorized* variational distribution over hidden variables (2.30). Now, we can substitute the distribution of Bayesian networks (2.4) into the variational posterior update equation (2.35) (see [32] for complete derivation):

$$\ln q_j^*(H_j) = \langle \ln p(H_j|\text{pa}_{H_j}) \rangle_{\sim q_j} + \sum_{k \in \text{ch}(H_j)} \langle \ln p(X_k|\text{pa}(X_k)) \rangle_{\sim q_j} + \text{const.} \tag{2.36}$$

where $\text{ch}(H_j)$ represents a set of children of H_j. Note that (2.36) only involves direct neighbors of H_j and co-parents of its children.

Hence, it is possible to update approximate posteriors q_j^* by using only local expectations from the neighboring nodes. These local expectations are often called *messages*, and the whole optimization algorithms boils down to passing messages from one random variable (or the corresponding node in the Bayesian network) to another, which is quite similar to what we have seen when discussing the sum-product algorithm (Section 2.3.1). The specific form of those message depends on the conditional distributions. In the following sections, we will describe the message passing algorithm for a specific (yet quite expressive) family of distributions.

2.4.2 Conjugate-Exponential Models

It turns out [32], that notable simplifications to the variational inference can be applied for a specific class of distributions - *conjugate-exponential models*. For a model to be conjugate-exponential, the two following properties should hold for the distribution of any variable conditioned on its parents:

- it should be in the exponential family;
- it should be conjugate to the distribution of the parents.

A conditional probability distribution $p(\boldsymbol{X}|\boldsymbol{Y})$ is said to be in the *exponential family* if it can be written in the following functional form:

$$p(\boldsymbol{X}|\boldsymbol{Y}) = \exp\left[\boldsymbol{\phi}(\boldsymbol{Y})^T \boldsymbol{u}(\boldsymbol{X}) + f(\boldsymbol{X}) + g(\boldsymbol{Y})\right], \tag{2.37}$$

where:

- $\boldsymbol{\phi}(\boldsymbol{Y})$ is the *natural parameter* vector;
- $\boldsymbol{u}(\boldsymbol{X})$ is the *natural statistics* vector;

- $g(\boldsymbol{Y})$, $f(\boldsymbol{X})$ are normalization terms.

To understand what it means for one distribution to be *conjugate* to another, let's consider an example of two distributions $p(X|Z)$ and $p(Y|X)$ in the exponential family:

$$p(X|Z) = \exp\left[\boldsymbol{\phi}_X(Z)^T \boldsymbol{u}_X(X) + f_X(X) + g_X(Y)\right], \tag{2.38}$$

$$p(Y|X) = \exp\left[\boldsymbol{\phi}_Y(X)^T \boldsymbol{u}_Y(Y) + f_Y(Y) + g_Y(X)\right]. \tag{2.39}$$

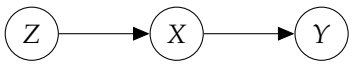


Figure 6: Bayesian network defined by $p(X|Z)$ and $p(Y|X)$.

Bayesian network for the model described via $p(X|Z)$ and $p(Y|X)$ is given in the Figure 6. A distribution $p(X|Z)$ is said to be *conjugate* to a child distribution $p(Y|X)$ if it has the same functional form with respect to X. In our settings, it means that $p(Y|X)$ can be rewritten or, in other words, *reparameterized*, to have the following form:

$$p(Y|X) = \exp\left[\boldsymbol{\phi}_{YX}(Y)^T \boldsymbol{u}_X(X) + \lambda(Y)\right], \tag{2.40}$$

that is, the natural statistics vector has the same form as in the distribution $p(X|Z)$.

It can be proven [32], that for any distribution in the exponential family, if the natural parameter $\boldsymbol{\phi}(\boldsymbol{Y})$ is known, the expectation of the natural statistics vector can be formulated as follows:

$$\langle \boldsymbol{u}(\boldsymbol{X}) \rangle_{p(\boldsymbol{X}|\boldsymbol{\phi})} = \frac{\partial \widetilde{g}(\boldsymbol{\phi})}{\partial \boldsymbol{\phi}}, \tag{2.41}$$

where $p(\boldsymbol{X}|\boldsymbol{\phi})$ is the original distribution reparameterized for the given parameter vector $\boldsymbol{\phi}$:

$$p(\boldsymbol{X}|\boldsymbol{\phi}) = \exp\left[\boldsymbol{\phi}^T \boldsymbol{u}(\boldsymbol{X}) + f(\boldsymbol{X}) + \widetilde{g}(\boldsymbol{\phi})\right]. \tag{2.42}$$

Now, let's show how the variational inference will look like for this kind of models. Following [32], let's consider optimization of a single

variational factor $q(Y)$ corresponding to a variable Y, that has a child node X. Keeping in mind that, since the model is conjugate-exponential, the log-conditional distributions for Y and X will look as follows:

$$\ln p(Y|\mathrm{pa}_Y) = \boldsymbol{\phi}_Y(\mathrm{pa}_Y)^T \boldsymbol{u}_Y(Y) + f_Y(Y) + g_Y(\mathrm{pa}_Y). \tag{2.43}$$

$$\ln p(X|Y, \mathrm{cp}_Y) = \boldsymbol{\phi}_X(Y, \mathrm{cp}_Y)^T \boldsymbol{u}_X(X) + f_X(X) + g_X(Y, \mathrm{cp}_Y), \tag{2.44}$$

where cp_Y denotes the parents of X excluding Y (co-parents with respect to Y).

First of all, we could notice that due to the conjugacy property, $\ln p(X|Y, \mathrm{cp}_Y)$ should be in exponential family with respect to Y, i.e. it can be reparameterized in the following way:

$$\ln p(X|Y, \mathrm{cp}_Y) = \boldsymbol{\phi}_{XY}(X, \mathrm{cp}_Y)^T \boldsymbol{u}_Y(Y) + \lambda(X, \mathrm{cp}_Y). \tag{2.45}$$

Then, we can substitute these forms of log-conditionals into the update equation (2.36):

$$\begin{aligned} \ln \boldsymbol{\phi}^*(Y) = \; & \langle \boldsymbol{\phi}_Y^T(\mathrm{pa}_Y) \boldsymbol{u}_Y(Y) \rangle_{\sim q_j} \\ & + \textstyle\sum_{k \in \mathrm{ch}(H_j)} \langle \boldsymbol{\phi}_{XY}^T(X_k, \mathrm{cp}_Y) \boldsymbol{u}_Y(Y) + \lambda(X_k, \mathrm{cp}_Y) \rangle_{\sim q_j}. \end{aligned} \tag{2.46}$$

Then, regrouping expectations by to get a standard functional form of the exponential family with respect to $\boldsymbol{u}_Y$, we will get:

$$\begin{aligned} \ln q_j^*(Y) = \; & \left(\langle \boldsymbol{\phi}_Y(\mathrm{pa}_Y) \rangle_{\sim q_y} + \textstyle\sum_{k \in \mathrm{ch}} \langle \boldsymbol{\phi}_{XY}(X_k, \mathrm{cp}_Y) \rangle_{\sim q_y} \right)^T \boldsymbol{u}_Y(Y) \\ & + f_Y(Y) + \mathrm{const}. \end{aligned} \tag{2.47}$$

Now one can see, that q_j^* is of the same functional form as $p(Y|\mathrm{par}_Y)$, with the following natural parameter vector $\boldsymbol{\phi}^*$:

$$\boldsymbol{\phi}^* = \langle \boldsymbol{\phi}_Y(\mathrm{pa}_Y) \rangle_{\sim q_y} + \sum_{k \in \mathrm{ch}} \langle \boldsymbol{\phi}_{XY}(X_k, \mathrm{cp}_Y) \rangle_{\sim q_y}. \tag{2.48}$$

2.4.3 VMP on Factor Graphs

So far, we have been applying VMP in Bayesian networks. However, it is often more convenient to represent probabilistic models using factor graphs [25], [17].

It is rather straightforward to formulate VMP rules for factor graphs [32]. The message from a hidden variable X_i to a factor node f_i will be:

$$m_{X_i \to f_j} = \langle \boldsymbol{u}_{X_i}(X_i) \rangle, \tag{2.49}$$

and the message from a factor node f_j to a variable node X_i is:

$$m_{f_j \to X_i} = \textbf{Natural}(\langle f_j \rangle_{\sim q_{X_i}}), \tag{2.50}$$

where **Natural** returns a natural parameter vector of a provided distribution. The natural parameter of the approximate posterior for a variable X_i then can be computed as a sum of all the messages it received from its neighboring factors:

$$\boldsymbol{\phi}^*_{X_i} = \sum_{f_j \in \text{ne}_i} m_{f_j \to X_i}. \tag{2.51}$$

2.4.4 Deterministic Functions

The class of models supported by VMP can be extended by introducing deterministic relationships between variables. Let's consider an arbitrary node X that represents a deterministic function of its inputs $X = f(Y_1, \ldots, Y_N)$, and has a child Z. Because of the conjugacy constraints, the form of the message that should be send from X to Z is determined by the functional form of the distribution of Z.

If the node X was stochastic, we could, as usual, derive an update for its approximate posterior distribution. The updated messages to its children then can be computed as an expectation of the natural statistics vector can be computed (2.41). Messages to a parent, in their turn, can be computed using the messages from co-parents and the sum of messages from children.

2.4.5 Mixture Distributions

Mixture distributions, particularly mixtures of Gaussians are widely used in Bayesian statistics [1]. A density function of a mixture distribution

over a random variable X is a convex sum of component distributions densities:

$$p(X|\boldsymbol{\pi}, \boldsymbol{Y}) = \sum_{i=1}^{N} \pi_i p_i(X|Y_i), \tag{2.52}$$

where π_i are *mixture weights*, which should sum up to one, and $p_i(X|Y_i)$ are separate distributions with parameters Y_i. Note, that with this definition we assume that the number of mixtures is finite and equals to N, whereas infinite mixture models also exist.

From (2.52) it is clear that mixture distributions do not belong to the exponential family. The workaround [32] is to introduce an auxiliary hidden discrete variable - λ, so that the mixture distribution can be rewritten as:

$$p(X|\lambda, \boldsymbol{Y}) = \prod_{i=1}^{N} p_i(X|Y_i)^{\delta(i,\lambda)}. \tag{2.53}$$

It is worth noting that this does not break our exponential-conjugacy constraints, since the distribution of λ is still in exponential family and, as will be clear $p(X|\lambda, \boldsymbol{Y})$ is of the same functional form with respect to $\boldsymbol{u}_\lambda$:

$$p(\lambda|\boldsymbol{p}) = \exp\left(\begin{bmatrix} \ln p_1 \\ \vdots \\ \ln p_N \end{bmatrix}^T \begin{bmatrix} \delta(\lambda - 1) \\ \vdots \\ \delta(\lambda - N) \end{bmatrix} \right), \tag{2.54}$$

where $\boldsymbol{p}$ is a parameter vector $[p_1, \ldots, p_N]$ whose i-th component is $p(\lambda = i)$. Luckily, Dirichlet distribution is conjugate to the discrete distribution and can be used as a prior on $\boldsymbol{p}$.

Back to the mixture distribution, if we fix all the component distributions to be in the exponential family and have the *same functional form*, we can write the log-conditional as follows:

$$\begin{aligned} \ln p(X|\lambda, \boldsymbol{Y}) &= \left[\textstyle\sum_i \delta(i,\lambda)\boldsymbol{\phi}(Y_i)\right]^T \boldsymbol{u}_X(X) + f_X(X) + \textstyle\sum_i \delta(i,\lambda) g_i(Y_i) \\ &= \boldsymbol{\phi}_X(\lambda, \boldsymbol{Y})^T \boldsymbol{u}_X(X) + f_X(X) + g(\boldsymbol{\phi}_X). \end{aligned} \tag{2.55}$$

The distribution (2.55) is in the exponential family and has a natural parameter vector $\boldsymbol{\phi}_X = \sum_i \delta(i,\lambda)\boldsymbol{\phi}(Y_i)$. Note that it is also of the same functional form as all of its mixture components.

We now can apply variational message passing. A message from X to any child Z_i will be:

$$m_{X\to Z_i} = \langle \boldsymbol{u}_X(X)\rangle = \sum_i \langle\delta(i,\lambda)\rangle\langle \boldsymbol{u}_{Y_i}(Y_i)\rangle. \tag{2.56}$$

A message from X to a parent Y_i will be the same that would be sent to Y_i if X was not a mixture distribution, but scaled with the posterior over λ: $q(\lambda = i) = \langle\delta(i,\lambda)\rangle$. Since we have introduced a new parent for X - λ, it is also necessary to send a message to update its posterior. Keeping in mind the form of the discrete distribution (2.54), the mixture distribution (2.55) can also be reparameterized accordingly, and the resulting message will look as follows:

$$m_{X\to\lambda} = \begin{bmatrix} \langle \ln p_1(X|Y_1)\rangle \\ \vdots \\ \langle \ln p_N(X|Y_N)\rangle \end{bmatrix}. \tag{2.57}$$

A message from a child Z_i to X, again, can be computed by reparameterizing $p(Z_i|X,\mathrm{cp}_X)$ to reach the functional form as in (2.55).

For reference, the variational joint distribution over X, Y_i and λ for this approach to modeling mixtures will look as follows:

$$q(X,\lambda,\boldsymbol{Y}) = q_\lambda(\lambda)\prod_i q_{X|\lambda}(X|\lambda = i)\prod_i q_{Y_i}(Y_i). \tag{2.58}$$

It is also possible to use a deterministic function [32] to define a generative model for mixtures, but it leads the mixture weight to be highly peaked around a single component, which makes mixtures less flexible.

2.4.6 Limitations of Variational Message Passing

There are many important distributions, that are widely used, but do not fit conjugate-exponential properties, such as logistic regression, Boltzmann machines and independent component analysis, they thus cannot be directly applied within the VMP framework. However, there are various approximate solutions that extend VMP to non-conjugate models [16], which is yet another reason in favor of using this type of approximate Bayesian inference.

2.5 Online Variational Inference

For all the methods described in preceding sections it was assumed that all the observations are available at any moment. However, in some settings and, in particular, in signal processing tasks such as speech processing, the observations are received *incrementally*. The methods that are capable of handling such settings are called *online*, in distinction from *batch* methods. For a short (yet comprehensive) description of existing approaches to online Bayesian inference, one can refer to [14].

Let $\boldsymbol{V}_{1:T}$ be a sequence of observations $\boldsymbol{V}_1, \ldots, \boldsymbol{V}_T$, similarly $\boldsymbol{H}_{1:T}$ denotes a sequence of hidden variables, and $\boldsymbol{H}_t = \boldsymbol{H}_{t:t}$. The goal of the inference then can be formulated as follows. Given a sequence of observations $\boldsymbol{V}_{1:t}$ up until the current moment, get the current estimate for hidden variables $\boldsymbol{H}_t$. In other words, we are looking for the posterior distribution over hidden variables:

$$p(\boldsymbol{H}_t|\boldsymbol{V}_{1:t}) \propto p(\boldsymbol{V}_t|\boldsymbol{H}_t, \boldsymbol{V}_{1:t-1})p(\boldsymbol{V}_{1:t}). \tag{2.59}$$

2.5.1 One-step Estimation

A straightforward approach is to use *one-step estimation* - that is using the posterior from the previous step $p(\boldsymbol{H}_t|\boldsymbol{V}_{1:t})$ as a prior for the current step. In many cases, however, it will not be a possible because the posterior and prior are from different families of distributions. Thus instead of using the real posterior, an approximate posterior $\widetilde{p}(\boldsymbol{H}_t|\boldsymbol{V}_{1:t}) \approx p(\boldsymbol{H}_t|\boldsymbol{V}_{1:t})$ is used. Typically the approximation distribution is selected from the exponential family (2.4.2). There are two popular approaches to fit the approximation: probability fitting and moment matching. The former searches $\widetilde{p}(\boldsymbol{H}_t|\boldsymbol{V}_{1:t})$ by optimizing some distance function between the approximate and the true posterior. Moment matching selects $\widetilde{p}(\boldsymbol{H}_t|\boldsymbol{V}_{1:t})$ such that its moments match the moments of the true posterior.

However, this approach is not universal - in some cases can lead to undesirable results, because the approximation is very local, and correlations between variables are lost [14].

2.5.2 Online Variational Bayes

Note that in the batch variational inference we approximate the intractable posterior using a factorized distribution that consists of exponential family distributions. The batch version can be extended to online settings. The difference will lie only in the order in which the computations are made: if in offline settings we have a luxury of iterating through *all the observations* several times, in online settings we use each observation once to update the posterior and then use it. Note that in non-stationary settings it might even be advantageous, since the observations used a long time ago might be non-relevant in the future. In fact, such online variational inference can be considered as a special case of one-step approximation, namely distribution fitting which is done using the KL divergence optimization.

The full derivation of this approach as well as the proof for convergence is given in [29], where it is also demonstrated that this algorithm is in fact a stochastic approximation for the maximization of the variational likelihood.

2.6 Model Selection

Within the Bayesian framework, selection of a model that is the best fit for the dataset of interest, is strongly based on probability theory. Let's say we are provided with a model M that has some parameter set $\boldsymbol{\theta}$, and with a dataset $\boldsymbol{X}$. The following quantity, known as the *Bayesian evidence* of the model, represents the probability of observing the data $\boldsymbol{X}$ given the model M, and is computed as an integral over all possible values of parameters of M:

$$p(\boldsymbol{X}|M) = \int_{\boldsymbol{\theta}} p(\boldsymbol{X}|\boldsymbol{\theta}, M)p(\boldsymbol{\theta}|M) \tag{2.60}$$

Note that e.g. in the description of the Bayes rule (2.1), we do mention M, but it should be always implied that the equations are written down for a specific model, that is currently being considered. For a more comprehensive introduction into Bayesian model selection, see, e.g. [1, Ch.12].

One of the huge advantages of using variational inference is that it allows to *simultaneously* estimate the parameters of the model and the variational likelihood (lower bound), which is an *approximation* of the log-

arithm of the Bayesian evidence for the model [32]. When applying VMP in conjugate-exponential models, the computation of the lower bound can be generalized. In fact, the major part of its constituents are computed along with the updates of approximate posteriors $\boldsymbol{q}^*(H)$ (2.47).

If we assume a fully factorized form of an approximate posterior, just like we did in Section 2.4.1 when discussing VMP, the lower bound on the log-evidence can be written as a sum of individual contributions from all the nodes of the model (both hidden and observed):

$$\begin{aligned} \mathcal{L}(q) &= \sum_i \langle \ln p(X_i|\mathrm{pa}_i) \rangle - \sum_j \langle \log q_j(H_j) \rangle \\ &= \sum_i \mathcal{L}_i. \end{aligned} \tag{2.61}$$

For a single hidden variable H_j, if we substitute the functional form of the exponential distribution (2.37), the contribution to the lower bound will be:

$$\mathcal{L}(H_j) = (\langle \boldsymbol{\phi}(\mathrm{pa}_j) \rangle - \boldsymbol{\phi}_j^*)^T \langle \boldsymbol{u}_j(H_j) \rangle + \langle g_j(\mathrm{pa}_j) \rangle - g_j'(\boldsymbol{\phi}_j^*). \tag{2.62}$$

For an observed variable V_j its contribution is:

$$\mathcal{L}(V_j) = \langle \boldsymbol{\phi}(\mathrm{pa}_j) \rangle^T \langle \boldsymbol{u}_j(H_j) \rangle + \langle g_j(\mathrm{pa}_j) \rangle. \tag{2.63}$$

The equations for specific distributions implemented in our framework are given in the Section 3.2.

3 Framework

In this section, we specify the structure of the developed framework, describe the specific inference equations required for inference and estimation of the variational likelihood, and also identify some important issues, such as initialization and possible numerical problems. When compared to the original VMP, described in [32], we, in addition to univariate variables, provide messages and computation identities for multivariate distributions, including multivariate Gaussian (A.2) and Wishart distributions (A.4).

It is probably worth explaining why the following set of the distributions is used. Gaussians are unimodal continuous distributions, and they are standard for describing continuous data. There are various reasons for that, e.g. the statement of the *central limit theorem*, that for a sufficiently large number of i.i.d. observations, their distribution will be approximately Gaussian [3, Ch.2.3, p.78]. Gaussian mixture models are very flexible, and, as we will see in Sections 4 and 5, can be used to solve rather diverse problems. Gamma distributions and Wishart distributions are used since they are conjugate-prior to Gaussian and multivariate Gaussian respectively, and allow us to model precision (matrices) of the quantities of interest. Discrete distribution with its conjugate prior, Dirichlet, are used mainly to support mixture models, even though ultimately they can be used when solving other tasks that involve discrete variables (for a description of the distributions, see Appendix A or refer to [1, Ch. 8.3, p.163]).

3.1 Basic Structure

The framework is based on the variational message passing approach for conjugate-exponential family. The basic building block is a random *variable* - it can be either hidden or observed, and should be of specific type. The type of variable basically specifies the distribution, according to which the variable is assumed to be sampled from, specifies what kind of parents it supports, and also defines the form of its moments and parameters.

Below you will find a list of supported variables and which parents they support, in the brackets, for convenience, we specify the name of the corresponding class name in the framework software implementation:

- univariate and multivariate Gaussians, support mean (multivariate or scalar Gaussian) and precision (univariate gamma or Wishart) parents (`Gaussian`, `MVGaussian<TMean, TPrec>`);
- (univariate) gamma distribution, no parents supported, constant scale and rate (`Gamma`);
- (multivariate) Wishart distribution, no parents supported, constant number of degrees of freedom and scale matrix (`Wishart`);
- (multivariate) Dirichlet distribution, no parents supported, constant concentration parameters (`Dirichlet`);
- categorical distribution, forced to have Dirichlet parent, parameterized using a vector of log-probabilities (`Discrete`);
- mixtures of Gaussians (multivariate or univariate) has three parents: array of priors on means, array of priors on precision, and discrete selector variable, specifying mixture weights (`MoG<TDistr, VariableArray<TMean>, VariableArray<TPrec> >`).

Note that the selection of parents is not arbitrary, the distribution of a parent should necessarily be conjugate to the distribution of its child, for variational message passing to be applicable [32].

The generative model itself can be specified by a network of variables. Such a network can be easily mapped from a factor graph, for more specific examples, see Sections 4 and 5.

The inference process on the model can be described by identifying a sequence in which variables should update their posterior distributions and recompute messages to each other. This sequence is then simply repeated along with the computation of the contributions to the variational lower bound, up until this lower bound converges. The results of the inference can then be obtained by observing the values of moments or parameters of the approximate posterior distributions of variables.

3.2 Inference Equations

The process of deriving VMP-based inference equations for exponential-family distributions, once one has identified conjugate prior distributions

on all the parameters, is not very complicated. Yet, this process is tedious, and prone to errors. Unfortunately, in the original manuscripts on variational message passing [32], authors limit to univariate distributions only. Thus here, for all the nodes, supported by our framework, we specify the identities required to compute updates for the approximate posterior distributions, messages to parents (if any), and contributions to the variational lower bound. The derivation itself can be done by writing the considered distribution in the exponential form (see Appendix A) and taking the required expectations. In order to obtain inference algorithm from these identities, one should substitute them to the equation (2.48) to get posterior updates, and to equations (2.62) and (2.63) for variational likelihood. Note that here, we omit the full exponential form of all the distributions, they are all specified in Appendix A, along with the used parameterization.

Univariate Gaussian

$$x \sim \mathcal{N}(\mu, \gamma^{-1}) \tag{3.1}$$

Parameters from parents:

$$\boldsymbol{\phi}(\mu, \gamma) = \begin{bmatrix} \phi_0 \\ \phi_1 \end{bmatrix} = \begin{bmatrix} \langle\mu\rangle\langle\gamma\rangle \\ -\langle\gamma\rangle/2 \end{bmatrix} \tag{3.2}$$

Moments from posterior parameters:

$$\begin{bmatrix} \langle x\rangle \\ \langle x^2\rangle \end{bmatrix} = \begin{bmatrix} \phi_0^*/\phi_1^* \\ (\phi_0^*/\phi_1^*)^2 - \frac{1}{2}(\phi_1^*)^{-1} \end{bmatrix} \tag{3.3}$$

Message to the mean (Gaussian) parent:

$$m_{x\to\mu} = \begin{bmatrix} \langle x\rangle\langle\gamma\rangle \\ -\langle\gamma\rangle/2 \end{bmatrix} \tag{3.4}$$

Message to the precision (gamma) parent:

$$m_{x\to\gamma} = \begin{bmatrix} \frac{1}{2} \\ \frac{1}{2}(\langle x^2\rangle - 2\langle x\rangle\langle\mu\rangle + \langle\mu^2\rangle) \end{bmatrix} \tag{3.5}$$

Normalization from parents:

$$\langle g(\mu, \gamma)\rangle = \frac{1}{2}(\langle\log\gamma\rangle - \langle\gamma\rangle\langle\mu\rangle - \log 2\pi) \tag{3.6}$$

Normalization from posterior parameters:

$$g(\phi^*) = \frac{1}{2}(\log(-2\phi_1^*) + \frac{1}{2}(\phi_0^*)^2/\phi_1^* - \log 2\pi) \tag{3.7}$$

Gamma

$$\gamma \sim \Gamma(a, b) \tag{3.8}$$

Parameters from parents:

$$\boldsymbol{\phi}(a, b) = \begin{bmatrix} \phi_0 \\ \phi_1 \end{bmatrix} = \begin{bmatrix} -b \\ a - 1 \end{bmatrix} \tag{3.9}$$

Moments from posterior parameters:

$$\begin{bmatrix} \langle\gamma\rangle \\ \langle\log\gamma\rangle \end{bmatrix} = \begin{bmatrix} -(\phi_0^* + 1)/\phi_1 \\ \psi(\phi_1^* + 1) - \log(-\phi_0^*) \end{bmatrix} \tag{3.10}$$

where ψ is digamma function (see Appendix B).

Normalization from parents:

$$\langle g(a, b)\rangle = a\log b - \log\Gamma(a) \tag{3.11}$$

Normalization from posterior parameters:

$$g(\phi^*) = a_{\phi^*}\log b_{\phi^*} - \log\Gamma(a_{\phi^*}), \tag{3.12}$$

where $a_{\phi^*} = \phi_1^* + 1$, $b_{\phi^*} = -\phi_0^*$.

Multivariate Gaussian

$$\boldsymbol{X} \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Lambda}^{-1}) \tag{3.13}$$

Parameters from parents:

$$\phi(\boldsymbol{\mu}, \boldsymbol{\Lambda}) = \begin{bmatrix} \phi_0 \\ \phi_1 \end{bmatrix} = \begin{bmatrix} \boldsymbol{\Lambda\mu} \\ -\frac{1}{2}\boldsymbol{\Lambda} \end{bmatrix} \tag{3.14}$$

Moments from posterior parameters:

$$\langle \boldsymbol{u}(\phi^*)\rangle = \begin{bmatrix} \langle\boldsymbol{X}\rangle \\ \langle\boldsymbol{X}\boldsymbol{X}^T\rangle \end{bmatrix} = \begin{bmatrix} \langle\boldsymbol{X}\rangle \\ \langle\boldsymbol{X}\rangle^2 - \frac{1}{2}(\phi_1^*)^{-1} \end{bmatrix}, \tag{3.15}$$

where $\langle \boldsymbol{X} \rangle = \frac{1}{2}(\phi_1^*)^{-1}\phi_0^*$.

Message to the mean (Multivariate Gaussian) parent:

$$m_{\boldsymbol{X}\to\boldsymbol{\mu}} = \begin{bmatrix} \langle \boldsymbol{\Lambda} \rangle \langle \boldsymbol{X} \rangle \\ -\frac{1}{2}\langle \boldsymbol{\Lambda} \rangle \end{bmatrix} \tag{3.16}$$

Message to the precision (Wishart) parent:

$$m_{\boldsymbol{X}\to\boldsymbol{V}} = \begin{bmatrix} -\frac{1}{2}(\langle \boldsymbol{X}^2 \rangle - \langle \boldsymbol{X} \rangle \langle \boldsymbol{\mu} \rangle^T - \langle \boldsymbol{\mu} \rangle \langle \boldsymbol{X} \rangle^T + \langle \boldsymbol{\mu}^2 \rangle) \\ \frac{1}{2} \end{bmatrix} \tag{3.17}$$

Normalization from parents:

$$\langle g(\boldsymbol{\mu}, \boldsymbol{\Lambda}) \rangle = -\frac{1}{2}\mathrm{Tr}(\boldsymbol{\mu}\boldsymbol{\mu}^T\boldsymbol{\Lambda}) + \frac{1}{2}\log \boldsymbol{\Lambda} \tag{3.18}$$

Normalization from posterior parameters:

$$g(\phi^*) = -\frac{1}{2}\boldsymbol{\mu}_{\phi^*}^T \boldsymbol{\Lambda}_{\phi^*} \boldsymbol{\mu}_{\phi^*} + \frac{1}{2}\log |\boldsymbol{\Lambda}_{\phi^*}|, \tag{3.19}$$

where $\boldsymbol{\Lambda}_{\phi *} = -2\phi_1^*$, $\boldsymbol{\mu}_{\phi^*} = \boldsymbol{\Lambda}_{\phi *}^{-1}\phi_0^*$.

Wishart

$$\boldsymbol{\Lambda} = \mathcal{W}(n, \boldsymbol{W}) \tag{3.20}$$

Parameters from parents:

$$\phi(n, \boldsymbol{W}) = \begin{bmatrix} \phi_0 \\ \phi_1 \end{bmatrix} = \begin{bmatrix} -\frac{1}{2}\boldsymbol{W} \\ \frac{1}{2}n \end{bmatrix} \tag{3.21}$$

Moments from posterior parameters:

$$\langle u(\phi^*) \rangle = \begin{bmatrix} \langle \log \boldsymbol{\Lambda} \rangle \\ \langle \log |\boldsymbol{\Lambda}| \rangle \end{bmatrix} = \begin{bmatrix} -\phi_1^*(\phi_0^*)^-1 \\ -\log|-\phi_0| + \psi_D(\phi_1) \end{bmatrix} \tag{3.22}$$

where ψ_D is D-dimensional digamma function (see Appendix B).

Normalization from parents:

$$\langle g(n, \boldsymbol{W}) \rangle = \frac{n}{2}\log|\boldsymbol{W}| - \log\Gamma_D(\frac{n}{2}) \tag{3.23}$$

where $\log\Gamma_D$ is the logarithm of multivariate gamma function (see Appendix B).

Normalization from posterior parameters:

$$g(\phi^*) = \frac{n}{2}\log|\boldsymbol{W}| - \frac{nD}{2}\log 2 - \log\Gamma_D(\frac{n}{2}) \tag{3.24}$$

Dirichlet

$$\boldsymbol{p} \sim \text{Dir}(\boldsymbol{\alpha}) \tag{3.25}$$

Parameters from parents:

$$\phi(\boldsymbol{\alpha}) = \begin{bmatrix} \alpha_1 - 1 \\ \vdots \\ \alpha_D - 1 \end{bmatrix} \tag{3.26}$$

Moments from posterior parameters:

$$\langle u(\phi^*) \rangle = \begin{bmatrix} \psi(\alpha_1) - \psi(\sum_i \alpha_i) \\ \vdots \\ \psi(\alpha_D) - \psi(\sum_i \alpha_i) \end{bmatrix} \tag{3.27}$$

Normalization from parents:

$$\langle g(\boldsymbol{\alpha}) \rangle = \log \Gamma(\sum_i \alpha_i) - \sum_i \log \Gamma(\boldsymbol{\alpha}) \tag{3.28}$$

Normalization from posterior parameters:

$$g(\phi^*) = \log \Gamma(\sum_i (\boldsymbol{\phi}_i^* + 1)) - \sum_i \log \Gamma(\boldsymbol{\phi}^* + \mathbf{1}) \tag{3.29}$$

Discrete

$$\boldsymbol{\lambda} \sim \text{Discrete}(\boldsymbol{p}) \tag{3.30}$$

Parameters from parents:

$$\phi(\boldsymbol{p}) = \begin{bmatrix} \log p_i \\ \vdots \\ \log p_D \end{bmatrix} \tag{3.31}$$

Moments from posterior parameters:

$$\langle u(\boldsymbol{\phi}^*) \rangle = \exp(\boldsymbol{\phi}^*) \tag{3.32}$$

Message to the parent (Dirichlet):

$$m_{k \to \boldsymbol{p}} = \exp(\boldsymbol{\phi}^*) \tag{3.33}$$

Normalization from posterior parameters and from parents are both equal to zero.

Mixture of Gaussians

$$x \sim \text{GMM}(\boldsymbol{\mu}, \boldsymbol{\gamma}, \lambda) \tag{3.34}$$

where $\lambda \in [1, M]$ is a discrete variable, $\boldsymbol{\mu}$ and $\boldsymbol{\gamma}$ are vectors of parameters of base Gaussians.

Parameters from parents:

$$\phi(\boldsymbol{\mu}, \boldsymbol{\gamma}, \lambda) = \sum_{m}^{M} p(\lambda = m)\phi_m(\mu_m, \gamma_m) \tag{3.35}$$

here $p(\lambda = m)$ is the component probability, ϕ_m is the natural parameter vector of the m-th component.

Moments from posterior parameters:

$$\langle u(\phi^*)\rangle = \sum_{m}^{M} p(\lambda = m)\langle u_m(\mu_m, \gamma_m)\rangle \tag{3.36}$$

Message to the selector (Discrete) variable:

$$m_{x\to\lambda} = \begin{bmatrix} \langle \log \mathcal{N}(x|\mu_1, \gamma_1^{-1})\rangle \\ \vdots \\ \langle \log \mathcal{N}(x|\mu_M, \gamma_M^{-1})\rangle \end{bmatrix} \tag{3.37}$$

Messages to mean and precision parents are the same as for the base Gaussian distribution, each multiplied by $p(\lambda = m)$ for the corresponding m.

Normalization from parents:

$$\langle g(\boldsymbol{\mu}, \boldsymbol{\gamma}, \lambda)\rangle \sum_{m}^{M} p(\lambda = m)\langle g_m(\boldsymbol{\mu}, \boldsymbol{\gamma})\rangle \tag{3.38}$$

Normalization from posterior parameters:

$$g(\phi^*) = \sum_{m}^{M} p(\lambda = m) g_m(\phi_m^*) \tag{3.39}$$

3.3 Implementation remarks

The developed framework is written in C++, and has experimental Matlab bindings at the current point. The basic building block is of base class `Variable` that correspond to a random variable of a generative model, either hidden or observed. Each variable class incorporates rules for updating its own approximate posterior distribution using `updatePosterior()` method, and computing its contribution to the variational likelihood using `logEvidence()`, as well as for computing message to parents out of messages received from children and possibly moments of other parents using methods `messageToParents()`.

Each type of `Variable` should specify the structure of its natural parameter vector and natural statistics vector through specification of `Parameters<TVariable>` and `Moments<TVariable>`. For instance, `Parameters<Gaussian>` consist of values of precision and multiplication of mean and precision, and `Moments<Gaussian>` contains the first and the second moment of the distribution.

3.4 Numerical issues

In this section, we describe solutions to numerical issues that arise when applying the update equations described in Section 3.2.

Log-probability Normalization

Discrete distribution is parameterized using log-probability vector. When normalizing this vector for probabilities to sum up to one, one can easily experience numerical overflow. To avoid that, one can use the following identity instead of simply summing up the exponentials of log-probabilities:

$$\sum_i p_i = \ln \sum_i \exp(\ln p_i - \max_i \ln p_i) + \max_i \ln p_i \tag{3.40}$$

Symmetric Positive-Definite Matrices

When computing the normalization of the Wishart distribution, one needs to compute log-determinant of a scale matrix. And the inverse of a precision matrix is required e.g. for the computation of moments for the

multivariate Gaussian distribution (3.15). Both of these can be done more efficiently using Cholesky decomposition [28].

That is, let matrix $\boldsymbol{A}$ be symmetric and positive definite. Then, one can write down the Cholesky decomposition of $\boldsymbol{A}$:

$$\boldsymbol{A} = \boldsymbol{L}\boldsymbol{L}^T \tag{3.41}$$

where $\boldsymbol{L}$ is a lower triangular matrix.

Then, one can compute the inverse and log-determinant using $\boldsymbol{L}$:

$$\boldsymbol{A}^{-1} = (\boldsymbol{L}^{-1})^T \boldsymbol{L}^{-1} \tag{3.42}$$

$$\log |\boldsymbol{A}| = \sum_i \log L_{ii} \tag{3.43}$$

The inverse of a triangular matrix can be computed more efficiently, thus it makes sense to use this approach instead of simply taking the inverse. Moreover, log-determinant computation is not only more efficient, but also helps to avoid numerical overflows.

3.4.1 Initialization

It is worth mentioning, that the results of inference highly depend on the initialization. A common practice is to use non-informative priors [4], which in practice basically means that we set very small initial precisions, and set identical weights for discrete variables.

For mixture distributions, it is important to remember that identical equations are used to compute messages to discrete parents. This means that, if initialized *identically*, one will get *identical* results for each mixture component. In other words, instead of fitting several different components, one will basically fit a single component several times. To avoid that, one should introduce some randomization into initialization. For instance, one could randomly assign each observation to an arbitrary component. Another common approach is to use the results of some basic clustering algorithm, such as k-means [4], or even the results of EM on a subset of the data [27].

3.5 Existing Frameworks

Bayesian inference is an active field of research, yet, there are not so many mature frameworks that would have comprehensive documentation and stable implementation. Here we describe several frameworks that we found most relevant to problems related to our target application area.

Bayes Blocks [30] is a cross-platform and open-source C++/Python framework that is based on variational Bayesian inference. The theoretical background is quite close to variational message passing [32], but authors focus more on hierarchical models, and thus allow non-conjugacy in models, sacrificing an ability to compute the variational likelihood analytically (they compute it in iterative fashion instead). Nevertheless, this provides more flexibility on model structure. For example, it is possible to build hierarchies on variance variables, while in the "classical" variational message passing it is not possible due to the fact that the prior of one of the parameters of gamma distribution is not in conjugate-exponential family (there are, however, ways to tackle such problems [16]). The main shortcoming of this framework is that it only supports univariate distributions, which, considering our target application in signal processing, is likely to be not sufficient.

Infer.NET [24] is a Microsoft's framework, it is very rich in terms of supported variable types, and implements several state-of-the art inference algorithms, namely expectation propagation, variational message passing and Gibbs sampling, though it is not possible to use a combination of those simultaneously. One of the framework's unique and useful features is its ability to do code generation for the models: this can improve speed dramatically. Unfortunately, it is not open-source, which makes it hard to modify it and implement non-standard models (such as the one described in Section 5) that do not meet conjugate-exponential requirement strictly. Moreover, it is only limited to a single platform and makes it nontrivial to integrate with non-**.NET** application.

BUGS is a cross-platform framework which, however, only supports sampling methods for inference, which means it will require a lot of computation power.

There are also quite a few frameworks that focus on discrete models, such as, for instance, **libDAI** [26], but they are out of the scope of our project.

The summary of the framework comparison described above is given

Framework / Criteria	Infer.NET	B.Blocks	BUGS	libDAI
Continuous	+	+	+	-
Multivariate	+	-	+	+
Variational	+	+	-	-
Online	-	+	-	-
Open-Source	-	+	+	+

Table 1: Comparison of existing frameworks for Bayesian modeling.

in the Table 1. In a nutshell, five criteria were used: support for continuous and multivariate distributions, support for variational inference and online inference, and whether the framework is open-source. Continuous and multivariate distributions are important due to the selected application area, signal processing. Variational inference is the method that we find the most appealing due to the reasons discussed in Section 2.3. Online inference can also be important when working with time-variant processes, which can be common in signal processing [31]. The latter criteria, whether the project is open-source, might seem inessential, but we believe that the possibilities of extending a framework strongly correlate with the availability of the sources.

After considering the existing frameworks and toolboxes, a decision have been made to create our own framework, written in C++, which can provide us and possible users with freedom to implement any inference method we find appropriate, and, moreover, makes the computational process more transparent.

4 Validation

In this section, we show how the framework can be applied to several basic models, in order to validate that the underlying approximation scheme and model selection criteria indeed work as expected and that the implementation is correct.

4.1 Learning a Gaussian

We start with a simplistic example - learning parameters of a univariate Gaussian from data. The generative model can be described using the factor graph depicted at the Figure 7. Observed data x_n, $1 \leq n \leq N$ is assumed to be a sequence of N independent observations, each sampled from a Gaussian with mean μ and precision γ. Both μ and γ are assumed to have a priors, Gaussian $\mathcal{N}(m, \beta^{-}1)$ and gamma $\Gamma(a, b)$ correspondingly.

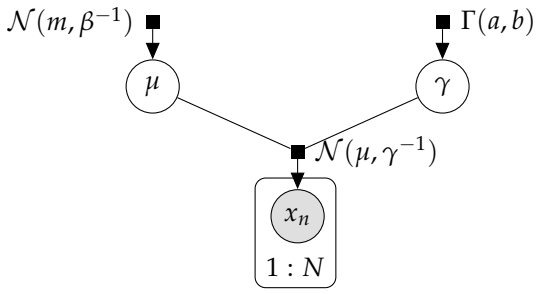


Figure 7: Factor graph for a sequence of samples from a Gaussian.

Within the framework such a model can be specified using an array of Gaussian variables, each having two parents - a Gaussian and a gamma, corresponding to the plate $n = 1 : N$ in the graph that contains independent observations x_n. In order to construct this node, we need to provide it two parents μ and γ of type `Gaussian` and `Gamma` correspondingly, which in their turn could need their own parents - priors for hyperparameters μ_0,γ_0, which in this case are simply constants.

Setup and results

For mean hyperparameters we used $\mu_0 = 0, \beta_0 = 0.01$, and for precision $\alpha_0 = 0.001, \beta_0 = 0.001$. The inference converges to $\widehat{\mu} = -3.3633$,$\widehat{\sigma^2} = 4.1754$, i.e. both mean and covariance are inferred correctly. The value of variational likelihood is $\mathcal{L} = -225.8488$. Note that in fact we infer the *distributions* over parameters rather then points estimates of parameters.

4.2 Learning Mixture Models

In this section we describe the application of the framework to learning and model comparison for a very popular and powerful model - Gaussian mixtures. In Section 2.4.5 we have described the theoretical background of approaching mixture distribution learning for conjugate-exponential models within the VMP framework. A factor graph for Gaussian mixture models is given in the Figure 8. Again, we model observed data as a sequence of independent observations x_n, $1 \leq n \leq N$. Each observation is drawn from one of the Gaussian components, a particular component is determined by a categorical variable λ_n. The probability of $\lambda_n = k$ is determined by an element $\boldsymbol{\pi}_k$, which in its turn is drawn from a Dirichlet distribution. All the described nodes are available in our framework, and they can be simply mapped from the factor graph. One thing worth mentioning is that different priors can be used for precisions of the Gaussians: in case of univariate it should be scalar gamma $\Gamma(a, b)$ distribution, and Wishart distribution $\mathcal{W}(n, \boldsymbol{W})$ (A.4) for multivariate data.

Setup and results

Here we describe the results produced by our framework in comparison to maximum likelihood EM, and also demonstrate how the lower bound can be quite effectively used for model comparison purposes. We use the log-likelihood of the data to compare the goodness of fit, and thus do not expect variational approximation to outperform maximum likelihood method, since it is optimized for the target function, but we expect the VMP-based approximation to perform similarly to the alternative.

We used the datasets described in [4], they include both synthetic and real data:

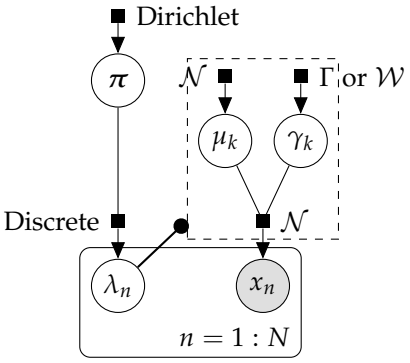


Figure 8: Factor graph for a mixture of Gaussians.

- 600 data points from a mixture of 5 Gaussians (means: [0, 0], [3, -3], [3, 3], [-3, 3], [-3, -3], covariances: [1, 0; 0, 1], [1, 0.5; 0.5, 1], [1, -0.5; -0.5, 1], [1, 0.5; 0.5, 1], [1, -0.5; -0.5, 1]);
- 900 data points from a mixture of 3 Gaussians (means: [0, -2], [0, 0], [0, 2], identical covariance [2, 0; 0, 0.2]);
- 1000 data points from a mixture of 3 Gaussians (identical mean: [0,0], covariances: [1, 0; 0, 0.2], [0.02, -0.08; -0.08, 1.5], [0.9, 0.4; 0.4, 0.2]);
- ”Enzime“, ”Acidity“ and ”Galaxy“ one-dimensional datasets
- ”Old faithful“ two-dimensional dataset.

As described in 3.4, the initialization is quite important. We initialized means using a basic clustering algorithm - *k-means* (see, e.g. [3, Ch.9.1]), and broke symmetry by initializing discrete selector variables with values also produced by the same clustering algorithm. The results for two-dimensional datasets are given in the Figure 10, the ellipses denote an area within two standard deviations of the fitted components. Zero means hyperparameters were used for Gaussian means, and the following hyperparameters were used

for $\Gamma(a, b)$:

$$a_0 = b_0 = 0.001,$$

for $\mathcal{W}(n, \boldsymbol{W})$:

$$n = 2,$$

$$\boldsymbol{W} = 0.001\boldsymbol{I},$$

for Dirichlet($\boldsymbol{\alpha}$):

$$\boldsymbol{\alpha}_0 = \mathbf{1}.$$

The value of log-likelihood of data is given in Table 2. As one can see, variational methods are very close to the results produced by maximum-likelihood based method, this means that the inference of the framework is quite reasonable for this task. The results for VMP-based inference on one-dimensional datasets for a fitted model with 3 Gaussian mixtures is provided in the Figure 9. Visually, one can see that the inferred distributions indeed fit the data quite well.

Dataset	**VMP**	**ML EM**
5 mixtures	-2550.26	-2549.09
3 mixtures (1st)	-3106.58	-3106.36
3 mixtures (2nd)	-1870.46	-1870.46
Old faithful	-1124.47	-1120.05
Enzime	-48.1637	-47.8271
Acidity	-179.4447	-178.7574
Galaxy	-212.7597	-212.1368

Table 2: Log-likelihood of data for VMP and ML EM algorithms.

One of the main advantages of using variational methods is the fact that it is possible to simultaneously estimate both model parameters and the variational likelihood, which can be used as a model selection score. The values of the variational likelihood for models with different number of mixtures are given in the Figure 11. As one can see, variational likelihood for all the datasets indeed is the highest for the most reasonable number of mixture components (see Figure 10). Thus, even though this score is an approximation for the true data likelihood, it is still possible to use it as a model selection criteria.

In the Figure 12 we demonstrate how the value of the variational likelihood changes as the inference process converges, it has an important property it never decreases [32]. It can be used as a necessary condition

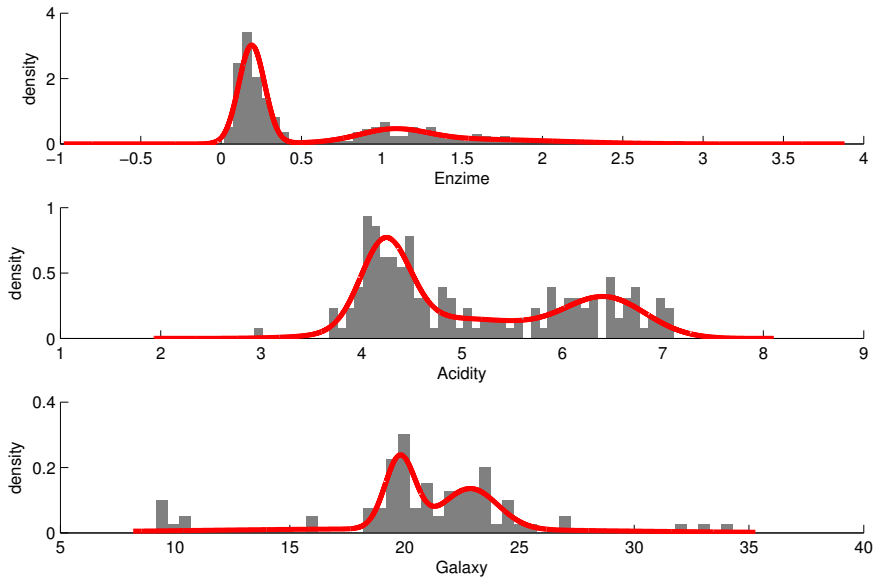
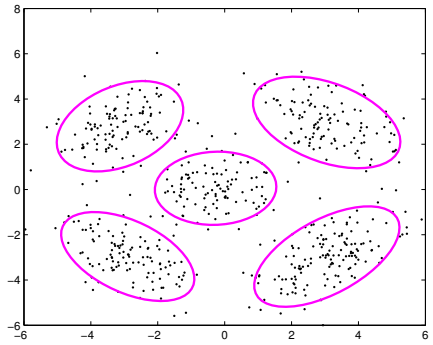
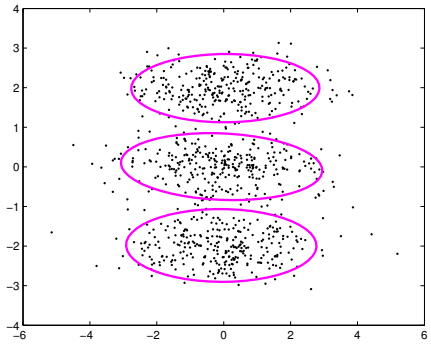


Figure 9: Results of VMP-based inference for one-dimensional datasets.

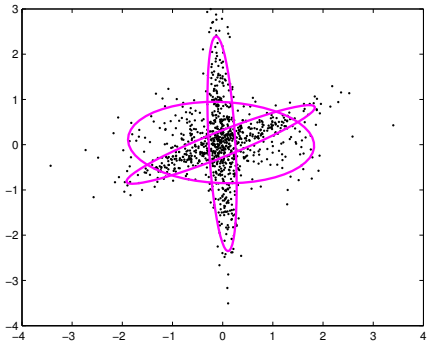
for the correctness of the inference algorithm: in case the inference implementation is correct, the variational likelihood will always increase (or stay the same, which is equivalent to convergence).



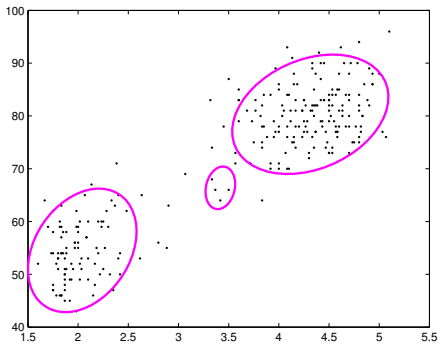
(a) A mixture of 5 Gaussians.



(b) A mixture of 3 Gaussians (1st).

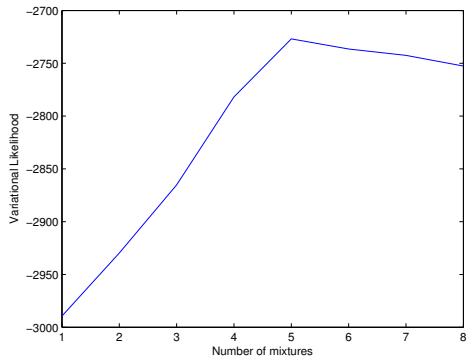


(c) A mixture of 3 Gaussians (2nd).

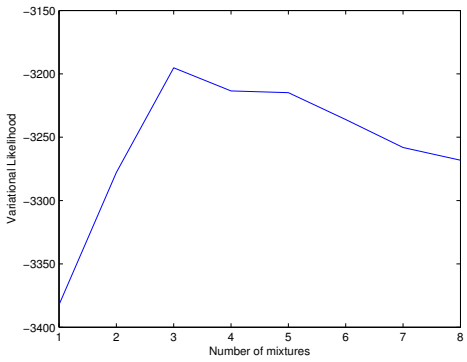


(d) ”Old faithful“.

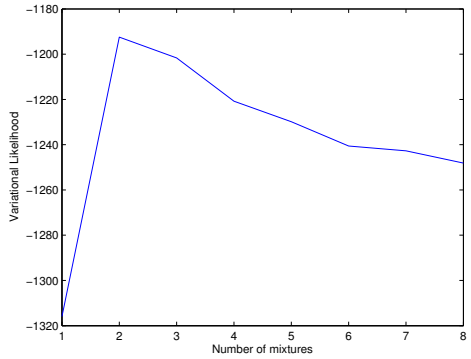
Figure 10: Results of VMP inference for two-dimensional datasets.



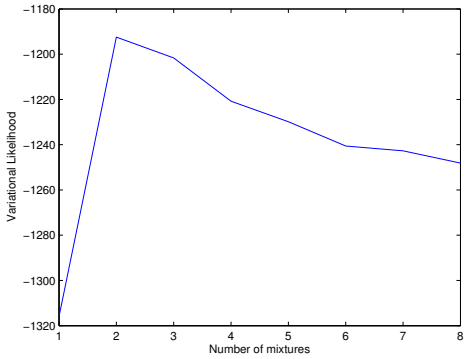
(a) A mixture of 5 Gaussians.



(b) A mixture of 3 Gaussians (1st).



(c) A mixture of 3 Gaussians (2nd).



(d) "Old faithful" dataset.

Figure 11: Variational lower bound for various number of mixture components.

4.3 Image Segmentation

One of the many real-world applications of Gaussian mixtures is image segmentation, which is one of the most challenging tasks in image processing, and also very useful, since it provides data for quantitative analysis [27]. Here we will demonstrate that our framework can be applied to this problem as well.

We are following an approach similar to the one described in [27]. A small subset of images from BSDS500 dataset [19] were used. An image is modeled as a sequence of independent pixels, each encoded as a vector of five dimensions: three dimensions encoding color and two values for

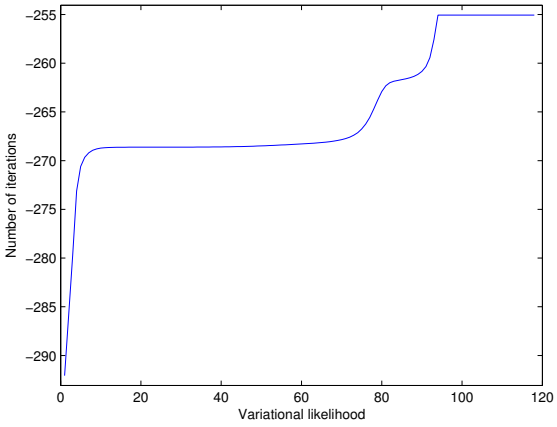


Figure 12: The evolution of the variational lower bound. "Galaxy" dataset.

position of the pixel on the original image. The values were scaled so that for every dimension numbers fall into interval from -1.0 to 1.0.

The results of image segmentation for various numbers of components in the model and different sample images is provided in the Figure 13. The segmented images themselves are obtained by setting each pixel to the color corresponding to the mean of the Gaussian component that has the highest a-posteriori probability of generating this pixel.

For quantitative estimation of the method's quality, the peak signal-to-noise ratio (PSNR) was used, computed between the original image to the result of segmentation.

$$\mathrm{PSNR} = 10 \log_{10} \left(\frac{255^2}{\mathrm{MSE}} \right), \tag{4.1}$$

where MSE is mean squared error, which we computed as an average of three MSE-s for color components.

The averaged values of PSNR for two approaches - classical maximum likelihood Gaussian mixture training and approach using our variational message passing framework, are provided in the Table 3. As one can see, variational approach outperforms the ML-based inference. Moreover, for some settings ML-based approach was suffering from singularities, and was not able to produce any results at all, whereas for VMP this

(a) Original. (b) Original. (c) Original.

(d) 3 components. (e) 3 components. (f) 3 components.

(g) 5 components. (h) 5 components. (i) 5 components.

(j) 8 components. (k) 8 components. (l) 8 components.

Figure 13: Segmentation results for VMP-based inference.

problem never appeared. In absolute numbers, improvements are not as high as, for example, described in [27], most probably, this is because we did not follow the same multiple-stage initialization procedure for hyperparameters, which was described by the authors.

In the Figure 14, values of variational likelihood and PSNR for mixtures with various number of components are provided. As one can see, variational lower bound strongly correlates with the selected quality mea-

Image	VMP	ML EM
Swan, 3 components	6.8551	6.8550
Zebra, 3 components	8.8207	8.8202
Ark, 3 components	10.9734	10.9702
Swan, 5 components	6.8583	-
Zebra, 5 components	8.8207	8.8207
Ark, 5 components	10.9741	10.9721
Swan, 8 components	6.8587	-
Zebra, 8 components	8.8225	8.8221
Ark, 8 components	10.9741	10.9721

Table 3: PSNR of data for VMP and ML EM. No data means EM resulted with singular covariance matrix.

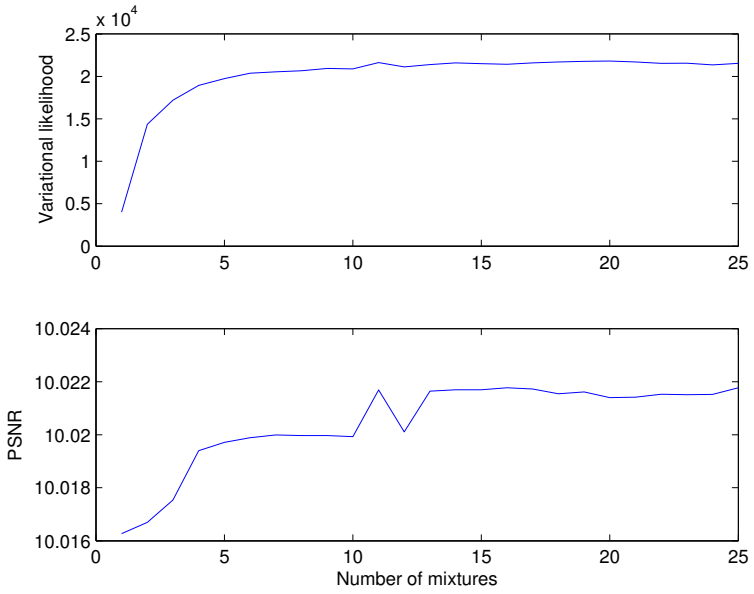


Figure 14: Variational likelihood and PSNR score for models fitted on the ”Swan“ image.

sure, and, thus, for this task, it is quite reasonable to use it as a model selection criteria.

5 Case Study: Noise Reduction for Speech Processing

When it comes to probabilistic inference, deterministic approximations, such as variational message passing or expectation propagation, mostly rely on certain structure of the probabilistic model. The distributions allowed by the method typically should be of a specific family and only certain forms of dependencies between variables are allowed. For instance, for variational message passing to be applicable, the model should be conjugate-exponential.

The goal of a researcher is to come up with a representative model of the process of interest, and then make reasonable approximations such that the selected inference engine can be used to solve the problem. Unfortunately, it is not always possible to come up with a reasonable model with the restrictions imposed by inference method. The reason for that could be rather fundamental, that is, a model requires a distribution that is impossible to represent using types of variables supported by the selected inference approach. For instance, it is not possible to represent a sigmoid using exponential family distribution [16], and thus variational inference is not directly applicable for modeling this type of variables. However, for some models, it is possible to overcome this by providing acceptable approximations [16], or by implementing inference for unsupported parts of the model as ”black boxes“, still using advantages of the framework for the remaining parts.

In this section, we describe how an important problem in speech processing: noise reduction, can be tackled using our framework. Note that we are not trying to suggest an algorithm that could compete with the state-of-the-art, rather, it is more of investigation to find out what kind of difficulties might arise when approaching a real-world problem, and how they can be solved within the framework. The motivation is an ability to design new signal processing algorithms using probabilistic framework, with a unified way of doing inference and, possibly, model comparison.

We start with a general description of modeling of speech signals with noise, then give a short overview of relevant existing work on noise reduction methods in general, and Bayesian speech denoising in particular, and finally describe the approach we have decided to pursue, finishing with its evaluation.

5.1 Modeling Speech Signals

It is typically assumed that the noisy speech signal y consists of the clean speech signal s which is distorted by additive noise n signal.

$$y(t) = s(t) + n(t), \tag{5.1}$$

where t is the sampling time unit index. The clean speech and noise signal are assumed to be statistically independent, which is rather reasonable since in the real world the speech signal and noise are most of the time generated by different and independent sources. An example of the speech signal in time domain, and the same signal corrupted with white noise are shown in the Figure 15.

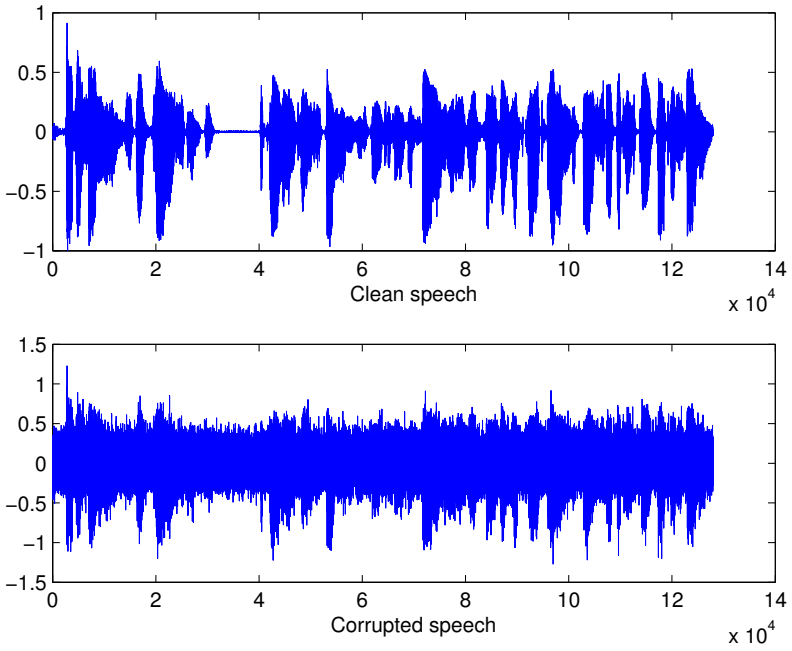


Figure 15: Speech signal in time domain.

The relationship (5.1) can also be rewritten in the frequency domain:

$$Y(k) = S(k) + N(k), \tag{5.2}$$

where k is the frequency index, and Y, S and N are coefficients of the Fourier transform of the original time-domain signals. In speech enhancement, the time-domain signal is usually divided into overlapping windows of a fixed size (typically the length around 30ms), using one of the standard windowing functions (e.g. Hanning window), with each window transformed into a fixed number of frequency bins using discrete Fourier transform (DFT) [31]. Many speech processing algorithms work in frequency domain [20], and the resulting signal can be obtained by applying the corresponding inverse transformations. One can see an example of speech signal in the frequency domain in the Figure 18.

5.2 Formal Problem Definition

The problem of noise reduction can be approached from two points of view. The first one focuses on obtaining the estimate of the clear speech directly from the results of probabilistic inference. The second approach is to estimate the noise (power), and then apply an approach such as spectral subtraction to get the speech signal. To some extent, the second approach is valuable since measures of speech intelligibility might be rather ambiguous, thus making it hard to estimate the performance of the algorithm, while comparing SNR is more-or-less straightforward. However, there is not much difference for the modeling stage, and, in fact, the model might incorporate both speech and noise, making it possible to estimate them simultaneously. In this work we will stick with the estimation of the clean speech.

The problem can be considered both as unsupervised and supervised learning. In unsupervised settings, the goal of noise reduction is, given a sequence of noisy samples $\{y_t, t = 1 \dots T\}$, get estimates of the clean speech signal $\hat{s}_t$. In supervised settings, the we are given a training set of *pairs* $\{(y_t, s_t) | t = 1 : T_s\}$ of noisy samples and corresponding clean speech samples, and the goal is to get estimates of the clean speech for samples outside of the training set. It is also possible that we also have a training set for noise $\{(y_t, s_t), t = 1 : T_n\}$. Note that by samples here one can understand any kind of representation of signal, e.g. frequency or time-domain representation.

Note that, in online settings, the samples are coming incrementally, and the number of them is unbounded, i.e. T can go into infinity.

5.3 Noise Reduction Algorithms

Given the representation of the speech signal as described above, the goal of noise reduction is then to get an estimate of the clean speech signal. For instance, in the DFT domain the estimate of the clean speech spectral coefficients - $\widehat{S}(k)$ - is the value of interest, where k is the index of a frequency bin. This estimate can be obtained by applying a spectral gain function $G(k)$ to the noisy signal $Y(k)$:

$$\widehat{S}(k) = G(k)Y(k) \tag{5.3}$$

where k is the index of a frequency bin. The gain function varies from one algorithm to another.

Spectral subtraction is probably one of the most popular methods for noise reduction. The idea is relatively straightforward: the *average* magnitude spectrum of noise is subtracted from the magnitude spectrum (similar formula can be written down for power domains):

$$\widehat{S}(k) = Y(k) - \alpha(k)\overline{N}(k), \tag{5.4}$$

where α controls the amount of noise to be subtracted. It also can also be expressed using the following gain function:

$$G(k) = (1 - \alpha(k))\frac{|\overline{N}(k)|}{|Y(k)|} \tag{5.5}$$

In the real-world settings $G(k)$ can be further modified to avoid negative values and smooth the output for lower signal-to-noise ratio values [31].

Wiener filter is a least mean squared error (LSE) filter. In the frequency domain for the speech signal the gain function of the filter will look as follows:

$$G(k) = \frac{P_s(k)}{P_s(k) + P_n(k)} \tag{5.6}$$

where $P_s = |S|^2$ and $P_n = |N|^2$ are speech and noise spectral powers.

The **minimum mean squared error (MMSE)** estimation is a Bayesian approach that uses Bayes rule to infer the posterior expectation of the spectral magnitude given the noisy signal likelihood and the clean speech prior. The canonical variation, Ephraim-Malah suppression rule, uses a complex Gaussian for the noisy signal likelihood distribution and a Gaussian as the clean speech prior.

Note that all the methods described above are not complete in a sense that they all rely on an estimate of the noise signal - either noise power or magnitude. The noise estimation is a non-trivial problem on its own, especially in the settings when the noise is non-stationary. In [20], authors provide an approach to noise power spectral density (PSD) estimation based on **minimum statistics**. The method of tracking the minimum is based on an important assumption that there are moments in the speech signal during which the speech energy is close to zero (such as pauses between words). Thus, considering a long enough signal fragment one can determine those moments with no speech present and get the noise floor estimate.

5.4 Bayesian Noise Reduction

Significant amount of work has already been done in applying Bayesian methods to speech processing and noise reduction in particular. The problem of noise reduction of its own is, of course, very well studied, but we are focusing on applying probabilistic inference to this problem, particularly, deterministic approximations to inference, since we want to understand how the problem can be approached in the context of our framework.

In [8], authors proposed a generative model for speech, and an algorithm for approximate inference named ALGONQUIN, that is based on variational methods. They are modeling the signal in log-power domain, with both speech and noise represented as mixture of Gaussians: the evaluation was done with 256 and 4 components respectively. They use Taylor approximation to tackle non-linearities that make inference intractable, and then EM-like algorithm is applied for parameter inference. In the following sections we will consider this algorithm within our framework in more detail and show how it can be extended by providing full distributions on parameters, rather then point estimates as it is done in the original work.

Authors of [6] derive an incremental Bayesian approach to estimate noise in the log-spectral domain. They use a similar trick as in [8] to overcome the non-linearity, use Gaussian to approximate the noise log-spectra, and a time-invariant mixture of Gaussians as a clean speech model. This approach seems very compelling for our goals, since it is

online in its nature and tracks the approximate noise prior *distribution*, rather then only point estimates. However, all the parameter update equations are derived from scratch, a difficulty that we would like to overcome by providing a unified framework.

Yet another interesting example of applying Bayesian methods to speech enhancement was demonstrated in [5]. Authors describe an approach for online voice activity detection using variational methods. They do not model speech directly, but rather model a feature that is based on the kurtosis of the linear prediction residual. The interesting part is that authors keep two models in parallel: the one with both speech and noise (represented as a Gaussian mixture) and the one with noise only. They use Bayesian evidence to compare the models and select the one with the highest evidence in online fashion.

The work [10] is interesting because authors estimate both the frequency coefficients and log-spectra of speech. They claim that this is important since the frequency coefficients tend to provide better quality in terms of SNR, while the log-spectra typically provides better recognition rate. Authors show that if the speech log-spectra is assumed to be GMM-distributed, then the frequency coefficients are log-normal distributed, which poorly represents the distribution of the frequency coefficients (which is super-Gaussian). To avoid this problem, they model the connection between log-spectra and coefficients stochastically, and show that it leads to better results. It is also notable that authors actually provided decent baselines in their results for both SNR and recognition error rate, based on state-of-the-art methods, showing that probabilistic modeling can be very beneficial.

5.5 Proposed Approach

In this work, we are discussing the Bayesian way to the noise reduction problem based on [8] and [6] in the context of out framework. First, we provide a generic model assumptions and describe the model within the factor graph framework. Second, we show how the inference can be done on this model for various.

5.5.1 ALGONQUIN Generative Model

Following the additive speech model in spectral domain in the Section 5.1 (note that time indices are omitted):

$$P_y \approx P_s + P_n \tag{5.7}$$

where P_y, P_s and P_n denote the powers of corrupted signal, speech and noise correspondingly.

However, these powers are non-negative, which makes it rather problematic to model using available inference algorithms, since they work well with Gaussians (or mixtures of those). Thus, instead of modeling the powers directly, the log-powers are modeled, which in very basic settings can be considered as approximately Gaussian distributed. Moreover, it is not always clear whether the direct modeling of the spectral coefficients is always profitable [10]. Denoting log-powers for the noisy signal, speech signal and the noise as y, s and n respectively, the (5.7) can be reformulated as:

$$\begin{aligned} e^y &\approx e^s + e^n \\ &= e^s(1 + \tfrac{e^n}{e^s}) \end{aligned} \tag{5.8}$$

And hence:

$$y \approx s + \ln(1 + e^{n-s}) = f_g(s, n) \tag{5.9}$$

One can see, that in the logarithmic domain the relationship is no longer linear, making inference non-tractable. In [8], [6], authors apply Taylor decomposition to make the inference tractable.

Furthermore, we will assume that a-priori speech and noise log-spectra are independent and distributed as mixtures of Gaussians:

$$p(\boldsymbol{s}) \sim \mathrm{GMM}(\boldsymbol{\mu}_s, \boldsymbol{\Lambda}_s, \boldsymbol{\pi}_s) \tag{5.10}$$

$$p(\boldsymbol{n}) \sim \mathrm{GMM}(\boldsymbol{\mu}_n, \boldsymbol{\Lambda}_n, \boldsymbol{\pi}_n) \tag{5.11}$$

The factor graph for this approach is given in the Figure 16. Unfortunately, the non-linearity in the node f_g makes the model non-tractable as is to the variational message passing.

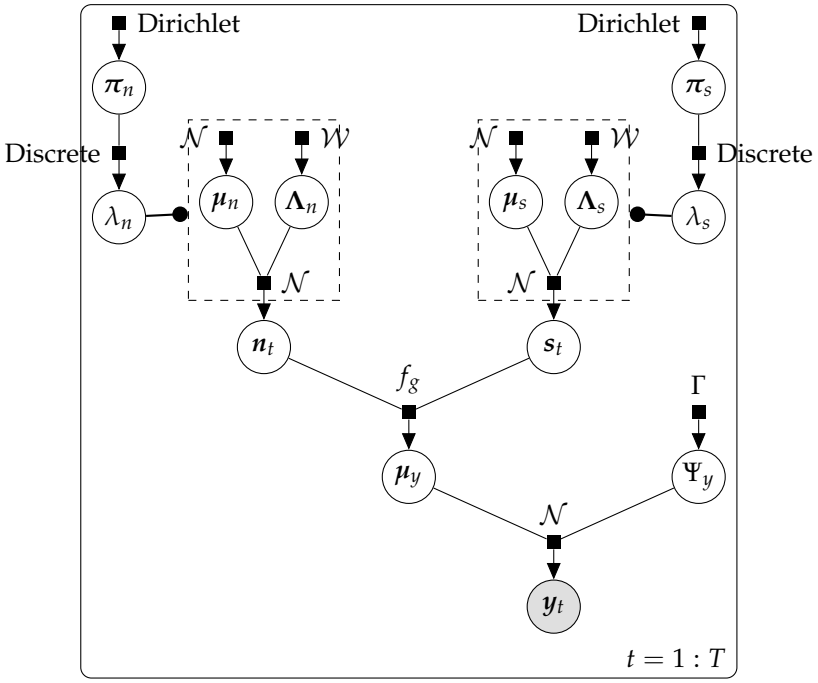


Figure 16: Factor graph for the ALGONQUIN algorithm.

More specificially, because under the following joint distribution [8]:

$$p(\boldsymbol{y}, \boldsymbol{s}, c_s, \boldsymbol{n}, c_n) = \mathcal{N}(\boldsymbol{y}|\boldsymbol{g}(\begin{bmatrix}\boldsymbol{s}\\ \boldsymbol{n}\end{bmatrix}]), \boldsymbol{\Psi}) p(c_s) \mathcal{N}(\boldsymbol{s}|\boldsymbol{\mu}_{c_s}, \boldsymbol{\Lambda}_{c_s}) p(c_n) \mathcal{N}(\boldsymbol{n}|\boldsymbol{\mu}_{c_n}, \boldsymbol{\Lambda}_{c_n}) \tag{5.12}$$

where c_s and c_n denote the speech and noise class index correspondingly, the posterior $p(\boldsymbol{s}, \boldsymbol{n}|c_s, c_n, \boldsymbol{y})$ may have multiple modes, the exact value of the speech and noise estimates $\widehat{\boldsymbol{s}}$ and $\widehat{\boldsymbol{n}}$ are intractable. Thus authors use the following partly factorized variational distribution to the true posterior:

$$p(\boldsymbol{s}, c_s, \boldsymbol{n}, c_n|\boldsymbol{y}) \approx q(\boldsymbol{s}, \boldsymbol{n}) q(c_s, c_n), \tag{5.13}$$

where:

$$q(\boldsymbol{s}, \boldsymbol{n}|c_s, c_n) = \mathcal{N}(\begin{bmatrix}\boldsymbol{s}\\ \boldsymbol{n}\end{bmatrix} | \begin{bmatrix}\boldsymbol{\eta}^s(c_s, c_n)\\ \boldsymbol{\eta}^n(c_s, c_n)\end{bmatrix}, \begin{bmatrix}\Phi^s(c_s c_n) & 0\\ 0 & \Phi^n(c_s, c_n)\end{bmatrix}) \tag{5.14}$$

$$q(c_s, c_n) = \rho(c_s, c_n) \tag{5.15}$$

With $\boldsymbol{\eta}^s(c_s, c_n)$, $\boldsymbol{\eta}^n(c_s, c_n)$ and $\Phi^s(c_s c_n)$ and $\Phi^n(c_s c_n)$ being variational mean and variance parameters, correspondingly, $\rho(c_s, c_n)$ being variational a-priori probability of the combination of mixtures c_s, c_n.

One can then apply e.g. (2.33) to update $q(\boldsymbol{s}, \boldsymbol{n}|c_s, c_n)$ and $q(c_s, c_n)$ in iterative fashion to get the inference algorithm. The complete derivation can be found in [8].

5.5.2 Implementation within the framework

The approach is implemented within a special kind of node - `AlgonquinNode`, that has two parents - mixtures of multivariate Gaussians, corresponding to speech and noise priors (5.10), (5.11). It basically implements the updates of distributions (5.14) and (5.15) as a "black box".

Internally, it implements the variational inference procedure that uses the parameters of the mixture priors and results in estimates for speech and noise for each separate frame. The inference algorithm itself takes place when the framework updates the posterior of the node, which corresponds to the general variational message passing approach.

Variational parameters of the approximate posterior, just like for exponential - family distributions, are defined as `Parameters<AlgonquinNode>`. As for messages to parents, currently the implementation updates only the parent mixture distribution corresponding to the noise, and the speech priors are considered to be constant.

5.6 Evaluation

Here, we wish to demonstrate on a small denoising task, that the proposed approach indeed works, and can be used as a basis for developing more mature models for noise reduction. More than that, we wish to show that fully Bayesian approach is indeed justified, and shows better performance when compared to point estimations, such as maximum likelihood.

5.6.1 Setup

The evaluation was done for the following offline denoising task. We have used a random subset of 29 sentences from the TIMIT database, divided into two parts: 22 and 7, for training and testing accordingly. The frames of length 512 were used for spectral analysis, with a shifting step of size 256. We used the Gaussian white noise from the noisex database to produce corrupted speech signal. Both speech and noise were sampled at the rate $f_s = 16$Hz. Models were pre-trained on the training set before denoising.

Performance of two models were compared, the one that uses maximum likelihood EM and the one based on our framework, based on variational message passing. Two following reference points were used: the corrupted signal without denoising at all and the time-varying Wiener filter. The latter can be considered as an experimental upper bound, since strong speech and noise priors were extracted *locally* from the true clean speech and noise signals. A more complete description of this reference point is given in [10].

The logarithmic error distortion measure is used to compare the estimated signal power $\widehat{P}_s$ to the true power P_s. As in [9], we measure the overall error by the sum of over- and under-estimation errors:

$$\text{LogErr} = \text{LogErrOver} + \text{LogErrUnder} \tag{5.16}$$

where:

$$\text{LogErrOver} = \frac{1}{N_t N_f} \sum_{t=0}^{N_t-1} \sum_{f=0}^{N_f-1} \left| \min\left(0, 10\log_{10}\left(\frac{P_s}{\widehat{P}_s}\right)\right)\right| \tag{5.17}$$

$$\text{LogErrUnder} = \frac{1}{N_t N_f} \sum_{t=0}^{N_t-1} \sum_{f=0}^{N_f-1} \max\left(0, 10\log_{10}\left(\frac{P_s}{\widehat{P}_s}\right)\right) \tag{5.18}$$

where N_t and N_f are the numbers of frames and frequency bins correspondingly.

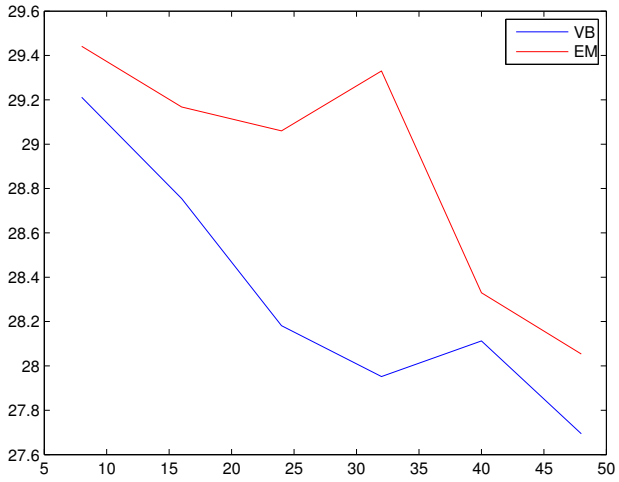


Figure 17: LogErr of variational and ML versions depending on the number of mixtures. Signal-to-noise ratio 5dB.

5.6.2 Results and discussion

In Figure 17, we provide results of denoising on the test set with noise added at SNR of 5dB. As one can see, the approach using our framework consistently outperforms the ML-based method, which is expected. However, the overall even though the absolute SNR values are not very high when compared to the state-of-the-art [10]. The reason for that might be the fact that we have diminished any dependencies between frequencies in the approximate posterior distribution in our ALGONQUIN implementation (i.e. used diagonal matrices in (5.14)).

In Figure 19, we provide the results of applying VMP-based ALGONQUIN to the test set. As one can see, the algorithm performance degrades gracefully, and gives reasonable estimation even for high levels of noise. The same conclusion can be derived from the spectrograms in the Figure 18, of the estimated speech for a sample speech sentence from the test set.

We believe that the main advantage of the described approach is a unified way of model specification, that is, ultimately, any inference algorithm such as ALGONQUIN can be implemented as a separate block within the framework, that exposes standard inputs and outputs in the

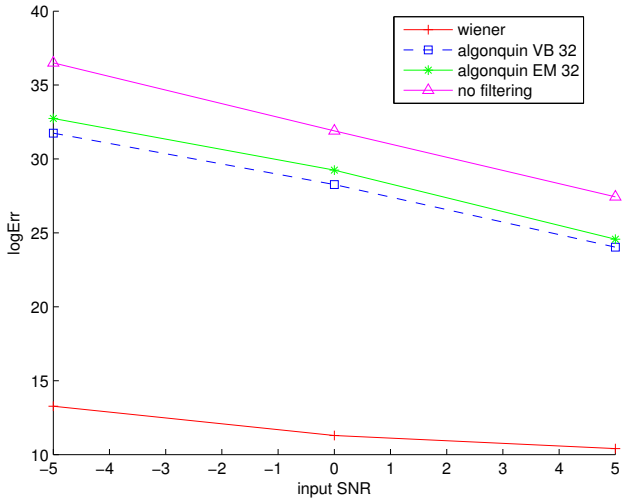


Figure 18: LogErr for VMP-based and ML-based versions of ALGONQUIN.

form of variational messages, i.e. distribution expectations and parameters.

The main disadvantage of the described approach is the fact that it does not capture any time-dependencies. This can be approached in the future by introducing methods for online inference, as e.g. the ones described in Section 2.5, including online methods incorporating "forgetting" approach [14].

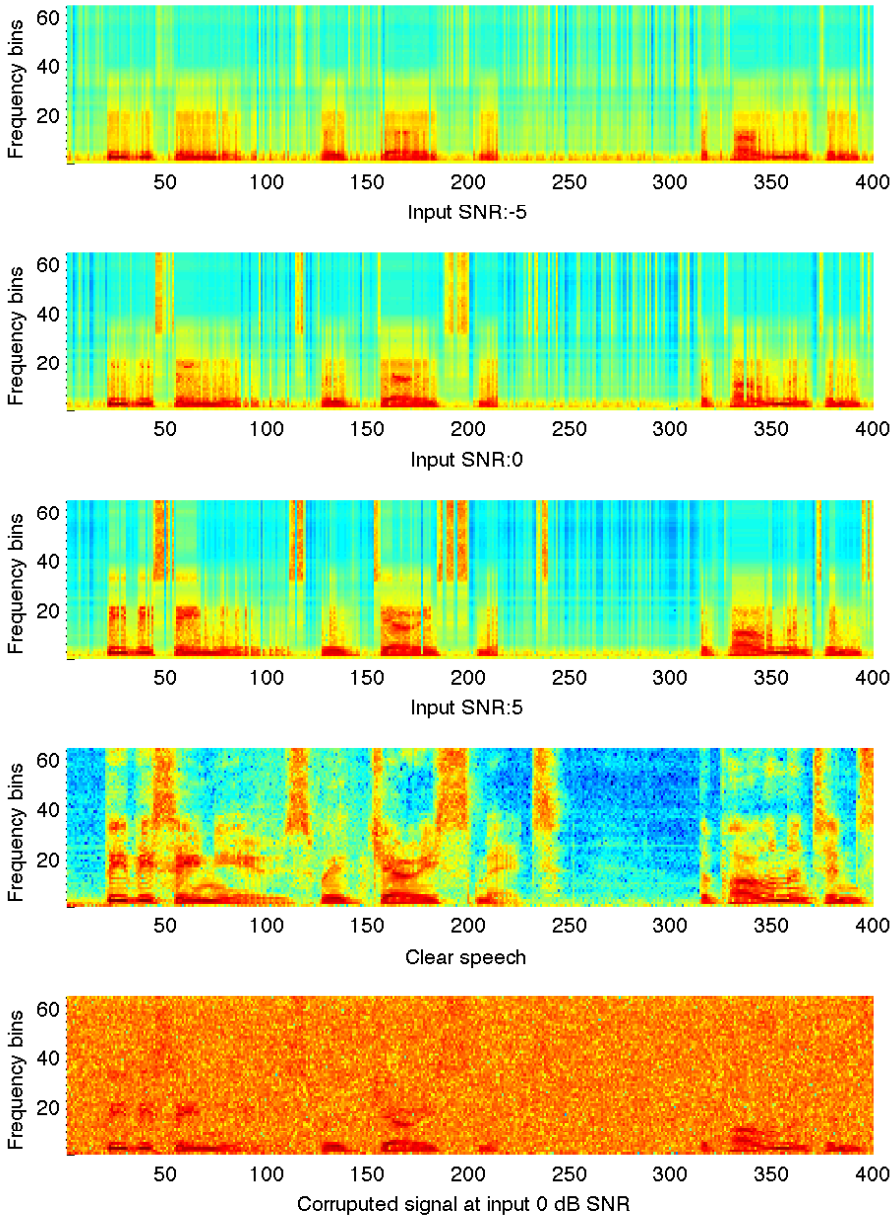


Figure 19: Spectrograms of estimated speech signal at various noise levels.

6 Conclusion

In this work we have presented a machine learning framework based on variational message passing that provides a unified way to specify probabilistic models, do inference and compare models with each other. We have described inference identities for all the supported nodes, and specified the possible numerical computation issues and solutions to those, so that the framework can be reproduced with little effort. Our own software implementation is open-source and is available at `http://github.com/psycharo/factor-graph/tree/vmp`.

Several examples of how the framework can be applied to various problems were shown, such as training Gaussian mixtures and application of those to image segmentation. The results demonstrate, that our VMP-based framework shows similar performance to maximum likelihood method. Furthermore, the results indicate, that the model comparison method used in the framework indeed can be quite efficient to estimate model performance.

We were also interested in applying our framework to a highly complex signal processing task, for which we have selected noise reduction for speech processing. It was demonstrated how one of state-of-the-art Bayesian noise reduction algorithms, ALGONQUIN, can be implemented within the framework. This conforms that the framework is flexible and extensible enough to solve problems some other existing tools do not allow to tackle at this stage. More than that, we have made an extension upon the algorithm by introducing hyperparameters and applying variational methods instead of maximum likelihood estimation, used in the original version of ALGONQUIN. As expected, this makes the model more flexible and less prone to overfitting, and, as one can see from preliminary results, brings slightly better performance.

It would be also interesting to look into performance in terms of computational speed, and compare our framework to the default implementations of algorithms e.g. for training Gaussian mixtures, or for speech denoising using ALGONQUIN.

To summarize, we believe that Bayesian modeling is a promising direction for speech processing research and signal processing in general, and believe that the examples presented in this work, as well as the case study performed, support this belief.

There are several ways of how the framework can be extended. First of

all, adding support for dynamical (time-varying) models would be very useful, especially in case of speech processing tasks. The overview of the methods that can allow that was given during the discussion of online inference in Section 2.5. Second, the number of supported nodes and probabilistic distributions can be extended, including non-linearities and various deterministic nodes: this could increase the flexibility and allow users specify more diverse models. It is also possible to incorporate other methods for probabilistic inference, such as expectation-propagation and sampling methods, which can be useful, again, to expand the number of supported models, especially if several methods can be applied within a single model. This would be a great step forward, since even the most advanced frameworks such as [24] do not yet have that ability.

Appendices

A Exponential Family Distributions

A.1 Univariate Gaussian

$$\ln p(x|\mu,\gamma^{-1}) = \exp\begin{bmatrix} \gamma\mu \\ -\frac{1}{2}\gamma \end{bmatrix}^T \begin{bmatrix} x \\ x^2 \end{bmatrix} + \frac{1}{2}(\ln\gamma - \gamma\mu^2 - \ln 2\pi) \quad \text{(A.1)}$$

where $\mu \in \mathbb{R}$ is mean, $\gamma > 0 \in \mathbb{R}$ is precision.

A.2 Multivariate Gaussian

$$\ln p(\boldsymbol{X}|\boldsymbol{\mu},\boldsymbol{\Lambda}^{-1}) = \begin{bmatrix} \boldsymbol{\Lambda\mu} \\ -\frac{1}{2}\boldsymbol{\Lambda} \end{bmatrix}^T \begin{bmatrix} \mathbf{X} \\ \mathbf{X}\mathbf{X}^2 \end{bmatrix} + \frac{1}{2}(\ln|\boldsymbol{\Lambda}| - \boldsymbol{\Lambda\mu}^2 - \ln 2\pi) \quad \text{(A.2)}$$

where $\boldsymbol{\mu} \in \mathbb{R}$ is mean, $\boldsymbol{\Lambda} \in \mathbb{R}^D \times \mathbb{R}^D$ is precision matrix.

A.3 Univariate Gamma

$$\ln p(x|a,b) = \begin{bmatrix} -b \\ a-1 \end{bmatrix}^T \begin{bmatrix} x \\ \ln x \end{bmatrix} + a\ln b - \ln\Gamma(a) \quad \text{(A.3)}$$

where $a, b > 0, a, b \in \mathbb{R}$ are shape and rate correspondingly.

A.4 Wishart

$$\ln p(\boldsymbol{\Lambda}|n,\boldsymbol{W}) = \begin{bmatrix} -\frac{1}{2}\boldsymbol{W} \\ \frac{1}{2}n \end{bmatrix}^T \begin{bmatrix} \boldsymbol{\Lambda} \\ \ln|\boldsymbol{\Lambda}| \end{bmatrix} + \tfrac{n}{2}\log|\boldsymbol{W}| - \tfrac{nD}{2}\log 2 - \log\Gamma_D(\tfrac{n}{2}) - \tfrac{D+1}{2}\log|\boldsymbol{\Lambda}| \quad \text{(A.4)}$$

where $n > D-1, n \in \mathbb{R}$ is degrees of freedom, $\boldsymbol{W} \in \mathbb{R}^D \times \mathbb{R}^D$ is positive-definite scale matrix.

A.5 Discrete

$$\ln p(\lambda|\boldsymbol{p}) = \begin{bmatrix} \ln p_1 \\ \vdots \\ \ln p_N \end{bmatrix}^T \begin{bmatrix} \delta(\lambda - 1) \\ \vdots \\ \delta(\lambda - N) \end{bmatrix} \tag{A.5}$$

where $\boldsymbol{p} \in \mathbb{R}^N$ is the probability vector, $\sum_{i=1}^N p_i = 1$, $\delta(\cdot)$ is the Dirac delta function.

A.6 Dirichlet

$$\ln p(\boldsymbol{p}|\boldsymbol{\alpha}) = \begin{bmatrix} \alpha_1 - 1 \\ \vdots \\ \alpha_N - 1 \end{bmatrix}^T \begin{bmatrix} \ln p_1 \\ \vdots \\ \ln p_N \end{bmatrix} + \ln \Gamma(\sum_{i=1}^N \alpha_i) - \sum_{i=1}^N \ln \Gamma(\alpha_i) \tag{A.6}$$

where $\boldsymbol{\alpha} \in \mathbb{R}^N$ is the concentration vector, $\alpha_i > 0$.

B Various Functions

Below D stands for the dimensionality of the data, $x \in \mathbb{R}$.

B.1 Dirac delta

$$\delta(x) = \begin{cases} +\infty, & x = 0 \\ 0, & x \neq 0 \end{cases} \tag{B.1}$$

B.2 Gamma

$$\Gamma(x) = \int_0^\infty t^{x-1} e^{-t}\, \mathrm{d}t \tag{B.2}$$

B.3 Digamma

$$\psi(x) = \frac{d}{dx} \ln \Gamma(x) = \frac{\Gamma'(x)}{\Gamma(x)} \tag{B.3}$$

B.4 Multivariate Gamma

$$\Gamma_D(x) = \pi^{\frac{D(D-1)}{4}} \prod_{i=1}^{D} \Gamma(x + \frac{1-i}{2}) \tag{B.4}$$

B.5 Multivariate Digamma

$$\psi_D(x) = \sum_{i=1}^{D} \psi(x + \frac{1-i}{2}) \tag{B.5}$$

C Proofs

C.1 Proof of (2.18)

To prove that (2.18) indeed holds, we can substitute values for $\mathcal{L}(q, \boldsymbol{\theta})$ and $\mathrm{KL}(q||p)$, and get the following equality:

$$\ln p(\boldsymbol{V}|\boldsymbol{\theta}) = \int_{\boldsymbol{H}} q(\boldsymbol{H}) \ln \frac{p(\boldsymbol{V}, \boldsymbol{H}|\boldsymbol{\theta})}{q(\boldsymbol{H})} - \int_{\boldsymbol{H}} q(\boldsymbol{H}) \ln \frac{p(\boldsymbol{H}|\boldsymbol{V}, \boldsymbol{\theta})}{q(\boldsymbol{H})} \tag{C.1}$$

Now, using the rule for the sum of logarithms, we can get:

$$\begin{array}{rl} \ln p(\boldsymbol{V}|\boldsymbol{\theta}) & = \int_{\boldsymbol{H}} q(\boldsymbol{H}) \ln p(\boldsymbol{V}, \boldsymbol{H}|\boldsymbol{\theta}) + \int_{\boldsymbol{H}} q(\boldsymbol{H}) \ln q(\boldsymbol{H}) \\ & - \int_{\boldsymbol{H}} q(\boldsymbol{H}) \ln p(\boldsymbol{H}|\boldsymbol{V}, \boldsymbol{\theta}) - \int_{\boldsymbol{H}} q(\boldsymbol{H}) \ln q(\boldsymbol{H}) \end{array} \tag{C.2}$$

The second and the fourth terms on the right-hand side eliminate each other, and, again, we can apply the rule for sum of logarithms for the remaining two terms (since both integrals are over $\boldsymbol{H}$), and get:

$$\ln p(\boldsymbol{V}|\boldsymbol{\theta}) = \int_{\boldsymbol{H}} q(\boldsymbol{H}) \ln \frac{p(\boldsymbol{V}, \boldsymbol{H}|\boldsymbol{\theta})}{p(\boldsymbol{H}|\boldsymbol{V}, \boldsymbol{\theta}} \tag{C.3}$$

Now, applying the definition of conditional probability to $p(\boldsymbol{H}|\boldsymbol{V}, \boldsymbol{\theta})$:

$$p(\boldsymbol{H}|\boldsymbol{V}, \boldsymbol{\theta}) = \frac{p(\boldsymbol{V}, \boldsymbol{H}|\boldsymbol{\theta})}{p(\boldsymbol{V}|\boldsymbol{\theta})}, \tag{C.4}$$

we will get the following equality:

$$\ln p(\boldsymbol{V}|\boldsymbol{\theta}) = \int_{\boldsymbol{H}} q(\boldsymbol{H}) \ln p(\boldsymbol{V}|\boldsymbol{\theta}) \tag{C.5}$$

which holds in case $q(\boldsymbol{H})$ is a proper probability distribution.

References

[1] David Barber. *Bayesian reasoning and machine learning*. Cambridge University Press, 2012.

[2] J. Besag. Statistical analysis of non-lattice data. *The statistician*, pages 179–195, 1975.

[3] Christopher M Bishop. Pattern recognition and machine learning (information science and statistics). 2007.

[4] Adrian Corduneanu and Christopher M Bishop. Variational bayesian model selection for mixture distributions. In *Artificial intelligence and Statistics*, volume 2001, pages 27–34. Morgan Kaufmann Waltham, MA, 2001.

[5] David Cournapeau, Shinji Watanabe, Atsushi Nakamura, and Tatsuya Kawahara. Online unsupervised classification with model comparison in the variational bayes framework for voice activity detection. *Selected Topics in Signal Processing, IEEE Journal of*, 4(6):1071–1083, 2010.

[6] Li Deng, Jasha Droppo, and Alex Acero. Incremental bayes learning with prior evolution for tracking nonstationary noise statistics from noisy speech data. In *Acoustics, Speech, and Signal Processing, 2003. Proceedings.(ICASSP'03). 2003 IEEE International Conference on*, volume 1, pages I–672. IEEE, 2003.

[7] B.J. Frey. Factor graphs and algorithms. In *PROCEEDINGS OF THE ANNUAL ALLERTON CONFERENCE ON COMMUNICATION CONTROL AND COMPUTING*, volume 35, pages 666–680. UNIVERSITY OF ILLINOIS, 1997.

[8] Brendan J Frey, Trausti T Kristjansson, Li Deng, and Alex Acero. Algonquin-learning dynamic noise models from noisy speech for robust speech recognition. *Advances in Neural Information Processing Systems*, 2:1165–1172, 2002.

[9] Timo Gerkmann and Richard C Hendriks. Noise power estimation based on the probability of speech presence. In *Applications of Signal*

Processing to Audio and Acoustics (WASPAA), 2011 IEEE Workshop on, pages 145–148. IEEE, 2011.

[10] Jiucang Hao, Te-Won Lee, and Terrence J Sejnowski. Speech enhancement using gaussian scale mixture models. *Audio, Speech, and Language Processing, IEEE Transactions on*, 18(6):1127–1136, 2010.

[11] Richard C Hendriks, Richard Heusdens, and Jesper Jensen. Mmse based noise psd tracking with low complexity. In *Acoustics Speech and Signal Processing (ICASSP), 2010 IEEE International Conference on*, pages 4266–4269. IEEE, 2010.

[12] Ralf Herbrich, Tom Minka, and Thore Graepel. TrueskillâĎć: A bayesian skill rating system. *Advances in Neural Information Processing Systems*, 19:569, 2007.

[13] Geoffrey E Hinton, Simon Osindero, and Yee-Whye Teh. A fast learning algorithm for deep belief nets. *Neural computation*, 18(7):1527–1554, 2006.

[14] Antti Honkela and Harri Valpola. On-line variational bayesian learning. In *4th International Symposium on Independent Component Analysis and Blind Signal Separation*, pages 803–808, 2003.

[15] A Jordan. On discriminative vs. generative classifiers: A comparison of logistic regression and naive bayes. *Advances in neural information processing systems*, 14:841, 2002.

[16] David A Knowles and Tom Minka. Non-conjugate variational message passing for multinomial and binary regression. In *Advances in Neural Information Processing Systems*, pages 1701–1709, 2011.

[17] H.A. Loeliger, J. Dauwels, J. Hu, S. Korl, L. Ping, and F.R. Kschischang. The factor graph approach to model-based signal processing. *Proceedings of the IEEE*, 95(6):1295–1322, 2007.

[18] Vikash Kumar Mansinghka. *Natively probabilistic computation*. PhD thesis, Massachusetts Institute of Technology, 2009.

[19] D. Martin, C. Fowlkes, D. Tal, and J. Malik. A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics. In *Proc. 8th Int'l Conf. Computer Vision*, volume 2, pages 416–423, July 2001.

[20] Rainer Martin. Noise power spectral density estimation based on optimal smoothing and minimum statistics. *Speech and Audio Processing, IEEE Transactions on*, 9(5):504–512, 2001.

[21] Christoph Daniel Mathys. *Hierarchical Gaussian filtering*. PhD thesis, Diss., Eidgenössische Technische Hochschule ETH Zürich, Nr. 20909, 2012, 2012.

[22] Thomas Minka. Power ep. *Dep. Statistics, Carnegie Mellon University, Pittsburgh, PA, Tech. Rep*, 2004.

[23] Thomas P Minka. *A family of algorithms for approximate Bayesian inference*. PhD thesis, Massachusetts Institute of Technology, 2001.

[24] Tom Minka, J Winn, J Guiver, and D Knowles. Infer.net 2.5 beta 2. microsoft research cambridge, April 2013.

[25] Tom Minka and John Winn. Gates: A graphical notation for mixture models. In *Proceedings of NIPS*, volume 21, pages 1073–1080, 2008.

[26] Joris M Mooij. libdai: A free and open source c++ library for discrete approximate inference in graphical models. *The Journal of Machine Learning Research*, 99:2169–2173, 2010.

[27] Nikolaos Nasios and Adrian G Bors. A variational approach for color image segmentation. In *Pattern Recognition, 2004. ICPR 2004. Proceedings of the 17th International Conference on*, volume 1, pages 680–683. IEEE, 2004.

[28] William H Press, Saul A Teukolsky, William T Vetterling, and Brian P Flannery. *Numerical Recipes in C: The Art of Scientific Computing (; Cambridge*, volume 683. Cambridge Univ. Press, 1992.

[29] Masa-Aki Sato. Online model selection based on the variational bayes. *Neural Computation*, 13(7):1649–1681, 2001.

[30] H Valpola, A Honkela, M Harva, A Ilin, T Raiko, and T Ostman. Bayes blocks software library, 2003.

[31] Saeed V Vaseghi. *Advanced digital signal processing and noise reduction*. Wiley, 2008.

[32] John Winn and Christopher M Bishop. Variational message passing. *Journal of Machine Learning Research*, 6(1):661, 2006.