Approximate Message Passing on General Factor Graphs using Shallow Neural Networks

Leonhard Hennicke [*1] **Jan Lemcke** [*1] **Rainer Schlosser** [1] **Ralf Herbrich** [1]

Abstract

Factor graphs offer an efficient framework for probabilistic inference through message passing, with the added benefit of uncertainty quantification, which is crucial in safety-critical applications. However, their applicability is limited by the need to analytically solve update equations for factors, which are problem-specific and may involve intractable integrals. We propose to approximate the message update equations of individual factors with shallow neural networks, which we train on data generated by sampling from the respective factor equations, to capture complex factor relationships while maintaining computational tractability.

1. Introduction

Factor graphs (Kschischang et al., 2001) are a powerful tool for probabilistic inference across numerous domains, such as robotics and autonomous vehicles (Dellaert, 2021; Wen & Hsu, 2021), computer vision (Sun et al., 2003) or genetics (Vaske et al., 2009). Their popularity stems from their ability to decompose complex probability distributions into products of simpler local functions, enabling efficient inference through message passing algorithms. A particularly valuable feature of factor graphs is their capacity to quantify uncertainty in their predictions. Despite these advantages, the practical application of factor graph methods faces a significant limitation: the requirement to derive analytical message update equations for each factor in the graph. These equations, which govern how information propagates between variables, must be specifically tailored to each problem domain and can involve complex integrals that may not have closed-form solutions. This analytical intractability restricts the range of probability models that can be solved by factor graphs.

While various approximation methods for factor graphs have been proposed, these usually solve problems such as convergence issues (Zou & Yang, 2022) or computational scaling (Kuck et al., 2020), but not analytical intractability. This necessitates the development of approaches capable of managing intricate factor relationships while preserving the computational and probabilistic advantages that render factor graphs appealing. By transcending this limitation, factor graphs can be appropriately applied in previously unexplored applications. In this paper, we present a novel approach to message passing in factor graphs using shallow neural networks that bridges this gap.

2. Related Work

Some previous works have explored the application of graph neural networks (GNNs) to enhance factor graphs (FGs) by learning to approximate the update functions used during message passing. Kuck et al. (2020) proposed a method to improve the computational scalability of FGs using GNNs, while Satorras & Welling (2021) developed an approach to compensate for inaccurate factor parameters in scenarios where the true underlying distribution is not precisely known. However, both approaches still require knowledge of the factor graph structure and factor types to generate training data from simplified, tractable versions of the target problems. This limitation means they do not directly address the extension of FGs beyond problems with analytically tractable message updates. Numerical integration techniques (Espelid & Genz, 2012) offer another potential solution to handling intractable message updates in FGs. However, these methods typically incur significant computational costs, especially in high-dimensional spaces or with complex factor relationships. Our approach leverages the structure of FGs, explicitly to avoid integration by sampling directly from the joint marginal distribution.

[*]Equal contribution [1]Hasso Plattner Institute, University of Potsdam, Potsdam, Germany. Correspondence to: Leonhard Hennicke <leonhard.hennicke@hpi.de>.

Accepted at Methods and Opportunities at Small Scale (MOSS), ICML 2025, Vancouver, Canada.

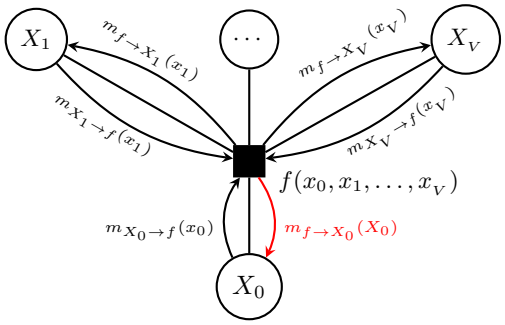


Figure 1: A general factor, where we want to calculate the message $m_{f\to X_0}(x_0)$, after a message from one of the variables $x_1, \ldots, x_V$ has been updated. However, depending on the factor f, this message might be analytically intractable.

3. Approach

Based on known identities for factor graphs, given a general factor $f(x_0, x_1, \ldots, x_V)$ in any arbitrary factor graph (see Figure 1), the message $m_{f\to X_0}$ from that factor to the variable X_0 is defined as

$$m_{f\to X_0}(x_0) = \int_{-\infty}^{+\infty} \cdots \int_{-\infty}^{+\infty} f(x_0, x_1, \ldots, x_V) \cdot m_{X_1\to f}(x_1) \cdot \ldots \cdot m_{X_V\to f}(x_V)\, \mathrm{d}x_1 \ldots\, \mathrm{d}x_V\,,$$

while the message $m_{X_0\to f}(x_0)$ back from the variable X_0 to the factor f is defined as

$$m_{X_0\to f}(x_0) = \prod_{f'\in \mathrm{ne}(X)\setminus\{f\}} m_{f'\to X_0}(x_0) \tag{1}$$

and the marginal probability $p(x_0)$ is defined as

$$p(x_0) = \prod_{f\in \mathrm{ne}(X_0)} m_{f\to X_0}(x_0)\,. \tag{2}$$

Table 2 contains the notation used in this work. To examine our general factor f as an isolated problem within the factor graph during message passing, we consider the point in time where the marginal probability of one of its neighboring variables other than y, i.e., $x_1, \ldots, x_V$, has just been updated by a message passed by another factor. Therefore, we can assume, that all messages $m_{X\to f}(x)$ from neighboring variables $X \in \mathrm{ne}(f)$ to f are given, either from initialization, previous iterations, or based on the known relationship between the marginal $p(x)$ and the messages:

$$p(x) = m_{f\to X}(x) \cdot m_{X\to f}(x)\,. \tag{3}$$

As long as we stay within the distributional family of Gaussians, we can easily calculate an updated message $m_{X\to f}(x)$ from the old message $m_{f\to X}(x)$ and an updated marginal using (3). Therefore, to update $p(x_0)$, after $p(x)$ and $m_{X\to f}(x)$ have been updated for $x \in \{x_1, \ldots, x_V\}$, we need to calculate

$$p(x_0) = m_{X_0\to f}(x_0) \cdot \int_{-\infty}^{+\infty} \cdots \int_{-\infty}^{+\infty} f(x_0, x_1, \ldots, x_V) \cdot m_{X_1\to f}(x_1) \cdot \ldots \cdot m_{X_V\to f}(x_V)\, \mathrm{d}x_1 \ldots\, \mathrm{d}x_V\,. \tag{4}$$

It also follows from (3), that it does not matter whether we approximate the updated $p(x_0)$ or $m_{f\to X_0}(x_0)$, as we can easily calculate one from the other, given $m_{X_0\to f}(x_0)$. Depending on the factor, the integral in (4) and (1) may be intractable, however, we can approximate the updated $p(x_0)$ by instead sampling from the joint distribution of all neighboring variables $X \in \mathrm{ne}(f)$. If we consider the message passing algorithm, one can easily see that at any point in time during message passing, our arbitrary factor f is equivalent to a full factor graph that consist only of f and its neighboring variables, but with each of the variables initialized to exactly $m_{X\to f}(x)$ by a factor that is only connected to that particular variable, as seen in Figure 2. Thus, the joint distribution $p(x_0, x_1, \ldots, x_V)$ of the variables $X \in \mathrm{ne}(f)$ can be described as

$$p(x_0, x_1, \ldots, x_V) = f(x_0, x_1, \ldots, x_V) \cdot \textstyle\prod_{X\in \mathrm{ne}(f)} m_{X\to f}(x). \tag{5}$$

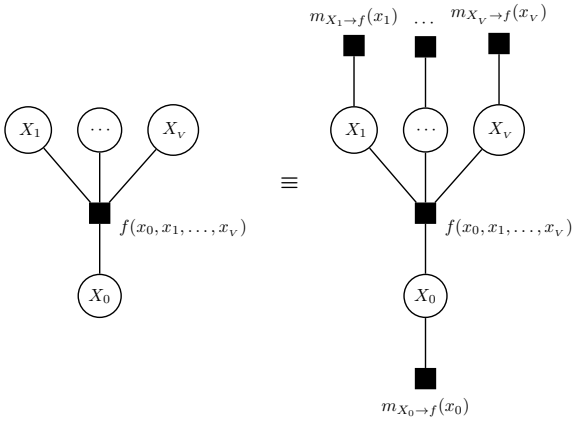


Figure 2: At any given moment in time during message passing, the arbitrary factor f on the left is equivalent to the full factor graph on the right, when calculating the current marginals of its neighboring variables.

This enables us to use the Metropolis-Hastings algorithm to generate samples for each of the marginals by drawing samples from (5), as Metropolis-Hastings can be used to sample from any target distribution with the density $p(x)$, as long as we have a function $f(x) \propto p(x)$. Using these samples, we calculate the empirical mean and variance of the marginal distribution for each neighboring variable and approximate them as Gaussian distributions via moment matching, which is known to minimize the Kullback–Leibler divergence for Gaussians. This approach allows us to approximate the updated marginals arbitrarily well for any given factor, depending on the number of samples that we use. However, the more samples we generate, the higher the computational cost will be, which makes sampling a less than ideal fit for inference with factor graphs. Instead, we propose to generate a wide range of data points of incoming messages $m_{X \to f}$ and their respective updated marginals $p(x)$ for each individual factor that needs to be approximated and fit shallow neural networks to these data points, i.e., matching the moments of the respective distributions.

More concretely, for each of the V variables $X_v \in \mathrm{ne}(f)$ we fit a neural network g_v with parameters θ to a data set of N data points, which are each generated using the sample mean and sample variance of S samples $\boldsymbol{X}_S = \{X_s\}_{s=1}^{S}$, where $X_s \overset{i.i.d.}{\sim} p(x_v)$. The samples are drawn from the joint marginal distribution of all neighboring variables for one combination of concrete incoming messages $m_{n,X_0 \to f}(x_0), \dots, m_{n,X_V \to f}(x_V)$. Consequently, the training of such a neural network, which approximates the updated marginals during message passing, can be described as:

$$\underset{\theta}{arg\,min}\ \frac{1}{N} \sum_{n=1}^{N} Loss\left([E[\boldsymbol{X}_{S,n}], Var[\boldsymbol{X}_{S,n}]], g_v(\theta, m_{n,X_0 \to f}(x_0), \dots, m_{n,X_V \to f}(x_V)) \right) . \tag{6}$$

4. Evaluation

4.1. Setup

Two validate our approach, we chose two known factors that have analytical closed-form solutions and evaluate our approach against these analytical solutions. The sampling in our approach is done via an adaptive version of the Metropolis-Hastings algorithm. Unless otherwise specified, we use 1 million samples as a maximum number of samples per data point, with 50% burn-in. We use independent Gaussian distributions as proposal distributions for the individual variables and we adjust their variances while sampling to achieve an acceptance rate of 0.23. We apply several convergence criteria (see Table 1, Appendix) to early stop the sampling for a given data point. During sampling, we also clamp the sampled marginals, so that for each sampled marginal, the variance is by at least by a factor of $\epsilon = 1e-2$ smaller than that of the respective incoming message from the same variable, to make sure that using it to calculate the respective updated messages $m_{f \to X_v}$ does not result in negative variances.

To generate a data set, we sample inputs, i.e., incoming messages $m_{X_v \to f}$ from uniform distributions (see Table 3, Appendix). On the generated data set we then train shallow neural networks with one hidden layer consisting of 512 neurons and $tanh$ as the activation function. We use z-score normalization for both inputs and outputs based on the training data. The output of each neural network is a 2-dimensional vector, representing the mean and variance of the updated marginal distribution of the respective variable. As the variance needs to be strictly positive and smaller than the variance of the incoming message

for the same variable, we add an additional residual layer. This layer combines the output of the fully-connected layers and the respective input variance to ensure the previously described constraints in (7), where z is the output of the previous layers, x_σ represents the variance of the incoming message $m_{X\to f}(x)$, y_σ is the variance of the predicted marginal $p_X(x)$ and σ is the sigmoid function:

$$y_\sigma = \sigma(z) \cdot x_\sigma \cdot (1 - \varepsilon)\,. \tag{7}$$

ε is fixed at 0.01 and defines the minimum expected difference in variance after a message update. As a loss function we use root mean square error (RMSE). As an optimizer we use Adam with an initial learning rate of 0.001 and reduce it adaptively by 5% (to a minimum of 0.000001) when 10 epochs have passed without a decrease in training loss. We train for a maximum of 2 000 epochs combined with early stopping (Prechelt, 2012) at a patience of 20 epochs.

4.2. Isolated Factors

We choose two known factors for our evaluation, the weighted sum factor $f_{WSF}(x, y, z) = \delta(z - (a \cdot x + b \cdot y))$ and the gaussian mean factor $f_{GM}(x, y) = \mathcal{N}\left(y; x, \beta^2\right)$, which can both be seen in Figure 5, see Appendix.

For both of these models, we generate data sets as described in Section 4.1. We plot the effects of generating the individual data points with varying numbers of samples per data point N_s in Figure 3, as well as the effects of changing the number of data points N in the generated training data sets in Figure 4. The graphs clearly show that higher parameters N and N_s lead to better results. However, this effect is approaching a certain limit and comes at a computational cost. The results suggest a sample size of one million per data point with at least 10 000 data points.

4.3. TrueSkill

TrueSkill, the skill estimation algorithm in Microsoft's Xbox Live gaming service, is a prime example of the scalability of factor graphs (Herbrich et al., 2007). We validate our approach by running a two-player setup (see Appendix, Figure 6), the core building block of the TrueSkill factor graph, using our method to approximate the weighted sum factor. We evaluate the two-player factor graph for various prior distributions. More concretely, for all combinations of the two prior distributions given by all permutations of means and variances where $\mu \in \{1, 6, \ldots, 96\}$ and $\sigma^2 \in \{10, 25, 50, 75, 100\}$. Averaged over all combinations of priors we report a mean absolute error (MAE) of 0.4398 and 1.0421 and a mean absolute percentage error (MAPE) of 0.1793 and 0.0223 for the mean and variance of the updated marginal distributions with a data set size of $5\,000$ respectively. We argue that MAPE is the more relevant metric for the variance, because the same absolute error is less impactful for larger variances, as these are dominated by the narrower distributions during message passing. We chose MAE and MAPE over other metrics such as Kullback–Leibler divergence, that are more often used to compare distributions, because we explicitly aim to match moments as closely as possible to reduce the error that is being introduced by our approximation.

5. Discussion

Limitations and Future Work. The main limitation of our approach so far is the assumption of normally distributed variables and message updates. While this holds for TrueSkill, it is a strong assumption that must be examined closely as our approach is applied to different domains. Using our approach for problems where this assumption does not hold introduces an approximation error that should be investigated further. Ideally, theoretical guarantees for the propagation of this error can be found. The subsequent step in our approach is to expand it to more intricate applications. We focused on the weighted sum factor, as it makes use of the Dirac delta function to model an equation. This concept can be used to model equation systems in the context of physics-informed machine learning, e.g., to model discretised partial differential equations as a strong inductive bias.

Conclusion. To broaden the scope of factor graphs to a wider set of problem domains, we propose a sampling-based framework in combination with shallow neural networks. This enables us to approximate message updates in factor graphs without the need for analytical integration. We validate our approach on factors with known closed-form solutions, directly and in the context of a factor graph. As factor graphs in general offer an efficient inference protocol to obtain probabilistic forecasts, our approach opens up promising avenues for further research on this expanded set of problem domains, such as physics informed machine learning.

References

Dellaert, F. Factor graphs: Exploiting structure in robotics. *Annu. Rev. Control. Robotics Auton. Syst.*, 4: 141–166, 2021. doi: 10.1146/ANNUREV-CONTROL-061520-010504. URL https://doi.org/10.1146/annurev-control-061520-010504.

Espelid, T. O. and Genz, A. Numerical integration: recent developments, software and applications. 2012.

Herbrich, R., Minka, T., and Graepel, T. Trueskill(tm): A bayesian skill rating system. In *Advances in Neural Information Processing Systems 20*, pp. 569–576. MIT Press, 01 2007. URL https://www.microsoft.com/en-us/research/publication/trueskilltm-a-bayesian-skill-rating-system/.

Kschischang, F. R., Frey, B. J., and Loeliger, H. Factor graphs and the sum-product algorithm. *IEEE Trans. Inf. Theory*, 47 (2):498–519, 2001. doi: 10.1109/18.910572. URL https://doi.org/10.1109/18.910572.

Kuck, J., Chakraborty, S., Tang, H., Luo, R., Song, J., Sabharwal, A., and Ermon, S. Belief propagation neural networks. In Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M., and Lin, H. (eds.), *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020. URL https://proceedings.neurips.cc/paper/2020/hash/07217414eb3fbe24d4e5b6cafb91ca18-Abstract.html.

Prechelt, L. Early stopping - but when? In Montavon, G., Orr, G. B., and Müller, K. (eds.), *Neural Networks: Tricks of the Trade - Second Edition*, volume 7700 of *Lecture Notes in Computer Science*, pp. 53–67. Springer, 2012. doi: 10.1007/978-3-642-35289-8_5. URL https://doi.org/10.1007/978-3-642-35289-8_5.

Satorras, V. G. and Welling, M. Neural enhanced belief propagation on factor graphs. In Banerjee, A. and Fukumizu, K. (eds.), *The 24th International Conference on Artificial Intelligence and Statistics, AISTATS 2021, April 13-15, 2021, Virtual Event*, volume 130 of *Proceedings of Machine Learning Research*, pp. 685–693. PMLR, 2021. URL http://proceedings.mlr.press/v130/garcia-satorras21a.html.

Sun, J., Zheng, N., and Shum, H. Stereo matching using belief propagation. *IEEE Trans. Pattern Anal. Mach. Intell.*, 25(7):787–800, 2003. doi: 10.1109/TPAMI.2003.1206509. URL https://doi.org/10.1109/TPAMI.2003.1206509.

Vaske, C. J., House, C., Luu, T., Frank, B., Yeang, C., Lee, N. H., and Stuart, J. M. A factor graph nested effects model to identify networks from genetic perturbations. *PLoS Comput. Biol.*, 5(1), 2009. doi: 10.1371/JOURNAL.PCBI.1000274. URL https://doi.org/10.1371/journal.pcbi.1000274.

Wen, W. and Hsu, L. Towards robust GNSS positioning and real-time kinematic using factor graph optimization. In *IEEE International Conference on Robotics and Automation, ICRA 2021, Xi'an, China, May 30 - June 5, 2021*, pp. 5884–5890. IEEE, 2021. doi: 10.1109/ICRA48506.2021.9562037. URL https://doi.org/10.1109/ICRA48506.2021.9562037.

Zou, Q. and Yang, H. A concise tutorial on approximate message passing. *CoRR*, abs/2201.07487, 2022. URL https://arxiv.org/abs/2201.07487.

6. Appendix

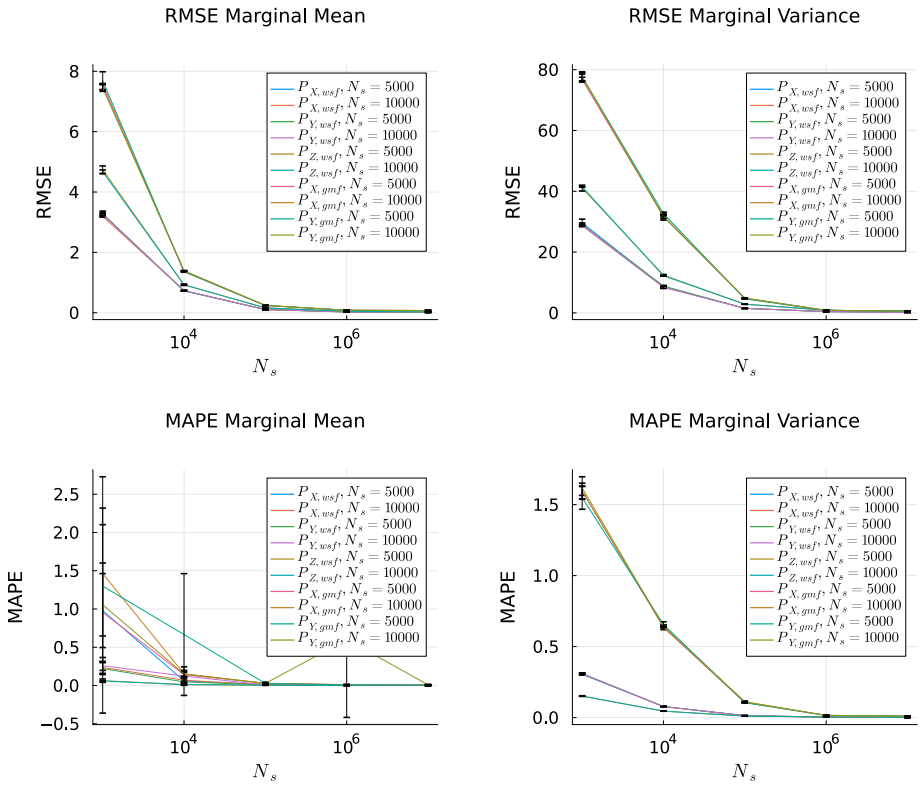


Figure 3: Results of sampled marginals against the analytical ones for the factors GMF and WSF by varying sample size N on the x-axis. The first column refers to the mean of the marginals, the second one to the variance. The x-axis is in log-space for better comparison.

Table 1: Convergence diagnostics used for model evaluation.

DIAGNOSTIC	CRITERION
GEWEKE DIAGNOSTIC	$z < 1.96,\ p > 0.05$
EFFECTIVE SAMPLE SIZE (ESS)	> 100
POTENTIAL SCALE REDUCTION FACTOR ($\hat{R}$)	< 1.1
BAYESIAN FRACTION OF MISSING INFORMATION (BFMI)	> 0.3
MONTE CARLO STANDARD ERROR (MCSE) ON MEAN	< 0.1
MCSE ON STANDARD DEVIATION	< 0.1

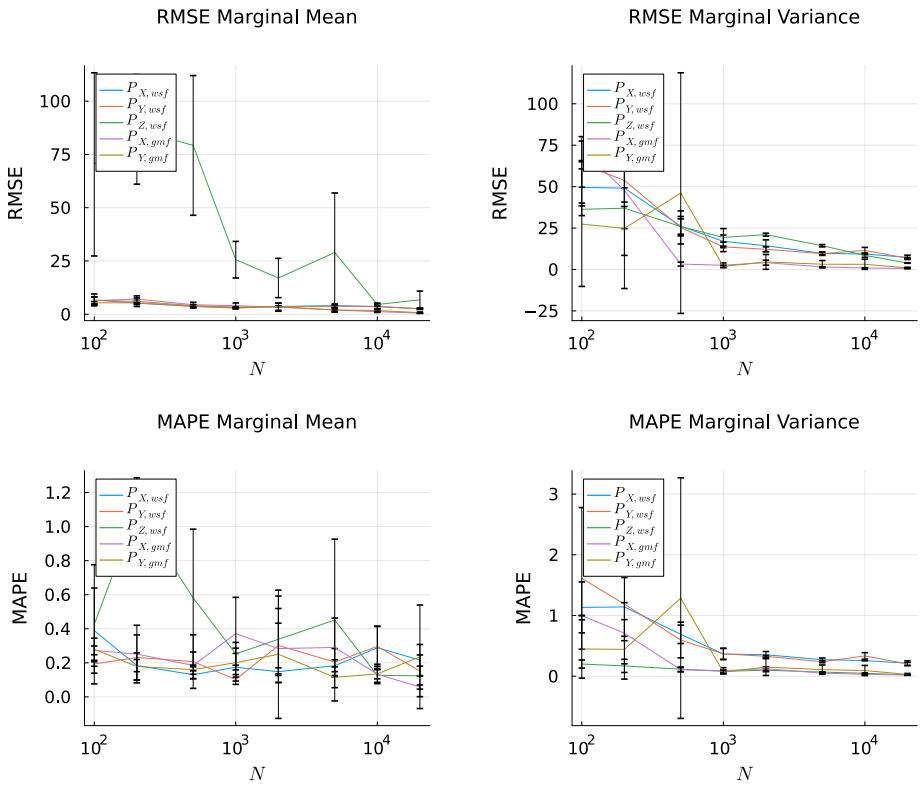
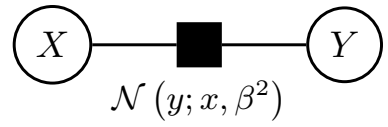
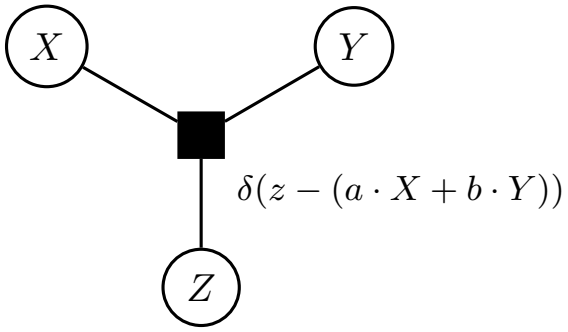


Figure 4: Results of predicted marginals against the analytical ones for the factors GMF and WSF by varying sample size N on the x-axis. The first column refers to the mean of the marginals, the second one to the variance. The x-axis is in log-space for better comparison.



(a) Gaussian Mean Factor



(b) Weighted Sum Factor

Figure 5: The two known factors used in the evaluation.

Table 2: Main parameters and variables of our model.

SYMBOL	DESCRIPTION
V	NUMBER OF VARIABLES, $v = 0, 1, \ldots, V$
N	NUMBER OF DATA POINTS THE NETWORK IS TRAINED ON, $n = 1, \ldots, N$
S	NUMBER OF SAMPLES FROM THE JOINT MARGINAL, $s = 1, \ldots, S$
f	EXEMPLARY FACTOR
X	EXEMPLARY VARIABLE
$p_X(x)$	MARGINAL OF A VARIABLE X WITH VALUES x
$ne(X)$	NEIGHBORING FACTORS OF VARIABLE X IN THE FACTOR GRAPH
$ne(f)$	NEIGHBORING VARIABLES OF FACTOR f IN THE FACTOR GRAPH
$Loss(\cdot, \cdot)$	LOSS FUNCTION (E.G., RMSE) USED FOR TRAINING
g_v	NEURAL NETWORK
θ	WEIGHTS OF A NEURAL NETWORK
a	FIRST PARAMETER IN WEIGHTED SUM FACTOR (TWO SUMMANDS)
b	SECOND PARAMETER IN WEIGHTED SUM FACTOR (TWO SUMMANDS)
β	PARAMETER IN THE GAUSSIAN MEAN FACTOR
ϵ	SMALL CONSTANT FOR SAMPLING AND NN CONSISTENCY, $\epsilon = 0.01$
δ	DIRAC DELTA: $\delta(x) = \lim_{\sigma^2 \to 0} \mathcal{N}(x; 0, \sigma^2)$

Table 3: Sampling ranges for messages and factor parameters.

PARAMETER	SAMPLING DISTRIBUTION
μ_v	$\sim \mathcal{U}(-100,\ 100)$
σ_v	$\sim \mathcal{U}(1,\ 20)$
a	$\sim \mathcal{U}(-10,\ 10)$
b	$\sim \mathcal{U}(-10,\ 10)$
β	$\sim \mathcal{U}(1,\ 10)$

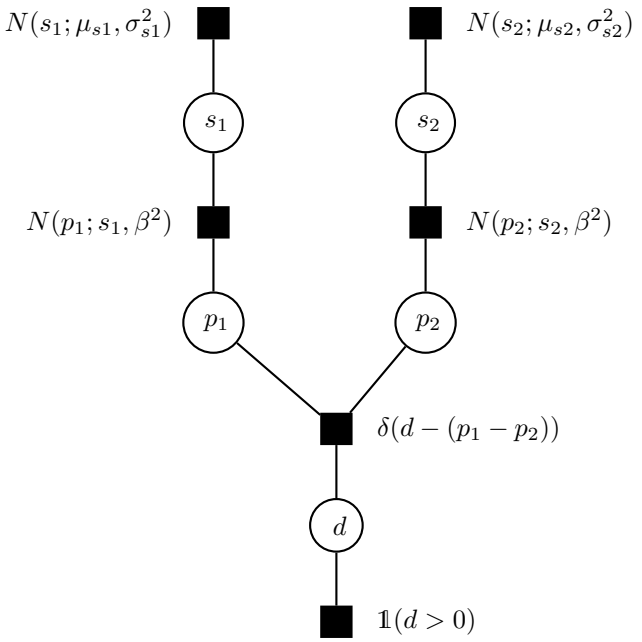


Figure 6: The two-player setup that is the core building block of Trueskill (Herbrich et al., 2007). It models the two players skills s_1 and s_2, their performances p_1 and p_2 and the difference d between them. It assumes match outcome and prior distributions over the players skills. Computing a full pass over the factor graph updates the marginal distributions over the players skills based on the match outcome.