entropy

MDPI

Article

Approximate Bayesian Computation for Discrete Spaces

Ilze A. Auzina [†] and Jakub M. Tomczak [*,†]

Department of Computer Science, Faculty of Science, Vrije Universiteit Amsterdam, De Boelelaan 1111, 1081 HV Amsterdam, The Netherlands; ilze.amanda.auzina@gmail.com

* Correspondence: jmk.tomczak@gmail.com

† These authors contributed equally to this work.

Abstract: Many real-life processes are black-box problems, i.e., the internal workings are inaccessible or a closed-form mathematical expression of the likelihood function cannot be defined. For continuous random variables, likelihood-free inference problems can be solved via Approximate Bayesian Computation (ABC). However, an optimal alternative for discrete random variables is yet to be formulated. Here, we aim to fill this research gap. We propose an adjusted population-based MCMC ABC method by re-defining the standard ABC parameters to discrete ones and by introducing a novel Markov kernel that is inspired by differential evolution. We first assess the proposed Markov kernel on a likelihood-based inference problem, namely discovering the underlying diseases based on a QMR-DTnetwork and, subsequently, the entire method on three likelihood-free inference problems: (i) the QMR-DT network with the unknown likelihood function, (ii) the learning binary neural network, and (iii) neural architecture search. The obtained results indicate the high potential of the proposed framework and the superiority of the new Markov kernel.

Keywords: Approximate Bayesian Computation; differential evolution; MCMC; Markov kernels; discrete state space

check for updates

Citation: Auzina, I.A.; Tomczak, J.M. Approximate Bayesian Computation for Discrete Spaces. *Entropy* **2021**, *23*, 312. https://doi.org/10.3390/e23030312

Academic Editor: Pierre Alquier

Received: 18 February 2021
Accepted: 2 March 2021
Published: 6 March 2021

1. Introduction

In various scientific domains, an accurate simulation model can be designed, yet formulating the corresponding likelihood function remains a challenge. In other words, there is a simulator of a process available that, when provided an input, returns an output, but the inner workings of the process are not analytically available [1–5]. Thus far, the existing tools for solving such problems are typically limited to continuous random variables. Consequently, many discrete problems are reparameterized to continuous ones via, for example, the Gumbel-softmax trick [6] rather than being solved directly. In this paper, we aim at providing a solution to this problem by translating the existing likelihood-free inference methods to discrete space applications.

Commonly, likelihood-free inference problems for continuous data are solved via a group of methods known under the term Approximate Bayesian Computation (ABC) [2,7]. The main idea behind ABC methods is to model the posterior distribution by approximating the likelihood as a fraction of accepted simulated data points from the simulator model, by the use of a distance measure δ and a tolerance value ϵ. The first approach, known as the ABC-rejection scheme, has been successfully applied in biology [8,9], and since, then many alternative versions of the algorithm have been introduced, with the three main groups represented by Markov Chain Monte Carlo (MCMC) ABC [10], Sequential Monte Carlo (SMC) ABC [11], and neural network-based ABC [12,13]. In the current paper, we focus on the MCMC-ABC version [14] for discrete data application, as it can be more readily implemented and the computational costs are lower [15]. Thus, the efficiency of our newly proposed likelihood-free inference method will depend on two parts, namely (i) on the design of the proposal distribution for the MCMC algorithm and (ii) the selected hyperparameter values for the ABC algorithm.

Our main focus is on optimal proposal distribution design as there is no "natural" notion of the search direction and scale for discrete data spaces. Hence, the presented solution is inspired by Differential Evolution (DE) [16], which has been shown to be an effective optimization technique for many likelihood-free (or black-box) problems [17,18]. We propose to define a probabilistic DE kernel for discrete random variables that allows us to traverse the search space without specifying any external parameters. We evaluate our approach on four test-beds: (i) we verify our proposal on a benchmark problem of the QMR-DTnetwork presented by [19]; (ii) we modify the first problem and formulate it as a likelihood-free inference problem; (iii) we assess the applicability of our method for high-dimensional data, namely training binary neural networks on MNIST data; (iv) we apply the proposed approach to Neural Architecture Search (NAS) using the benchmark dataset proposed by [20].

The contribution of the present paper is as follows. First, we introduce an alternative version of the MCMC-ABC algorithm, namely a population-based MCMC-ABC method, that is applicable to likelihood-free inference tasks with discrete random variables. Second, we propose a novel Markov kernel for likelihood-based inference methods in a discrete state space. Third, we present the utility of the proposed approach on three binary problems.

2. Likelihood-Free Inference and ABC

Let $x \in \mathcal{X}$ be a vector of parameters or decision variables, where $\mathcal{X} = \mathbb{R}^D$ or $\mathcal{X} = \{0,1\}^D$, and $y \in \mathbb{R}^M$ is a vector of observable variables. Typically, for a given collection of observations of y, $y_{data} = \{y_n\}_{n=1}^N$, we are interested in solving the following optimization problem (we note that the logarithm does not change the optimization problem, but it is typically used in practice):

$$x^* = \arg\max \ln p(y_{data}|x), \tag{1}$$

where $p(y_{data}|x)$ is the likelihood function. Sometimes, it is more advantageous to calculate the posterior:

$$\ln p(x|y_{data}) = \ln p(y_{data}|x) + \ln p(x) - \ln p(y_{data}), \tag{2}$$

where $p(x)$ denotes the prior over x and $p(y_{data})$ is the marginal likelihood. The posterior $p(x|y_{data})$ could be further used in Bayesian inference.

In many practical applications, the likelihood function is unknown, but it is possible to obtain (approximate) samples from $p(y|x)$ through a simulator. Such a problem is referred to as likelihood-free inference [3] or a black-box optimization problem [1]. If the problem is about finding the posterior distribution over x while only a simulator is available, then it is considered as an Approximate Bayesian Computation (ABC) problem, meaning that $p(y_{data}|x)$ is assumed to be given represented as the simulator.

3. Population-Based MCMC

Typically, a likelihood-free inference problem or an ABC problem is solved through sampling. One of the most well-known sampling methods is the Metropolis–Hastings algorithm [21], where the samples are generated from an ergodic Markov chain, and the target density is estimated via Monte Carlo sampling. In order to speed up the computations, it is proposed to run multiple chains in parallel rather than sampling from a single chain. This approach is known as population-based MCMC methods [22]. A population-based MCMC method operates over a joint state space with the following distribution:

$$p(x_1, \ldots, x_C) = \prod_{c \in \mathcal{C}} p_c(x_c) \tag{3}$$

where $\mathcal{C}$ denotes the population of chains and at least one of $p_c(x_c)$ is equivalent to the original distribution we want to sample from (e.g., the posterior distribution $p(x|y_{data})$).

Given a population of chains, a question of interest is what is the best proposal distribution for an efficient sampling convergence. One approach is parallel tempering. It introduces an additional temperature parameter and initializes each chain at a different temperature [23,24]. However, the performance of the algorithm highly depends on an appropriate cooling schedule rather than a smart interaction between the chains. A different approach proposed by [25] relies on a suitable proposal that is able to adapt the shape of the population at a single temperature. We further expand on this idea by formulating population-based proposal distributions that are inspired by evolutionary algorithms.

3.1. Continuous Case

Reference [26] successfully formulated a new proposal called Differential Evolution Markov Chain (DE-MC) that combines the ideas of differential evolution and population-based MCMC. In particular, he redefined the DE-1 equation [16] by adding noise, ε, to it:

$$x_{new} = x_i + \gamma(x_j - x_k) + \varepsilon, \tag{4}$$

where ε is sampled from a Gaussian distribution, $\gamma \in \mathbb{R}_+$. The created proposal automatically implies the invariance of the underlying distribution, as the reversibility condition is satisfied:

- Reversibility is met, because the suggested proposal could be inverted to obtain x_i.

Furthermore, the created Markov chain is ergodic, as the following two conditions are met:

- Aperiodicity is met, because the Markov chain follows a random walk.
- Irreducibility is solved by applying the noise.

Hence, the resulting Markov chain has a unique stationary distribution. The results presented by [26] indicate an advantage of DE-MC over conventional MCMC with respect to the speed of calculations, convergence, and applicability to multimodal distributions, therefore positioning DE as an optimal method for choosing an appropriate scale and orientation of the jumping distribution for a population-based MCMC.

3.2. Discrete Case

In this paper, we focus on binary variables, because categorical variables could always be transformed to a binary representation. Hence, the most straightforward proposal for binary variables is the independent sampler that utilizes the product of Bernoulli:

$$q(x) = \prod_d B(\theta_d), \tag{5}$$

where $B(\theta_d)$ denotes the Bernoulli distribution with a parameter θ_d. However, the above proposal does not utilize the information available across the population; hence, the performance could be improved by allowing the chains to interact. Exactly this possibility we investigate in the following section.

4. Our Approach

4.1. Markov Kernels

We propose to utilize the ideas outlined by [26], but in a discrete space. For this purpose, we need to relate the DE-1 equation to logical operators, as now the vector x is represented by a string of bits, $\mathcal{X} = \{0,1\}^D$, and properly defined noise. Following [19], we propose to use the *xor*operator between two bits b_1 and b_2:

$$b_1 \otimes b_2 = \begin{cases} 1, & b_1 \neq b_2 \\ 0, & b_1 = b_2 \end{cases} \tag{6}$$

instead of the subtraction in (4). Next, we define a difference between two chains x_i and x_j as $\delta_k = x_i \otimes x_j$ and a set of all possible differences between two chains, $\Delta = \{\delta_k : \forall_{x_i, x_j \in \mathcal{C}}\ \delta_k = x_i \otimes x_j\}$ (a similar construction could be done for the continuous case). We can construct a distribution over δ_k as a uniform distribution:

$$q(\delta|\mathcal{C}) = \frac{1}{|\Delta|} \sum_{\delta_k \in \Delta} \mathbb{I}\left[\delta_k = \delta\right], \tag{7}$$

where $|\Delta|$ denotes the cardinality of Δ and $\mathbb{I}[\cdot]$ is an indicator function such that $\mathbb{I}[\delta_k = \delta] = 1$ if $\delta_k = \delta$ and zero otherwise. Now, we can formulate a binary equivalence of the DE-1 equation by adding a difference drawn from $q(\delta|\mathcal{C})$:

$$x_{new} = x_i \otimes \delta_k. \tag{8}$$

However, the proposal defined in (8) is not a valid ergodic Markov kernel, as is shown in the following Proposition.

Remark 1. *The proposal defined in (8) fulfills reversibility and aperiodicity, but it does not meet the irreducibility requirement.*

Proof. Reversibility is met, as x_i can be re-obtained by applying the difference to the left side of (8). Aperiodicity is met because the general setup of the Markov chain is kept unchanged (it resembles a random walk). However, the operation in (8) is deterministic; thus, it violates the irreducibility assumption. □

The missing property of (8) could be fixed by including the following mutation (*mut*) operation:

$$x_l = \begin{cases} 1 - x_l & \text{if } p_{flip} \geq u \\ x_l & \text{otherwise} \end{cases} \tag{9}$$

where $p_{flip} \in (0, 1)$ corresponds to an independent probability of flipping a bit and $U(0, 1)$ denotes the uniform distribution. Then, the following proposal could be formulated [19] as in Proposition 1.

Proposition 1. *The proposal defined as a mixture $q_{mut+xor}(x|\mathcal{C}) = \pi q_{mut}(x|\mathcal{C}) + (1-\pi) q_{xor}$ $(x|\mathcal{C})$, where $\pi \in (0, 1)$, $q_{mut}(x|\mathcal{C})$ is defined by (9) and $q_{xor}(x|\mathcal{C})$ is defined by (8), is a proper Markov kernel.*

Proof. Reversibility and aperiodicity were shown in Proposition 1. The irreducibility is met, because the *mut* proposal assures that there is a positive transition probability across the entire search space. □

However, we notice that there are two potential issues with the mixture proposal *mut+xor*. First, it introduces another hyperparameter, π, that needs to be determined. Second, improperly chosen π could negatively affect the convergence speed, i.e., a fixed value that is either too frequent or scarce would drastically halt the convergence.

In order to overcome these issues, we propose to apply the *mut* operation in (9) directly to δ_k, in a similar manner as the Gaussian noise is added to $\gamma(x_i - x_j)$ in the proposition of [26]. As a result, we obtain the following proposal:

$$x_{new} = x_i \otimes (mut(\delta_k)). \tag{10}$$

Importantly, this proposal fulfills all requirements for an ergodic Markov kernel.

Proposition 2. *The proposal defined in (10) is a valid ergodic Markov kernel.*

Proof. Reversibility and aperiodicity are met in the same manner as shown in Proposition 1. Adding the mutation operation directly to δ_k allows obtaining all possible states in the discrete space; thus, the irreducibility requirement is met. □

We refer to this new Markov kernel for discrete random variables as the discrete differential evolution Markov chain (*dde-mc*).

4.2. Population-MCMC-ABC

Since we formulated a proposal distribution that utilizes a population of chains, we propose to use a population-based MCMC algorithm for the discrete ABC problems. The core of the MCMC-ABC algorithm is to use a proxy of the likelihood-function defined as an ϵ-ball from the observed data, i.e., $\|y - y_{data}\| \leq \epsilon$, where $\epsilon > 0$ and $\|\cdot\|$ is a chosen metric. The convergence speed and the acceptance rate highly depend on the value of ϵ [27–29]. In this paper, we consider two approaches to determine the ϵ value: (i) by setting a fixed value and (ii) by sampling $\epsilon \sim Exp(\tau)$ [30]. See the Appendix A for details.

A single step of the population-MCMC-ABC algorithm is presented in Algorithm 1. Notice that in Line 5, we take advantage of the symmetricity of all the proposal. Moreover, in the procedure, we skip an outer loop over all chains for clarity. Without loss of generality, we assume a simulator to be a probabilistic program denoted by $\tilde{p}(y|x)$.

Algorithm 1 Population-MCMC-ABC.

```
1: Given $x \in \{0,1\}^D$
2: $x' \sim q(x|\mathcal{C})$                          ▷ Either (5), mut+xor or dde-mc.
3: Simulate $y \sim \tilde{p}(y|x')$.
4: if $\|y - y_{data}\| \leq \epsilon$ then
5:     $\alpha = \min\{1, \frac{p(x')}{p(x)}\}$
6:     $u \sim U(0,1)$
7:     if $u \leq \alpha$ then
8:         $x = x'$
9: return $x$
```

5. Experiments

In order to verify our proposed approach, we use four test-beds:

1. QMR-DT network (likelihood-based case): First, we validate the novel proposal, *dde-mc*, on a problem when the likelihood is known.
2. QMR-DT network (likelihood-free case): Second, we verify the performance of the presented proposal by modifying the first test-bed as a likelihood-free problem.
3. Binarized Neural Network Learning: Third, we investigate the performance of the proposed approach on a high-dimensional problem, namely learning binary neural networks.
4. Neural architecture search: Lastly, we consider a problem of Neural Network Architecture Search (NAS).

With each test-bed, we increase the complexity of the problem. Hence, the number of iterations chosen varies per experiment. The code of the methods and all experiments is available at the following link: https://github.com/IlzeAmandaA/ABCdiscrete (accessed on 5 March 2021).

5.1. A Likelihood-Based QMR-DT Network

5.1.1. Implementation Details

The overall setup was designed as described by [19], i.e., we considered a QMR-DT network model. The architecture of the network follows a two-level or bipartite graphical model, where the top level of the graph contains nodes for the diseases and the bottom level contains nodes for the findings [31]. The following density model captures the relations between the diseases (x) and findings (y):

$$p(y_i = 1|x) = 1 - (1 - q_{i0}) \prod_{l} (1 - q_{il})^{x_l} \tag{11}$$

where y_i is an individual bit of string y and q_{i0} is the corresponding leak probability, i.e., the probability that the finding is caused by means other than the diseases included in the

QMR-DT model [31]. q_{il} is the association probability between disease l and finding i, i.e., the probability that the disease l alone could cause the finding i to have a positive outcome. For a complete inference, the prior $p(x)$ is specified. We follow the assumption made by [19] that the diseases are independent:

$$p(x) = \prod_{l} p_l^{x_l} (1 - p_l)^{(1-x_l)} \tag{12}$$

where p_l is the prior probability for disease l.

We compare the performance of the *dde-mc* kernel to the *mut* proposal, the *mut-xor* proposal, the *mut+crx* proposal (see [19] for details), and the independent sampler (*ind-samp*) as in (5) with sampling probability $\theta_d = 0.5$. We expect the DE-inspired proposals to outperform *ind-samp*, and *dde-mc* to perform similarly, if not surpass, *mut+xor*. Out of the possible parameter settings we investigate, the following population sizes $C = \{8, 12, 24, 40, 60\}$, as well as bit-flipping probabilities $p_{flip} = \{0.1, 0.05, 0.01, 0.005\}$. All experiments were run for 10,000 iterations, as in earlier work by [19], it was observed that the performance differences after 10,000 steps were negligible, and initial experiments revealed that in the current work, all proposals approximately converged at this mark. Furthermore, the performance was validated over 80 random problem instances, and the resulting mean and its standard error are reported.

In this experiment, we used the error that is defined as the average Hamming distance between the real values of x and the most probable values found by the population-MCMC with different proposals. The number of diseases was set to $m = 20$, and the number of findings was $n = 80$.

5.1.2. Results and Discussion

DE-inspired proposals, *dde-mc* and *mut+xor*, are superior to kernels stemming from genetic algorithms or random search, i.e., *mut+crx*, *mut*, and *ind-samp* (Figure 1). In particular, *dde-mc* converged the fastest (see the first 4000 evaluations in Figure 1), suggesting that an update via a single operator rather than a mixture is most effective. As expected, *ind-samp* requires many evaluations to obtain a reasonable performance. Even more so, the obtained difference in wall-clock time between *dde-mc* and *ind-samp* was negligible, 148 versus 117 min, respectively, even though the computational complexity of the new method is theoretically higher: given a search space of $\{0, 1\}^D$, the *dde-mc* proposal costs O(D), while the time complexity of *ind-samp* is O(1).

Based on the obtained results, the subsequent experiments were carried out only with *dde-mc*, *mut+xor*, and *ind-samp* as a baseline. *mut+crx* and *mut* were not selected due to to their very slow convergence with high-dimensional problems.

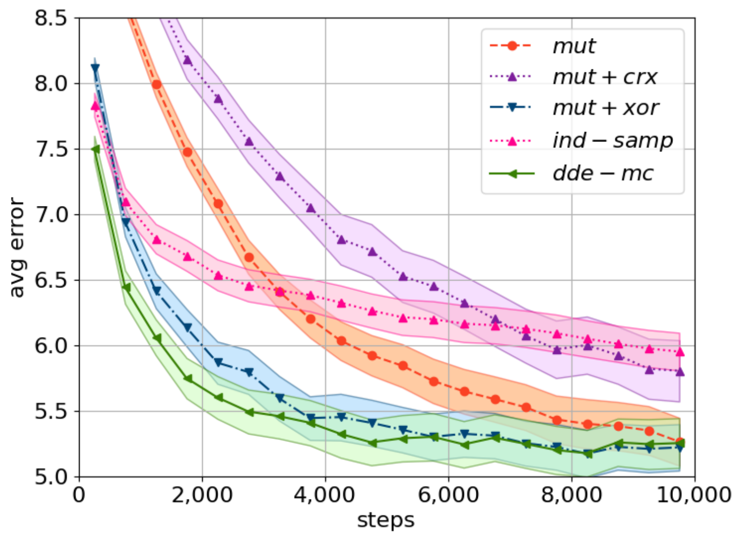

Figure 1. A comparison of the considered proposals using the population average error. The obtained mean and its corresponding standard error (shaded area) across 80 random problem instances are plotted. The following settings were used: $C = 24$, $p_{flip} = 0.01$, $p_{cross} = 0.5$. The corresponding equations for each proposal are as follows: *mut* as in (9), *ind-samp* as in (5), *dde-mc* as in (10), and *mut+xor*, *mut+crx* as in [19].

5.2. *A Likelihood-Free QMR-DT Network*

5.2.1. Implementation Details

In this test-bed, the QMR-DT network is redefined as a simulator model, i.e., the likelihood is assumed to be intractable. The Hamming distance is selected as the distance metric, but due to its equivocal nature for high-dimensional data, the dimensionality of the problem is reduced. In particular, the number of diseases and observations (i.e., findings) are decreased to 10 and 20, respectively, while the probabilities of the network are sampled from a beta distribution, $Beta(0.15, 0.15)$. The resulting network is more deterministic as the underlying density distributions are more peaked; thus, the stochasticity of the simulator is reduced. Multiple tolerance values are investigated to find the optimal settings, $\epsilon = \{0.5, 0.8, 1., 1.2, 1.5, 2.\}$, respectively. The minimal value is chosen to be 0.5 due to variability across the observed data y_{data}. Additionally, we checked sampling ϵ from the exponential distribution. All experiments were cross-evaluated 80 times, and each experiment was initialized with different underlying parameter settings.

5.2.2. Results and Discussion

First, for the fixed value of ϵ, we notice that *dde-mc* converged faster and to a better (local) optimum than *mut+xor*. However, this effect could be explained by a lower dimensionality of the problem compared to the first experiment. Second, utilizing the exponential distribution had a profound positive effect on the convergence rate of both *dde-mc* and *mut+xor* (Figure 2). This confirmed the expectation that an adjustable ϵ has a better balance between exploration and exploitation. In particular, $\epsilon \sim Exp(2)$ brought the best results with *dde-mc* converging the fastest, followed by *mut+xor* and *ind-samp*. This is in line with the corresponding acceptance rates for the first 10,000 iterations (Table 1), i.e., the use of a smarter proposal allows increasing the acceptance probability, as the search space is investigated more efficiently.

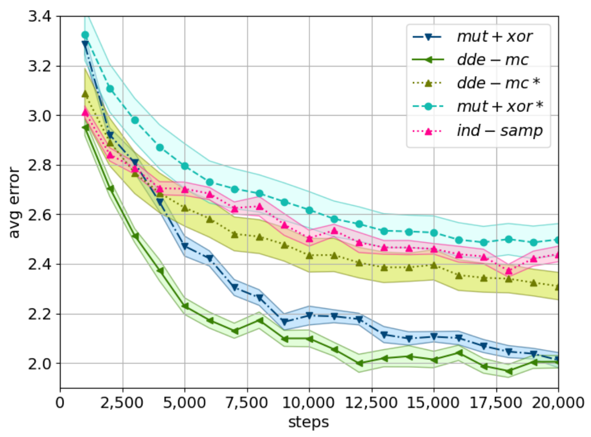

Figure 2. A comparison of the considered proposals using the population error for exponentially adjusted ϵ and the fixed ϵ (indicated by *). The shaded area corresponds to the standard error across 80 random problem instances. The parameter settings are as follows: $C = 24$, $p_{flip} = 0.01$, $\epsilon = 2.0$. The following equations describe the proposal distributions utilized in Algorithm 1: *ind-samp* as in (5), *dde-mc* as in (10), and *mut+xor* as in [19].

Table 1. Percentage of acceptance ratio, α.

Proposal	Mean (std)
dde-mc	24.47 (1.66)
mut+xor	25.81 (1.38)
ind-samp	13.14 (0.33)

Furthermore, the final error obtained by the likelihood-free inference approach is comparable with the results reported for the likelihood-based approach (Figures 1 and 2). This is a positive outcome as any approximation of the likelihood will always be inferior to an exact solution. In particular, the final error obtained by the *dde-mc* proposal is lower; however, this is accounted for by the reduced dimensionality of the problem. Interestingly, despite approximating the likelihood, the computational time only increased twice, while the best performing chain was already identified after 4000 evaluations (Figure 3).

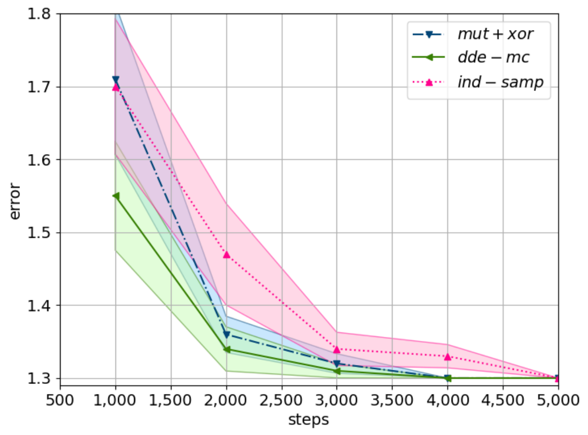

Figure 3. A comparison of the considered proposal using the minimum average error (i.e., the lowest error found by the population) on QMR-DTwith adjusted ϵ. The shaded area corresponds to the standard error across 80 random problem instances. The parameter settings are as follows: $C = 24$, $p_{flip} = 0.01$, $\epsilon = 2.0$. The corresponding equations represent the proposal distributions: *ind-samp* in (5), *dde-mc* in (10), and *mut+xor* as in [19].

Lastly, the obtained results were validated by comparing the true approximate posterior distribution to the approximate posterior distribution of the last five generations of the multi-chain ensemble. In Figure 4, the negative logarithm of the posterior distribution is plotted. The main conclusion is that all proposals converge towards the approximate posterior, yet the obtained distributions are more dispersed.

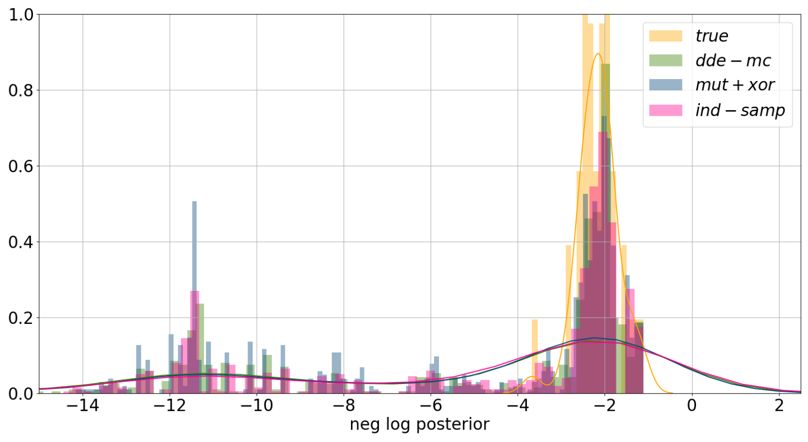

Figure 4. Approximate posterior distribution. The approximate posterior distribution, $p(x|y_{data}) \approx p(y_{data}|x) * p(x)$, was computed using the last population of each chain for all 80 random problem instances. To reconstruct the true posterior, the true underlying parameters were used.

5.3. Binary Neural Networks

5.3.1. Implementation Details

In the following experiment, we aimed at evaluating our approach on a high-dimensional optimization problem. We trained a Binary Neural Network (BinNN) with a single fully-connected hidden layer on the image dataset of ten handwritten digits (MNIST [32]). We used 20 hidden units, and the image was resized from 28px × 28px to 14px × 14px. Furthermore, the image was converted to polar values of +1 or −1, while the network was created in accordance to [33], where the weights and activations of the network were binary, meaning that they were constrained to +1 or −1 as well. We simplified the problem to a binary classification by only selecting two digits from the dataset. As a result, the total number of weights equaled 3940. We used the $tanh$activation function for the hidden units and the sigmoid activation function for the outputs. Consequently, the distance metric becomes the classification error:

$$\|y_{data} - y\| = 1 - \frac{1}{N}\sum_{n=1}^{N} \mathbb{I}[y_n = y_n(x)], \tag{13}$$

where N denotes the number of images, $\mathbb{I}[\cdot]$ is an indicator function, y_n is the true label for the n-th image, and $y_n(x)$ is the n-th label predicted by the binary neural net with weights x.

For the Metropolis acceptance rule, we define a Boltzmann distribution over the prior distribution of the weights x inspired by the work of [34]:

$$p(x) = \frac{h(x)}{\sum_i h(x_i)}, \tag{14}$$

where $h(x) = exp(-\frac{1}{D}\sum_{i=1}^{D} x_i)$ and D denotes the dimensionality of x. As a result, the prior distribution acts as a regularization term as it favors parameter settings with fewer active weights. The distribution is independent of the data y thus, the partition function $\sum_i h(x_i)$ cancels out in the computation of the Metropolis ratio:

$$\alpha = \frac{p(x')}{p(x)} = \frac{h(x')}{h(x)}. \tag{15}$$

The original dataset consists of 60,000 training examples and 10,000 test examples. For our experiment, we selected the digits 0 and 1; hence, the dataset size was reduced to 12,665 training and 2115 test examples. Different tolerance values were investigated to obtain the best convergence, ranging from 0.03 to 0.2, and each experiment was run for at least 200,000 iterations. All experiments were cross-evaluated five times. Lastly, we evaluated the performance by computing both the minimum test error obtained by the final population, as well as the test error obtained by using a Bayesian approach, i.e., we computed the true predictive distribution via majority voting by utilizing an ensemble of models. In particular, we selected the five last updated populations, resulting in $5 \times 24 \times 5$ = 600 models per run, and we repeated this with different seeds 10 times.

Because the classification error function in (13) is non-differentiable, the problem could be treated as a black-box objective. However, we want to emphasize that we do not propose our method as an alternative to gradient-based learning methods. In principle, any gradient-based approach will be superior to a derivative-free method, as what a derivative-free method tries to achieve is to implicitly approximate the gradient [1]. Therefore, the purpose of the presented experiment is not to showcase a state-of-the-art classification accuracy, as that already has been done with gradient-based approaches for BinNN [33], but rather showcase the population-MCMC-ABC applicability to a high-dimensional optimization problem.

5.3.2. Results and Discussion

For the high-dimensional data problem, the *mut+xor* proposal converged the fastest towards the optimal solution in the search space (Figure 5). In particular, the minimum error on the training set was already found after 100,000 iterations, and a tolerance threshold of 0.05 had the best trade-off between the Markov chain error and the likelihood approximation bias.

With respect to the error within the entire population (Figure 6), *dde-mc* converged the fastest, although its performance was on par with *ind-samp*. In general, the drop in performance with respect to the convergence rate of the entire population could be explained by the high dimensionality of the problem, i.e., the higher the dimensionality, the more time is needed for every chain to explore the search space. This observation was confirmed by computing the test error via utilizing all the population members in a majority-voting setting. In particular, the test error based on the ensemble approach was alike across all three proposals, yet the minimum error (i.e., for a single best model) was better for *dde-mc* and *mut+xor* compared to *ind-samp* (Table 2). This result suggests that there seems to be an added advantage of utilizing DE-inspired proposals in faster convergence towards a local optimal solution.

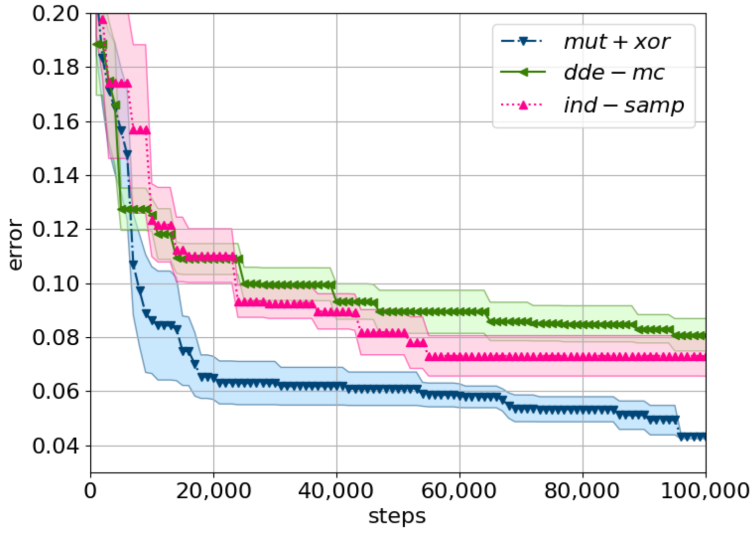

Figure 5. A comparison of the considered proposals using the minimum training error on MNIST. The mean minimum error across five cross-evaluations is plotted with the shaded area corresponding to the standard error. Tolerance is set to $\epsilon = Exp(0.05)$, with the prior and the Metropolis ratio as described in (14) and (15). The following equations describe the proposals: *ind-samp* in (5), *dde-mc* in (10), and *mut+xor* as in [19].

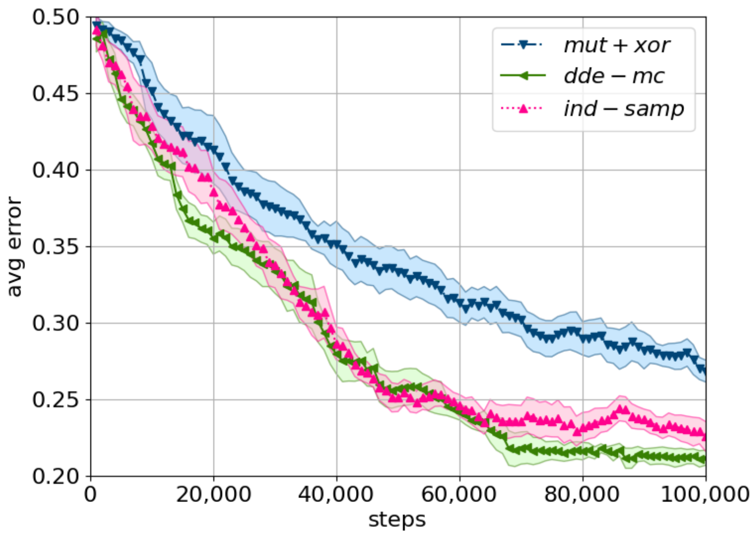

Figure 6. A comparison of the considered proposals using the avg. training error on MNIST. The mean population error across five cross-evaluations is plotted with the shaded area corresponding to the standard error. Tolerance is set to $\epsilon = Exp(0.05)$, with the prior and the Metropolis ratio as described in (14) and (15). The following equations describe the proposals: *ind-samp* in (5), *dde-mc* in (10), and *mut+xor* as in [19].

Table 2. Test error of BinNN on MNIST.

Proposal	**Error (ste)**	
	Single Best	***Ensemble***
dde-mc	0.045 (0.002)	0.013 (0.001)
mut+xor	0.046 (0.002)	0.014 (0.002)
ind-samp	0.051 (0.002)	0.012 (0.001)

5.4. Neural Architecture Search

5.4.1. Implementation Details

In the last experiment, we aimed at investigating whether the proposed approach is applicable for efficient neural architecture search. In particular, we made use of the NAS-Bench-101 dataset, the first public architecture dataset for NAS research [20]. The dataset is represented as a table, which maps neural architectures to their training and evaluations metrics, and as such, it represents an efficient solution for querying different neural topologies. Each topology is captured by a directed acyclic graph represented by an adjacency matrix. The number of vertices was set to seven, while the maximum amount of edges was nine. Apart from these restrictions, we limited the search space by constricting the possible operations for each vertex. Consequently, the simulator was captured by querying the dataset, while the distance metric now was simply the validation error. The prior distribution was kept the same as for the previous experiment.

Every experiment was run for at least 120,000 iterations, with five cross-evaluations. To find the optimal performance, the following tolerance threshold values were investigated $\epsilon = \{0.01, 0.1, 0.2, 0.3\}$. As we are approaching the problem as an optimization task, the aim is to find a chain with the lowest test error, rather than covering the entire distribution. Therefore, to evaluate the performance, we plot the minimum error obtained through the training process, as well as the lowest test error obtained by the final population.

5.4.2. Results and Discussion

dde-mc identified the best solution the fastest with ϵ set to $\epsilon \sim Exp(0.2)$ (Figure 7). The corresponding test error is reported in Table 3, and it follows the same pattern, namely *dde-mc* is superior. Interestingly, here, the *mut+xor* proposal performed almost on par with the *ind-samp* proposal for the first 10,000 iterations, and then, both methods converged to almost the same result. Our proposed Markov kernel obtained again the best result, and also it was the fastest.

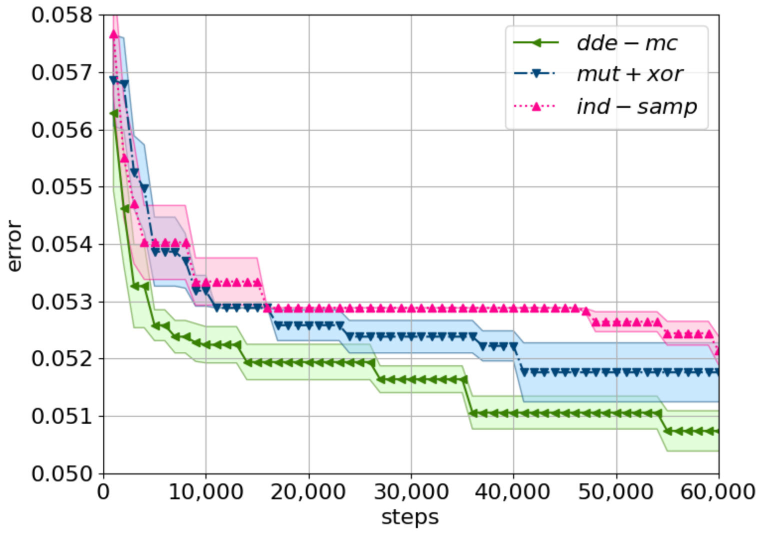

Figure 7. A comparison of the considered proposals using the minimum training error on NAS-Bench-101. The mean minimum error with its corresponding standard error (shaded area) across five cross-evaluations is plotted. Tolerance is set to $\epsilon = Exp(0.2)$. The prior distribution is as described in (14), with the corresponding Metropolis ratio (15). The following equations describe the proposals: *ind-samp* in (5), *dde-mc* in (10), and *mut+xor* as in [19].

Table 3. Test error on NAS-Bench-101.

Proposal	**Error (ste)**
dde-mc	0.058 (0.001)
mut+xor	0.060 (<0.001)
ind-samp	0.062 (<0.001)

6. Conclusions

In this paper, we note that there is a gap in the available methods for likelihood-free inference on discrete problems. We propose to utilize ideas known from evolutionary computing similarly to [26], in order to formulate a new Markov kernel, *dde-mc*, for a population-based MCMC-ABC algorithm. The obtained results suggest that the newly designed proposal is a promising and effective solution for intractable problems in a discrete space.

Furthermore, Markov kernels based on differential evolution are also effective to traverse a discrete search space. Nonetheless, great attention has to be paid to the choice of the tolerance threshold for the MCMC-ABC methods. In other words, if the tolerance is set too high, then the performance of the DE-based proposals drops to that of an independent sampler, i.e., the error of the Markov chain is high. For high-dimensional problems, the proposed kernel seems to be most promising; however, its population error becomes similar to that of *ind-samp*. This is accounted for by the fact that for high dimensions, it takes more time for the entire population to converge.

In conclusion, we would like to highlight that the present work offers new research directions:

- Alternative ABC algorithms like SMC should be further investigated.
- In this work, we focused on calculating distances in the data space. However, utilizing summary statistics is almost an obvious direction for future work.
- As the whole algorithm is based on logical operators and the input variables are also binary, the algorithm could be encoded using only bits, thus saving considerable amounts of memory storage. Consequently, any matrix multiplication could be replaced by an XNORoperation followed by a sum, thus reducing the computation costs and possibly allowing implementing the algorithm on relatively simple devices. Therefore, a natural consequence of this work would be a direct hardware implementation of the proposed methods.
- In this paper, we outline a number of potential applications of the presented methodology and indicate that the obtained results are of great practical potential. From the optimization perspective, a discrete ABC gives an opportunity to solve a problem in a principled manner. This is extremely important for applications associated with deep learning, e.g., NAS [20,35], neural network quantization, and learning binary neural networks, but also in other domains like topology or relationship discovery in biological networks (e.g., Boolean networks) [36]. Moreover, ABC as a Bayesian framework allows calculating model evidence that is crucial for model selection. In practice, very often, a problem is of combinatorial (discrete) nature, e.g., contamination control or pest control [35]. Therefore, our approach could be seemingly applied without the necessity of dequantizing a problem.

Author Contributions: Conceptualization, J.M.T.; methodology, I.A.A. and J.M.T.; software, I.A.A.; validation, I.A.A.; formal analysis, I.A.A. and J.M.T.; investigation, I.A.A. and J.M.T.; writing—original draft preparation, I.A.A. and J.M.T.; writing—review and editing, I.A.A. and J.M.T.; visualization, I.A.A.; supervision, J.M.T. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The code used in this paper is available at: https://github.com/IlzeAmandaA/ABCdiscrete (accessed on 5 March 2021).

Acknowledgments: We would like to thank [37] for the usage of the Distributed ASCI Supercomputer (DAS).

Conflicts of Interest: The authors declare no conflict of interest. The funders had no role in the design of the study; in the collection, analyses, or interpretation of data; in the writing of the manuscript, or in the decision to publish the results.

Abbreviations

ABC	Approximate Bayesian Computation
SMC	Sequential Monte Carlo
DE	Differential Evolution
MCMC	Markov Chain Monte Carlo
DE-MC	Differential Evolution Markov Chain
mut+xor	a mixture of a mutation-based proposal and an xor-based proposal
dde-mc	discrete differential evolution Markov chain
Population-MCMC-ABC	a population-based MCMC ABC
NAS	Neural Network Architecture Search
ind-samp	independent sampler
mut+crx	a mixture of a mutation-based proposal and a cross-over-based proposal
BinNN	a binary neural network

Appendix A. ϵ Determination

The choice of ϵ defines which data points are going to be accepted; as such, it implicitly models the likelihood. Setting the value too high will result in a biased estimate; however, it will improve the performance of Monte Carlo as more samples are utilized per unit time. Hence, as [4] already has stated: "the goal is to find a good balance between the bias and the Monte Carlo error".

Appendix A.1. Fixed ϵ

The first group of tolerance selection methods are all based on a fixed ϵ value. The possible approaches are summarized as follows:

- Determine a desirable acceptance ratio: For example, define a proportion, 1%, of the simulated samples that should be accepted ([2]).
- Re-use the generated samples: Determine the optimal cutoff value by a leave-one-out cross-validation approach of the underlying parameters of the generated simulations. In particular, minimize the Root Mean Squared Error (RMSE) for the validation parameter values [28].
- Use a pilot run to tune: Based on the rates of convergence [27], define fixed alterations to the initial tolerance value in order to either increase the number of accepted samples, reduce the mean-squared error, or increase the (expected) running time.
- Set ϵ to be proportional to $N_s^{-1/(d+5)}$: where d is the number of dimensions (for a complete overview, see [4]).

Nonetheless, setting ϵ to a fixed value hinders the convergence as it clearly is a suboptimal approach due to its static nature. Ideally, we want to promote exploration at the beginning of the algorithm and, subsequently, move towards exploitation, hence alluding to the second group of tolerance selection methods: adaptive ϵ.

Appendix A.2. Adaptive ϵ

In general, the research on adaptive tolerance methods for MCMC-ABC is very limited as traditionally, adaptive tolerance is seen as part of SMC-ABC. In the current literature, two adaptive tolerance methods for MCMC-ABC are mentioned:

- An exponential cooling scheme: Reference [29] suggested using an exponential temperature scheme combined with a cooling scheme for the covariance matrix $\sum_t$.
- Sample from the exponential distribution: Similarly, Reference [30] assumed a pseudo-prior for $\epsilon : \pi(\epsilon)$, where $\pi(\epsilon) \sim Exp(\tau)$ and $\tau = 1/10$, thus allowing occasionally generating larger tolerance values to adjust mixing.

In order to establish a clear baseline for MCMC-ABC in a discrete space, we decided to implement both fixed and adaptive ϵ. Such an approach allows us to evaluate what is the effect of an adaptive ϵ in comparison to a fixed ϵ in a discrete space, as well as to compare how well our observations are in line with the observations drawn in a continuous space.

References

1. Audet, C.; Hare, W. *Derivative-Free and Blackbox Optimization*; Springer: Berlin/Heisenberg, Germany, 2017.
2. Beaumont, M.A.; Zhang, W.; Balding, D.J. Approximate Bayesian computation in population genetics. *Genetics* **2002**, *162*, 2025–2035.
3. Cranmer, K.; Brehmer, J.; Louppe, G. The frontier of simulation-based inference. *Proc. Natl. Acad. Sci. USA* **2020**, 117, 30055–30062. [CrossRef]
4. Lintusaari, J.; Gutmann, M.U.; Dutta, R.; Kaski, S.; Corander, J. Fundamentals and recent developments in approximate Bayesian computation. *Syst. Biol.* **2017**, *66*, e66–e82. [CrossRef]
5. Toni, T.; Welch, D.; Strelkowa, N.; Ipsen, A.; Stumpf, M.P. Approximate Bayesian computation scheme for parameter inference and model selection in dynamical systems. *J. R. Soc. Interface* **2009**, *6*, 187–202. [CrossRef]
6. Jang, E.; Gu, S.; Poole, B. Categorical reparameterization with gumbel-softmax. *arXiv* **2016**, arXiv:1611.01144.
7. Alquier, P. Approximate Bayesian Inference. *Entropy* **2020**, *22*, 1272. [CrossRef]
8. Pritchard, J.K.; Seielstad, M.T.; Perez-Lezaun, A.; Feldman, M.W. Population growth of human Y chromosomes: A study of Y chromosome microsatellites. *Mol. Biol. Evol.* **1999**, *16*, 1791–1798. [CrossRef]
9. Tavaré, S.; Balding, D.J.; Griffiths, R.C.; Donnelly, P. Inferring coalescence times from DNA sequence data. *Genetics* **1997**, *145*, 505–518. [CrossRef]
10. Marjoram, P.; Molitor, J.; Plagnol, V.; Tavaré, S. Markov chain Monte Carlo without likelihoods. *Proc. Natl. Acad. Sci. USA* **2003**, *100*, 15324–15328. [CrossRef] [PubMed]
11. Beaumont, M.A.; Cornuet, J.M.; Marin, J.M.; Robert, C.P. Adaptive approximate Bayesian computation. *Biometrika* **2009**, *96*, 983–990. [CrossRef]
12. Papamakarios, G. Neural density estimation and likelihood-free inference. *arXiv* **2019**, arXiv:1910.13233.
13. Papamakarios, G.; Sterratt, D.; Murray, I. Sequential neural likelihood: Fast likelihood-free inference with autoregressive flows. In Proceedings of the The 22nd International Conference on Artificial Intelligence and Statistics, Okinawa, Japan, 16–19 April 2019; pp. 837–848.
14. Andrieu, C.; Roberts, G.O. The pseudo-marginal approach for efficient Monte Carlo computations. *Ann. Stat.* **2009**, *37*, 697–725. [CrossRef]
15. Jasra, A.; Stephens, D.A.; Holmes, C.C. On population-based simulation for static inference. *Stat. Comput.* **2007**, *17*, 263–279. [CrossRef]
16. Storn, R.; Price, K. Differential evolution–a simple and efficient heuristic for global optimization over continuous spaces. *J. Glob. Optim.* **1997**, *11*, 341–359. [CrossRef]
17. Vesterstrom, J.; Thomsen, R. A comparative study of differential evolution, particle swarm optimization, and evolutionary algorithms on numerical benchmark problems. In Proceedings of the 2004 Congress on Evolutionary Computation (IEEE Cat. No. 04TH8753), Portland, OR, USA, 19–23 June 2004; Volume 2, pp. 1980–1987.
18. Maučec, M.S.; Brest, J.; Bošković, B.; others. Improved differential evolution for large-scale black-box optimization. *IEEE Access* **2018**, *6*, 29516–29531.
19. Strens, M. Evolutionary MCMC sampling and optimization in discrete spaces. In Proceedings of the 20th International Conference on Machine Learning (ICML-03), Washington, DC, USA, 21–24 August 2003; pp. 736–743.
20. Ying, C.; Klein, A.; Christiansen, E.; Real, E.; Murphy, K.; Hutter, F. Nas-bench-101: Towards reproducible neural architecture search. In Proceedings of the International Conference on Machine Learning, Long Beach, CA, USA, 10–15 June 2019; pp. 7105–7114.
21. Metropolis, N.; Ulam, S. The monte carlo method. *J. Am. Stat. Assoc.* **1949**, *44*, 335–341. [CrossRef] [PubMed]
22. Iba, Y. Population monte carlo algorithms. *Trans. Jpn. Soc. Artif. Intell.* **2001**, *16*, 279–286. [CrossRef]

23. Hukushima, K.; Nemoto, K. Exchange Monte Carlo method and application to spin glass simulations. *J. Phys. Soc. Jpn.* **1996**, *65*, 1604–1608. [CrossRef]
24. Liang, F.; Wong, W.H. Evolutionary Monte Carlo: Applications to C p model sampling and change point problem. *Stat. Sin.* **2000**, *10*, 317–342.
25. Strens, M.J.; Bernhardt, M.; Everett, N. Markov Chain Monte Carlo Sampling Using Direct Search Optimization. In Proceedings of the Nineteenth International Conference on Machine Learning, ICML, Sydney, Australia, 8–12 July 2002; pp. 602–609.
26. Ter Braak, C.J. A Markov Chain Monte Carlo version of the genetic algorithm Differential Evolution: Easy Bayesian computing for real parameter spaces. *Stat. Comput.* **2006**, *16*, 239–249. [CrossRef]
27. Barber, S.; Voss, J.; Webster, M. The rate of convergence for approximate Bayesian computation. *Electron. J. Stat.* **2015**, *9*, 80–105. [CrossRef]
28. Faisal, M.; Futschik, A.; Hussain, I. A new approach to choose acceptance cutoff for approximate Bayesian computation. *J. Appl. Stat.* **2013**, *40*, 862–869. [CrossRef]
29. Ratmann, O.; Jørgensen, O.; Hinkley, T.; Stumpf, M.; Richardson, S.; Wiuf, C. Using likelihood-free inference to compare evolutionary dynamics of the protein networks of H. pylori and P. falciparum. *PLoS Comput. Biol.* **2007**, *3*, e230. [CrossRef] [PubMed]
30. Bortot, P.; Coles, S.G.; Sisson, S.A. Inference for stereological extremes. *J. Am. Stat. Assoc.* **2007**, *102*, 84–92. [CrossRef]
31. Jaakkola, T.S.; Jordan, M.I. Variational probabilistic inference and the QMR-DT network. *J. Artif. Intell. Res.* **1999**, *10*, 291–322. [CrossRef]
32. LeCun, Y.; Bottou, L.; Bengio, Y.; Haffner, P. Gradient-based learning applied to document recognition. *Proc. IEEE* **1998**, *86*, 2278–2324. [CrossRef]
33. Courbariaux, M.; Hubara, I.; Soudry, D.; El-Yaniv, R.; Bengio, Y. Binarized neural networks: Training deep neural networks with weights and activations constrained to+ 1 or-1. *arXiv* **2016**, arXiv:1602.02830.
34. Tomczak, J.M.; Zieba, M. Probabilistic combination of classification rules and its application to medical diagnosis. *Mach. Learn.* **2015**, *101*, 105–135. [CrossRef]
35. Oh, C.; Tomczak, J.M.; Gavves, E.; Welling, M. Combinatorial Bayesian optimization using the graph cartesian product. In Proceedings of the Conference on Neural Information Processing Systems, Vancouver, BC, Canada, 8–14 December 2019.
36. Friedman, N.; Linial, M.; Nachman, I.; Pe'er, D. Using Bayesian networks to analyze expression data. *J. Comput. Biol.* **2000**, *7*, 601–620. [CrossRef] [PubMed]
37. Bal, H.; Epema, D.; de Laat, C.; van Nieuwpoort, R.; Romein, J.; Seinstra, F.; Snoek, C.; Wijshoff, H. A medium-scale distributed system for computer science research: Infrastructure for the long term. *Computer* **2016**, *49*, 54–63. [CrossRef]